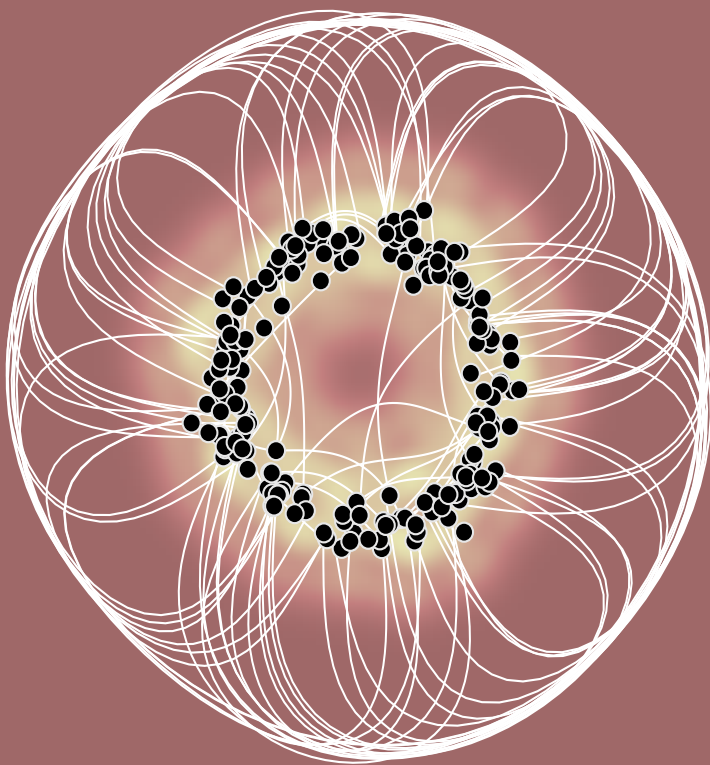


*A guide to  
differential geometry for  
machine learning*



Søren Hauberg



# Contents

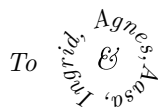
|                                                                                 |         |
|---------------------------------------------------------------------------------|---------|
| <i>Author's note</i>                                                            | Page 5  |
| <i>In which the author states his intent.</i>                                   |         |
| <i>Mathematical notation</i>                                                    | Page 6  |
| <i>In which equations are given without context.</i>                            |         |
| Chapter 1                                                                       |         |
| <i>Introduction</i>                                                             | Page 9  |
| <i>In which we meet our first manifolds, and the story begins.</i>              |         |
| Chapter 2                                                                       |         |
| <i>Making maps</i>                                                              | Page 17 |
| <i>In which we pause to look at the world.</i>                                  |         |
| Chapter 3                                                                       |         |
| <i>Embedded and immersed manifolds</i>                                          | Page 21 |
| <i>In which we begin to observe manifolds.</i>                                  |         |
| Chapter 4                                                                       |         |
| <i>Distances on manifolds</i>                                                   | Page 29 |
| <i>In which we start being able to tell what is close and what is far away.</i> |         |
| Chapter 5                                                                       |         |
| <i>Riemannian metrics</i>                                                       | Page 38 |
| <i>In which we measure from the inside out.</i>                                 |         |
| Chapter 6                                                                       |         |
| <i>Examples of learned metrics</i>                                              | Page 48 |
| <i>In which we take an exemplary intermission.</i>                              |         |

|                                                                         |          |
|-------------------------------------------------------------------------|----------|
| Chapter 7                                                               |          |
| <i>Geodesics</i>                                                        | Page 53  |
| <i>In which shortcuts are repeatedly taken.</i>                         |          |
| Chapter 8                                                               |          |
| <i>Computing geodesics numerically</i>                                  | Page 66  |
| <i>In which we actually do something practical.</i>                     |          |
| Chapter 9                                                               |          |
| <i>Arithmetic on Riemannian manifolds</i>                               | Page 77  |
| <i>In which we joyfully rediscover plus and minus.</i>                  |          |
| Chapter 10                                                              |          |
| <i>Integration on manifolds</i>                                         | Page 82  |
| <i>In which the metric speaks volumes about space.</i>                  |          |
| Chapter 11                                                              |          |
| <i>Submanifolds</i>                                                     | Page 90  |
| <i>In which recursion takes over (again).</i>                           |          |
| Chapter 12                                                              |          |
| <i>Basics of topology</i>                                               | Page 94  |
| <i>In which we forget about appearances and go for inner qualities.</i> |          |
| Chapter 13                                                              |          |
| <i>Statistical identifiability</i>                                      | Page 100 |
| <i>In which we're all unique.</i>                                       |          |
| Chapter 14                                                              |          |
| <i>Statistical manifolds</i>                                            | Page 105 |
| <i>In which we are probably in the ambient space.</i>                   |          |
| Chapter 15                                                              |          |
| <i>Noisy manifolds</i>                                                  | Page 118 |
| <i>In which data ruins the party only to randomly save it again.</i>    |          |



|                                                               |          |
|---------------------------------------------------------------|----------|
| Chapter 16                                                    |          |
| <i>Expected Gaussian manifolds</i>                            | Page 127 |
| <i>In which great expectations aren't all that great.</i>     |          |
| Chapter 17                                                    |          |
| <i>Spacetime manifolds</i>                                    | Page 143 |
| <i>In which time goes by.</i>                                 |          |
| Chapter 18                                                    |          |
| <i>Parameter manifolds</i>                                    | Page 148 |
| <i>In which everything is latent and the distant is near.</i> |          |
| Chapter 19                                                    |          |
| <i>Parting remarks &amp; further reading</i>                  | Page 159 |
| <i>In which our story ends, while many new begin.</i>         |          |
| Appendix A                                                    |          |
| <i>Ordinary differential equations</i>                        | Page 163 |
| <i>In which we take one step at a time.</i>                   |          |
| Appendix B                                                    |          |
| <i>Functional derivatives</i>                                 | Page 167 |
| <i>In which the derivatives are rather high-dimensional.</i>  |          |
| <i>Bibliography</i>                                           | Page 169 |
| <i>In which references are given.</i>                         |          |
| <i>Index</i>                                                  | Page 174 |
| <i>In which we get self-referential.</i>                      |          |

## *Author's note*

To  Agnes, Asa, & Ingrid

A former professor of mine once described a popular textbook on differential geometry as ‘toilet reading’. Admittedly, he used more crude terms. This description meant that the text was so easily accessible that he could read the book while frequenting the bathroom. That description was *not* meant as a compliment. . .

This brief book aims to be ‘toilet reading’ for machine learners. Differential geometry is all too often presented abstractly with the driving intuitions hidden well away. Instead, I have tried to make this text as easy as possible to digest. . .

That said, I assume you have some experience with machine learning and the standard mathematical tools associated with this field. That is, you should be familiar with elementary calculus, linear algebra, and probability theory.

## Mathematical notation

I have limited the mathematical rigor of this text, with the aim of only providing enough detail for an operational understanding of the topic. For a deeper dive, existing textbooks are available. We cannot, and should not, completely skip mathematics, so some notation is in order.

In general, boldface variables of lower case, e.g.,  $\mathbf{x}$ , are vectors, while matrices are boldface capital letters, e.g.,  $\mathbf{J}$ . Scalars are lowercase, such as  $a$ . Different spaces are denoted with calligraphic letters, such as  $\mathcal{M}$ , or ‘mathematical bold’, such as  $\mathbb{R}$ , which denotes the set of real numbers. The table below gives a list of commonly used symbols and variables. These will be defined throughout the text and are only listed here for reference.

|                                                         |                                                                                                                                                                    |
|---------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\mathbb{R}^D$                                          | The space of $D$ -dimensional real vectors.                                                                                                                        |
| $\Omega \subset \mathbb{R}^d$                           | The input domain of a manifold.                                                                                                                                    |
|                                                         | In machine learning jargon: the latent space.                                                                                                                      |
| $\mathcal{M}$                                           | A manifold, usually $\mathcal{M} = f(\Omega) \subset \mathbb{R}^D$ .                                                                                               |
| $T_{\mathbf{x}}$                                        | The tangent space at $\mathbf{x} \in \Omega$ .                                                                                                                     |
| $\text{Sym}_+^{d \times d}$                             | The space of symmetric positive $d \times d$ matrices.                                                                                                             |
| $\mathbf{x}_n, \mathbf{x}_m \in \Omega$                 | Points in the input domain.                                                                                                                                        |
|                                                         | In machine learning jargon: latent variables.                                                                                                                      |
| $\mathbf{y}_n, \mathbf{y}_m \in \mathbb{R}^D$           | Point in the ambient space, usually $\mathbf{y} = f(\mathbf{x})$ .                                                                                                 |
|                                                         | These may be thought of as observational data.                                                                                                                     |
| $\delta_n, \delta_m$                                    | Infinitesimals, i.e. very small vectors.                                                                                                                           |
| $[\mathbf{x}]_i$                                        | The $i^{\text{th}}$ element of vector $\mathbf{x}$ .                                                                                                               |
|                                                         | Similar notation is used for matrices.                                                                                                                             |
| $\mathbf{J}_{\mathbf{x}} \in \mathbb{R}^{D \times d}$   | The Jacobian matrix of $f$ at $\mathbf{x} \in \Omega$ .                                                                                                            |
| $\mathbf{G}_{\mathbf{x}} \in \text{Sym}_+^{d \times d}$ | The Riemannian metric $\mathbf{J}_{\mathbf{x}}^T \mathbf{J}_{\mathbf{x}}$ .                                                                                        |
| $c$                                                     | A curve of the form $c : [0, 1] \rightarrow \Omega$ . The curve is indexed by $t$ , and we write $c_t = c(t)$ to determine the point along the curve at time $t$ . |
| $\dot{c}$                                               | The derivative of the curve $c$ , i.e. $\dot{c} = dc/dt$ .                                                                                                         |

|                                           |                                                                              |
|-------------------------------------------|------------------------------------------------------------------------------|
| $c_{nm}$                                  | The shortest path (geodesic) connecting $\mathbf{x}_n$ and $\mathbf{x}_m$ .  |
| $\ \cdot\ $                               | The norm of a vector.                                                        |
| $\langle\cdot,\cdot\rangle$               | The inner product between vectors.                                           |
| $\ \cdot\ _{\mathbf{x}}$                  | The norm of a vector with respect to the metric at $\mathbf{x} \in \Omega$ . |
| $\langle\cdot,\cdot\rangle_{\mathbf{x}}$  | The inner product with respect to the metric at $\mathbf{x} \in \Omega$ .    |
| $\text{Length}[c]$                        | The length of a curve $c$ .                                                  |
| $\mathcal{E}[c]$                          | The energy of a curve $c$ .                                                  |
| $\text{dist}(\cdot,\cdot)$                | The geodesic distance between two points.                                    |
| $\text{Log}_{\mathbf{x}_n}(\mathbf{x}_m)$ | The logarithm map of $\mathbf{x}_m$ evaluated at $\mathbf{x}_n$ .            |
| $\text{Exp}_{\mathbf{x}}(\mathbf{v})$     | The exponential map of $\mathbf{v}$ starting from $\mathbf{x}$ .             |

# Chapter 1

## *Introduction*

*In which we meet our first manifolds,  
and the story begins.*

The figure below shows a manifold. It is a collection of points in  $\mathbb{R}^3$  that are somehow connected to form a smooth object. It is these kinds of objects we will investigate together to build useful models derived from observed data. You may notice that data does not appear in the figure — don't worry we will eventually get to data, but it will not be our initial focus.

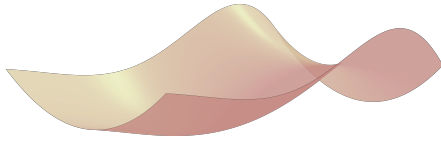


FIG. 1.1. A simple two-dimensional manifold embedded in  $\mathbb{R}^3$ .

Geometry tells us how to measure; statistics rely on our choice of measure. This book touches on some geometric foundations that are useful when constructing statistical models. In particular, we will work towards understanding *Riemannian manifolds* as they can appear in machine learning. We won't focus on how to learn models, but rather on what to do with models once they have been learned.

The existing literature is focused on the mathematics of geometry, often presented abstractly. From a practical perspective, this is unwarranted as the topic is often simple and intuitive. This book aims to be a starting ground that establishes the most basic constructions to preserve intuition. As such, this is by no means a complete exposition, but rather an opening that allows the reader to get started.

On that note, let's get started...

### 1.1. Example: Identifiability issues

In *latent variable models*, we describe our hypothesis of how the observed data was generated through the use of an underlying ‘hidden’ quantity. For example, we may assume that an observation  $\mathbf{y}$  is generated as

$$\mathbf{y} = f(\mathbf{x}), \quad (1.1)$$

where both the function  $f$  and its input  $\mathbf{x}$  are ‘hidden’, i.e. not observed. In probabilistic terms, we would perhaps prefer to work with a conditional distribution rather than a function, such that the latent variable model becomes

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}. \quad (1.2)$$

Here  $\mathbf{x}$  is called the *latent variable*, and the intuition is that this variable captures the essence of our observations. This is a reasonably general approach to building statistical models, and depending on the assumptions we are willing to make the model may have different degrees of feasibility.

### 1.2. Probabilistic PCA

Let us assume that we observe  $D$ -dimensional data  $\mathbf{y} \in \mathbb{R}^D$  and that the latent variable is a lower-dimensional vector  $\mathbf{x} \in \mathbb{R}^d, d < D$ . We can now create the *probabilistic principal component analysis* [46] model if we assume that there is an *affine* relationship between  $\mathbf{x}$  and  $\mathbf{y}$ , i.e.

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{Ax} + \mathbf{b}, \sigma^2\mathbf{I}). \quad (1.3)$$

Here  $\mathcal{N}$  denotes the density of a multivariate normal distribution,  $\mathbf{A} \in \mathbb{R}^{D \times d}, \mathbf{b} \in \mathbb{R}^D$  reflect the affine relationship, and  $\mathbf{I} \in \mathbb{R}^{D \times D}$  is an identity matrix. Informally

$$\mathbf{y} = f(\mathbf{x}) = \mathbf{Ax} + \mathbf{b} + \text{noise}. \quad (1.4)$$

If we further assume that  $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I})$ , then we have a fully specified model. As normal distributions are reasonably easy to manipulate one can carry out the integration of Eq. 1.2 in closed form and get that

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{b}, \mathbf{A}\mathbf{A}^\top + \sigma^2\mathbf{I}). \quad (1.5)$$

This gives a likelihood that can be optimized to estimate the model parameters.

This model is commonly used to visualize and otherwise understand the data, by inspection of the latent variable  $\mathbf{x}$ . The intuition is that  $\mathbf{x}$  is a low-dimensional projection of the observation  $\mathbf{y}$ .

Here, some care should be taken as the model is not *identifiable*, i.e. the distribution  $p(\mathbf{y})$  may be parametrized differently without changing the density. For example, if  $\mathbf{x}$ ,  $\mathbf{A}$ ,  $\mathbf{b}$ , and  $\sigma^2$  are the maximum likelihood estimators of the model parameters, then we can create a new set of parameters that gives an identical likelihood. Let  $\mathbf{R} \in \mathbb{R}^{d \times d}$  be a rotation matrix.<sup>1</sup> Now replace the (unobserved) latent variables with

$$\hat{\mathbf{x}} = \mathbf{R}\mathbf{x} \quad (1.6)$$

then  $p(\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I})$ . That is, if we rotate the latent variables, then they still follow a unit Gaussian. Then define  $\hat{\mathbf{A}} = \mathbf{A}\mathbf{R}^\top$  and let

$$p(\mathbf{y}|\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{y}|\hat{\mathbf{A}}\hat{\mathbf{x}} + \mathbf{b}, \sigma^2\mathbf{I}). \quad (1.7)$$

We have now constructed a new model by rotating the latent variable and changing the affine mapping correspondingly. The resulting data density is

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{b}, \mathbf{A}\mathbf{R}^\top\mathbf{R}\mathbf{A}^\top + \sigma^2\mathbf{I}), \quad (1.8)$$

which is identical the original model (1.5) as  $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$ . So the latent variable can be arbitrarily rotated without seeing a change in the resulting data density. Practically speaking, this implies that we can only estimate (identify) the latent variable  $\mathbf{x}$  up to an unknown rotation. If we plot the latent variable, we should, thus, expect it to be arbitrarily rotated. This is illustrated in Fig. 1.2.

---

<sup>1</sup>a rotation matrix is defined as satisfying  $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$

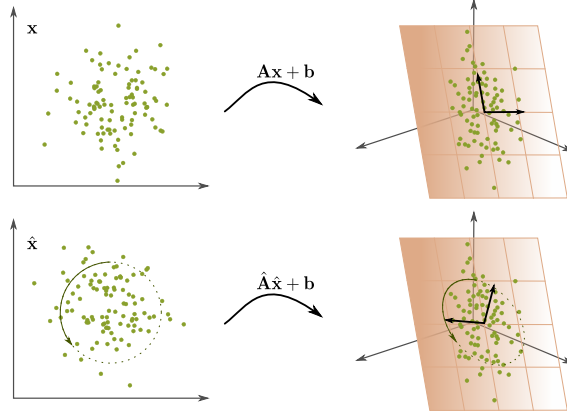


FIG. 1.2. *In probabilistic principal component analysis, we can get two equally good models by rotating the latent variable of one model and adapting the affine mapping correspondingly.*

### 1.3. Deep generative models

Instead of an affine model, as in probabilistic PCA, we can also consider general nonlinear functions, such as neural networks. This can, e.g., give rise to probabilistic models of the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}) \quad (1.9)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|f(\mathbf{x}), \sigma^2 \mathbf{I}). \quad (1.10)$$

That is, the latent variable is mapped through a function  $f: \mathbb{R}^d \rightarrow \mathbb{R}^D$ , after which independent Gaussian noise is added. Assuming  $f$  is nonlinear, then this results in a more flexible model than probabilistic PCA. This, however, also implies that the identifiability problem just got worse.

Consider a nonlinear invertible function  $h: \mathbb{R}^d \rightarrow \mathbb{R}^d$  with the property

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \Rightarrow \quad h(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (1.11)$$

One example of such a function is

$$h(\mathbf{x}) = \mathbf{R}_{\theta(\mathbf{x})} \mathbf{x}, \quad (1.12)$$



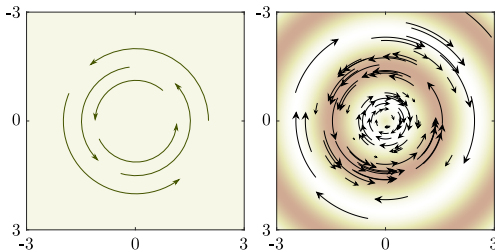


FIG. 1.3. Two example transformations  $h : \mathbb{R}^d \rightarrow \mathbb{R}^d$  that preserve distances to the origin. The left rotates by a constant angle, while the right rotates by a position-dependent angle. The figure has two panels. The first panel shows a rotation of the plane. The second panel shows a deformation of the plane obtained by rotating points with an angle that depends on the distance to the origin. The latter is akin to rotating concentric disks.

where  $\mathbf{R}_\theta$  is a linear transformation that rotates points by  $\theta(\mathbf{x}) = \sin(\pi\|\mathbf{x}\|)$ . This is illustrated in the right panel of Fig. 1.3.

Note that, in practice, we never work explicitly with  $h$ . This unknown function merely expresses the inherent flexibility of the model that we must take into account when trying to make sense of the latent representations (Fig. 1.4).

Using the function  $h$ , we can significantly distort the latent space without changing the distribution of the latent variables. Specifically, we can repeat the exercise from the previous section and introduce a new model from  $\hat{\mathbf{x}} = h(\mathbf{x})$ ,

$$p(\hat{\mathbf{x}}) = \mathcal{N}(\hat{\mathbf{x}}|\mathbf{0}, \mathbf{I}) \quad (1.13)$$

$$p(\mathbf{y}|\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{y}|f(h^{-1}(\hat{\mathbf{x}})), \sigma^2\mathbf{I}). \quad (1.14)$$

Thus, if we are working with a function class for  $f(\cdot)$  that is sufficiently general to also contain  $f(h^{-1}(\cdot))$ , then we can construct the same probabilistic model using very different parametrizations. As before, this may influence our ability to understand the generative process through inspection of the latent variables.

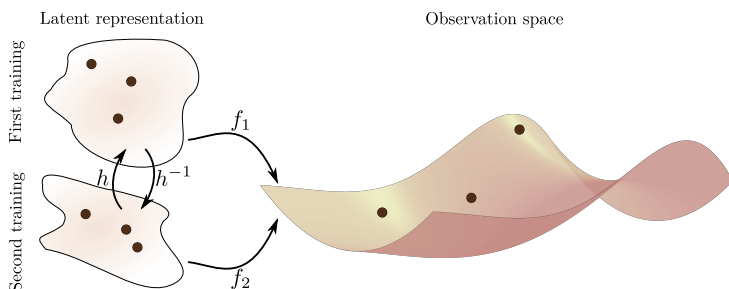


FIG. 1.4. *Fitting a model twice to the same data may yield different latent representations. Mathematically, we can connect the two representation spaces through a (unknown) function  $g : \mathbb{R}^d \rightarrow \mathbb{R}^d$ . Our goal is a set of tools for analyzing the latent representations that give identical conclusions for all equally good models.*

## 1.4. Outline & reading guide

The structure of this text is trivial: you are supposed to start at the beginning and stop at the end. A few sections and chapters here and there may be skipped; these are marked with a ‘yawn’ in the margin (see example<sup>2</sup> on the right).



The text emphasizes the parts of differential geometry that are of most use in the statistical analysis of data. As such, many central topics of differential geometry, such as *curvature* or *topology*, at best only play a secondary role. The predominant focus is on the tools needed for making quantitative statements, such as *distances*, *angles*, and *volumes*, as this is what the practitioner will tend to need.

Related work is only cited sparsely throughout the text. The concluding chapter provides some, albeit limited, remedies to this.

<sup>2</sup>These wonderful ‘yawnings’ have been hand-drawn by my daughter, Agnes Feragen-Hauberg. I hope they delight you as much as they do me.

## Exercises

### Exercise 1.1: PPCA distances and angles.

Probabilistic PCA can be equivalently expressed as

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{Ax} + \mathbf{b}, \sigma^2\mathbf{I}) \quad (1.15)$$

(with latent variables  $\mathbf{x}$ ) or

$$p(\mathbf{y}|\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{y}|\hat{\mathbf{A}}\hat{\mathbf{x}} + \mathbf{b}, \sigma^2\mathbf{I}) \quad (1.16)$$

(with latent variables  $\hat{\mathbf{x}} = \mathbf{R}\mathbf{x}$  and  $\hat{\mathbf{A}} = \mathbf{A}\mathbf{R}^\top$ , where  $\mathbf{R}$  is a rotation matrix). The two latent representations  $\mathbf{x}$  and  $\hat{\mathbf{x}}$  are clearly different (for  $\mathbf{R} \neq \mathbf{I}$ ). Show that

1. *Euclidean distances between points in these representations are identical, i.e.  $\|\mathbf{x}_i - \mathbf{x}_j\| = \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|$ , and*
2. *angles between points in these representations are identical, i.e.  $\angle(\mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_k - \mathbf{x}_j) = \angle(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j, \hat{\mathbf{x}}_k - \hat{\mathbf{x}}_j)$ .*

### Exercise 1.2: Autoencoder distances and angles.

An *autoencoder* is a nonlinear generalization of PCA. Let  $g_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^d$  be a neural network with weight parameters  $\theta$ , and let  $f_\psi : \mathbb{R}^d \rightarrow \mathbb{R}^D$  be another network with parameters  $\psi$ . Furthermore, let  $D$  denote the dimensionality of training data  $\mathbf{y}$  and let  $d < D$ . The autoencoder learns a mapping of data  $\mathbf{y} \in \mathbb{R}^D$  into a low-dimensional latent representation  $\mathbf{x} \in \mathbb{R}^d$  alongside a mapping that reconstructs the data from the representation. Specifically, the autoencoder is trained to minimize the reconstruction error

$$\theta^*, \psi^* = \operatorname{argmin}_{\theta, \psi} \sum_{n=1}^N \|f_\psi(g_\theta(\mathbf{y}_n)) - \mathbf{y}_n\|^2. \quad (1.27)$$

1. *Implement an autoencoder for a simple dataset. Pick the latent dimension  $d = 2$  and plot the resulting low-dimensional representations.*
2. *Plot the latent representations for different models trained from different random initialization. Do you recover the same representations?*

3. *Compute pairwise distances  $\|g(\mathbf{y}_i) - g(\mathbf{y}_j)\|$  between latent representations in different models. Are the distances between representations identical across models? Discuss what the answer implies.*

*Exercise 1.3: Reparametrizations.*

Consider the function  $h(\mathbf{x}) = \mathbf{R}_{\theta(\mathbf{x})}\mathbf{x}$  (see Eq. 1.12 and Fig. 1.3) in two dimensions (i.e.  $\mathbf{x} \in \mathbb{R}^2$ ). This is an example of a reparametrization of a representation space.

1. *Numerically simulate 100 standard normally distributed points  $\mathbf{x}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and plot how  $h(\mathbf{x}_n)$  changes compared to  $\mathbf{x}_n$ .*
2. *Visualize how the reparametrization changes pairwise distances. Specifically, compute  $d_{nm} = \|\mathbf{x}_n - \mathbf{x}_m\|$  and  $\hat{d}_{nm} = \|h(\mathbf{x}_n) - h(\mathbf{x}_m)\|$  and make a scatter plot containing points  $(d_{nm}, \hat{d}_{nm})$ . What conclusion do you draw from the plot?*

## Chapter 2

### *Making maps*

*In which we pause to look at the world.*

Planet Earth is approximately spherical, making it a prototypical manifold. Since carrying around spherical maps is impractical, much effort has gone into making *flat* maps, such as the one in Fig. 2.2 below. One reason so much effort has been required is that the task is fundamentally impossible. The impossibility was made precise by Carl Friedrich Gauss in 1827 through his *Theorema Egregium*.<sup>1</sup>

This theorem states that *the Gaussian curvature of a surface is preserved by local isometries*. Admittedly, we haven't defined most terms in this statement, but we can try to understand it nonetheless. An *isometry* is a transformation that preserves all distances, and the Gaussian curvature is a measure of how much a surface bends. So, the theorem informs us that *if* we find a transformation that takes the spherical Earth and places it on a flat piece of paper such that distances between all points are identical in the two representations, then the curvature of the sphere and the paper is the same as well. Since the curvature of a sphere and a flat piece of paper surely must be different, the theorem informs us that the transformation cannot preserve distances. The map must be distorted.

The astute reader may reasonably object here. Perhaps the issue at hand here is that we cannot make a map of the *entire* sphere as that requires us to 'tear it open', which must cause trouble. *Perhaps we can*



FIG. 2.1. Planet Earth.  
Image courtesy of NASA.

---

<sup>1</sup>Translates to *remarkable theorem*, which indicates the level of excitement from Gauss upon his discovery.



FIG. 2.2. World map by Diego Ribero, 1527. While beautiful, the map is fundamentally misleading as any flat depiction of a curved object (here: Earth) will not reflect distances accurately. A photograph of a 15th century map of the world.

make an accurate map of a part of the sphere? Again, Gauss' theorem informs us that this cannot be done. If we take a part of the sphere, then this remains a *curved* surface, so any map onto a *flat* plane must distort distances.

The takeaway message is that whenever we deal with curved objects, such as manifolds estimated from data, we cannot make this flat without getting a distorted picture. *This is the fundamental challenge that geometry solves.*

In practice, distortions will be less severe if we only consider small segments of the sphere at a time. Historically, books of smaller maps have been created to provide less distorted maps of the world (see, e.g., Fig. 2.3). These books are known as *atlases*. Individual pages contain maps of small regions, but jointly the atlas can cover a wider region. To help the reader, individual pages overlap in the coverage of the world. This structure can also be mimicked mathematically, which we will touch upon in chapter 12.

## Exercises

### Exercise 2.1: Maps of the world.

Humans have long aimed to map their surroundings to navigate reliably. The resulting maps have often been used to form mental pictures of

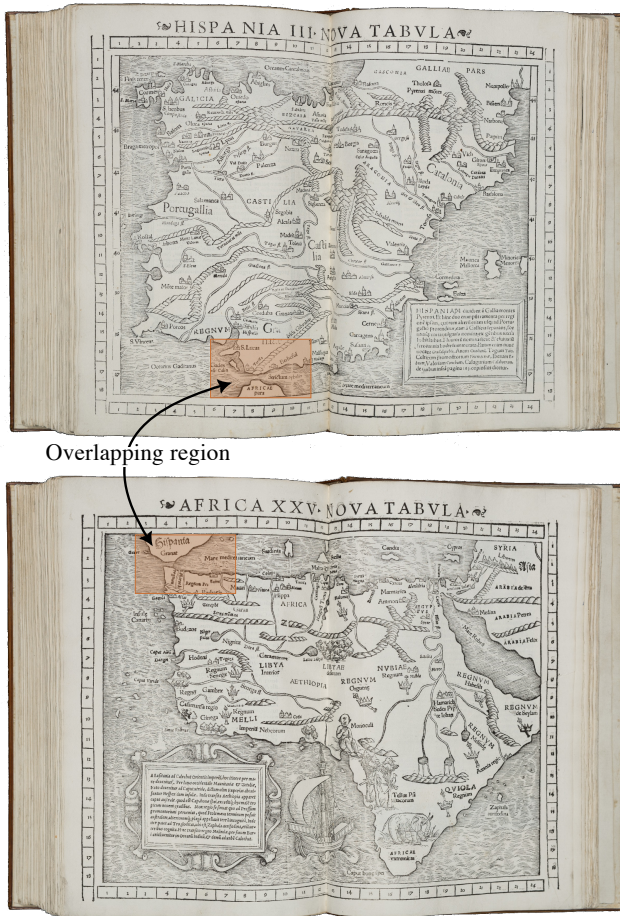


FIG. 2.3. Sebastian Münster's *Geographia Universalis* (1552), here showing Spain and Africa. The individual maps partially overlap, allowing the reader to navigate a larger region than the individual page covers.

the world in the general population. Rulers of the world (kings, queens, presidents, etc.) have historically been keen on shaping these maps to influence how people perceive the world.

1. *Find a map of planet Earth, which is made according to Mercator's projection. Next, find a map made according to Peter's projection. What are the differences? Is one more correct than the other? Why would a ruler prefer one map over the other?*
2. *We make the analogy between maps of planet Earth and learned representations to gain an intuition about the parametrization issue. What do the answers to question 1 tell you about representation learning?*



## Chapter 3

### *Embedded and immersed manifolds*

*In which we begin to observe manifolds.*

In most data-centric uses of manifolds, we observe data  $\mathbf{y}_1, \dots, \mathbf{y}_N$  as a collection of vectors in  $\mathbb{R}^D$ . We call this the *ambient space*. We then fit a lower-dimensional manifold  $\mathcal{M}$  to this data, drawing on the assumption that the intrinsic structure of the data inherently has a lower dimension than the space in which we made our observations.

From a practical perspective, we construct this manifold through a mapping

$$f : \mathbb{R}^d \rightarrow \mathbb{R}^D, \quad d \leq D \quad (3.1)$$

from a low-dimensional space into a high-dimensional observation space. Depending on how we choose  $f$ , we may then create different manifolds, e.g., if  $f$  is a linear function, the manifold will be a hyperplane in  $\mathbb{R}^D$ ; see Fig. 3.1 for an example. Our interest will predominantly be in cases where  $f$  is nonlinear, though, as this will allow our models to capture more variability in the data.

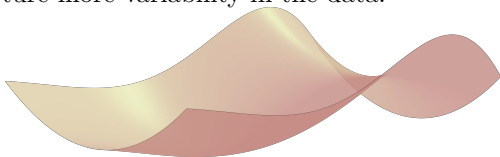


FIG. 3.1. A manifold spanned by a function  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ .

### *3.1. Manifold learning*

To fit a manifold to data, a common strategy is to pick a suitable *loss function* that reflects the properties we seek in our model. For instance, we may seek a least-squares fit to our data,

$$\theta^*, \mathbf{x}_1^*, \dots, \mathbf{x}_n^* = \operatorname{argmin} \mathcal{L}(\theta, \mathbf{x}_1, \dots, \mathbf{x}_N) \quad (3.2)$$

$$= \operatorname{argmin} \sum_{n=1}^N \|f_{\theta}(\mathbf{x}_n) - \mathbf{y}_n\|^2. \quad (3.3)$$

Here, two important points must be raised.

**First**, we have parametrized the mapping  $f$  by a vector  $\theta$ . For instance, if we have chosen  $f$  to be a polynomial, then  $\theta$  will contain the corresponding coefficients, or if  $f$  is a neural network,  $\theta$  will be the weights. To ease notation, we will generally not write the parameters explicitly.

**Second**, as we do not know  $f$  beforehand, we also do not know the input points  $\mathbf{x}_n$  that correspond to the observed data  $\mathbf{y}_n$ . In Eq. 3.3, we, therefore, treat these input points  $\mathbf{x}_n$  as unknown (latent) and seek those points that minimize our loss function. Had we known  $\mathbf{x}_n$ , our problem would have been reduced to finding  $f$  by regression.

Many alternatives to the least-squares loss function (3.3) exist, but we commonly rely on some similarity measure between  $f(\mathbf{x}_n)$  and  $\mathbf{y}_n$  to determine the quality of the fit. This will be the root cause of many of the troubles examined in this book. Specifically, consider an invertible function  $h : \mathbb{R}^d \rightarrow \mathbb{R}^d$ , and assume that  $\{f, \mathbf{x}_1, \dots, \mathbf{x}_N\}$  span an optimal manifold (according to our loss function), then (Fig. 3.2)

$$\hat{f}(\hat{\mathbf{x}}) = f(h^{-1}(\hat{\mathbf{x}})) \quad (3.4)$$

$$\hat{\mathbf{x}}_n = h(\mathbf{x}_n) \quad (3.5)$$

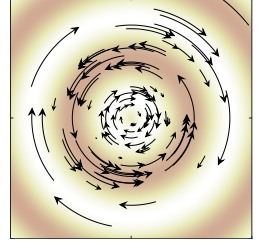


FIG. 3.2. An example of a reparametrization given by a function  $h : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ . This figure is a repetition of Fig. 1.3.

will be equally optimal. In fact, these two manifolds are identical as  $\hat{f}(\hat{\mathbf{x}}) = f(\mathbf{x})$ . Note how  $h$  and  $h^{-1}$  cancel each other out. This implies that a learned manifold will generally not have a unique set of optimal parameters and that we can always parametrize the manifold in a different, but equally good, way. We call this a *reparametrization issue*, but the statistically inclined reader may prefer to call it an *identifiability problem*. The practical implication of this issue is that our model will be difficult to interpret.

We often think of the inputs  $\mathbf{x}_n$  as low-dimensional representations of our data. We may therefore hope to learn something about our data and the process that generated it, by inspecting  $\mathbf{x}_n$ . For instance,

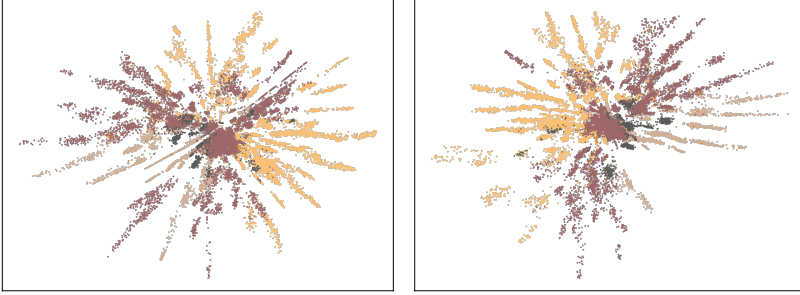


FIG. 3.3. *The low-dimensional representations  $\mathbf{x}_n$  of two variational autoencoders fitted to the same data [15]. Here, each point corresponds to one protein from the beta-lactamase family. The two different representations may result in different biological interpretations.*

we may choose to plot  $\mathbf{x}_1, \dots, \mathbf{x}_N$  if their dimension is sufficiently low. The reparametrization issue tells us that this is a dangerous strategy.

### 3.2. Reparametrizations in practice

In practice, the reparametrization issue appears in the form that if we fit a model to data several times, each instance may result in different low-dimensional representations. Figure 3.3 shows an example of this. Here, a *variational autoencoder (VAE)* [26, 37] is fitted twice to high-dimensional data using different random initializations. The data in question is a collection of protein sequences from the beta-lactamase family, and the colors in the plots indicate subfamilies [39, 15]. We observe that the two models have similarities in their low-dimensional representations, but also notable differences. If the objective is to extract knowledge from the plots about the underlying biology that drives the proteins, then clearly this is a brittle process. Depending on which plot we choose to investigate, we may end up drawing different conclusions. If the purpose of the analysis is to phrase biological hypotheses that are to be tested in the lab, we quickly run the risk of wasting the time (and money) of the experimental scientists.

### 3.3. Relation to map-making

In Chapter 2 we briefly touched upon Gauss' *Theorema Egregium*, which states that curved objects cannot be faithfully represented by a flat surface. In the context of manifold learning, this result indicates the difficulty of the learning task: the idea of learning an Euclidean representation of a curved data manifold must imply that a distorted representation is learned.

### 3.4. Statistical identifiability

Consider a statistical model  $p_\theta$  that is parametrized by  $\theta$ . We then say that the model is *identifiable* if the mapping  $\theta \mapsto p_\theta$  is one-to-one (injective). This may sound terribly abstract, but it really isn't. Being identifiable, merely say that any given model  $p_\theta$  can *only* be parametrized in *one* way.

For example, the normal distribution  $p_\theta = \mathcal{N}(\mu, \sigma^2)$  is a statistical model with parameters  $\theta = \{\mu, \sigma\}$ . If we change these parameters, we also change the statistical model. Therefore  $p_\theta$  is identifiable.

In contrast, a mixture of normal distributions,

$$p_\theta(\mathbf{y}) = \frac{1}{2}\mathcal{N}(\mathbf{y}|\mu_1, \sigma_1^2) + \frac{1}{2}\mathcal{N}(\mathbf{y}|\mu_2, \sigma_2^2), \quad (3.6)$$

is *not* identifiable (Fig. 3.4). To see this, note that the model has parameters  $\theta = \{\mu_1, \sigma_1, \mu_2, \sigma_2\}$ . By a simple 'swapping', we can create a new parameter vector  $\theta' = \{\mu_2, \sigma_2, \mu_1, \sigma_1\}$  that gives the same

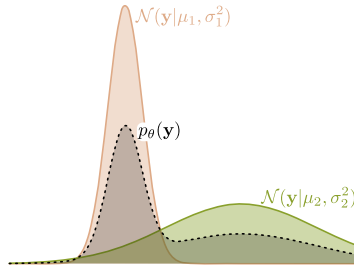


FIG. 3.4. A mixture model is not identifiable because the mixture components can be swapped.

statistical model as before. Formally,  $p_\theta = p_{\theta'}$  even if  $\theta \neq \theta'$ , such that the mapping  $\theta \mapsto p_\theta$  is not one-to-one.

Chapter 1 noted that probabilistic PCA gives a statistical model

$$p_\theta(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{b}, \mathbf{A}\mathbf{A}^\top + \sigma^2\mathbf{I}), \quad (3.7)$$

which can be equivalently written as

$$p_{\theta'}(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{b}, \mathbf{A}\mathbf{R}^\top\mathbf{R}\mathbf{A}^\top + \sigma^2\mathbf{I}), \quad (3.8)$$

for some rotation matrix  $\mathbf{R}$ . These two (identical) models can be written as  $\theta = \{\mathbf{b}, \mathbf{A}, \sigma\}$  and  $\theta' = \{\mathbf{b}, \mathbf{A}\mathbf{R}^\top, \sigma\}$ . Again, we see that  $p_\theta = p_{\theta'}$  even if  $\theta \neq \theta'$ , such that the probabilistic PCA model is *not* identifiable. Fig. 3.5 illustrates the issue.

In many cases, only an unknown latent *reparametrization* (e.g., a rotation in the PCA example) prevents the model from being identifiable. This book focuses on these cases and we will interchange between the terms reparametrizations and *statistical identifiability* even if these concepts are not entirely the same.

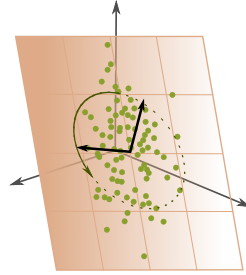


FIG. 3.5. *Probabilistic PCA is not identifiable as we cannot determine a unique basis for the estimated subspace.*

### 3.5. Manifolds

Most loss functions of interest do not have closed-form solutions, and we will generally have to rely on iterative optimization methods to fit a manifold to data. This will be significantly easier if we can differentiate our loss function, which, in turn, requires us to be able to differentiate the mapping  $f$ . We will therefore assume that  $f$  is a smooth function from now on. This assumption restricts what we can model and will give rise to our first definition of the term *manifold*.

Consider a smooth function

$$f : \Omega \rightarrow \mathbb{R}^D, \quad (3.9)$$

where  $\Omega \subset \mathbb{R}^d$  is the input domain, which may be the entire  $\mathbb{R}^d$  or some connected subset thereof. We say that a *manifold*  $\mathcal{M}$  is the image

of  $\Omega$  under  $f$ , i.e.,  $\mathcal{M} = f(\Omega)$ . That's it for now. We will revisit this definition later when it becomes too restrictive, but for most machine learning tasks, we may think of a manifold as the collection of all possible outputs a given smooth function may generate.

Since  $f$  is smooth, we may locally approximate the manifold by its *tangent space*. According to Taylor's theorem, the Jacobian matrix of  $f$  provides a basis for this tangent space. We often think of manifolds as point collections that are locally Euclidean, and this view will provide us with many of the tools needed to study manifolds.

### 3.6. Embedded manifolds

We say that a manifold is *embedded* if  $f$  is invertible on  $\mathcal{M}$ . Practically speaking, this implies that the manifold does not self-intersect or locally change dimensionality. Consider the curve in Fig. 3.6, which is described by a function  $f : [a, b] \rightarrow \mathbb{R}^2$ . In the point of self-intersection we have  $f(\mathbf{x}_0) = f(\mathbf{x}_1)$  for  $\mathbf{x}_0 \neq \mathbf{x}_1$ , such that  $f$  is *not* invertible at this point. Consequently, this manifold is *not* embedded.

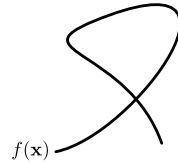


FIG. 3.6. The curve  $f(\mathbf{x}) \in \mathbb{R}^2$  self-intersect and is not an embedded manifold.

Mathematically, we can ensure that a manifold is embedded if the function  $f$  is *injective*, i.e., if  $f(\mathbf{x}_1) = f(\mathbf{x}_2) \Rightarrow \mathbf{x}_1 = \mathbf{x}_2$ . This implies that  $f$  is invertible on its image, i.e. if  $\mathbf{y} = f(\mathbf{x})$  then the inverse mapping<sup>1</sup> exists and  $\mathbf{x} = f^{-1}(\mathbf{y})$ . Most function classes (e.g., neural networks) that are commonly used in machine learning are not guaranteed to be injective, so it is rare that we get to work with embedded manifolds in practice.

### 3.7. Immersed manifolds

Practically, it is too restrictive to only study embedded manifolds, and we, therefore, open up for allowing functions  $f$  that are only *locally invertible*. That is, if the Jacobian  $\mathbf{J} = \partial f / \partial \mathbf{x}$  has full rank for all

<sup>1</sup>Technically, we refer to this as the *left inverse* to emphasize that  $f$  is only invertible on its image.

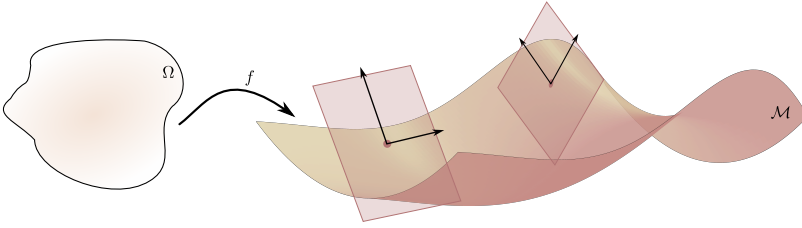


FIG. 3.7. The Jacobian of  $f$  can be interpreted as giving a basis for a tangential plane to the manifold. When the Jacobian has full rank, then this tangential plane has the same dimensionality as the manifold (i.e., the manifold does not collapse anywhere).

$\mathbf{x} \in \Omega$ , then we say that  $\mathcal{M} = f(\Omega)$  is an *immersed manifold*. Since the Jacobian can be seen as a local linear approximation to the manifold, we can think of the Jacobian as a basis of a plane that is tangential to the manifold (see Fig. 3.7). When we require the Jacobian to be full rank, we merely say that the dimensionality of this tangential plane should always match that of the manifold. Note that embedded manifolds are immersed, but the opposite need not hold true.

## Exercises

*Exercise 3.1: You really should....*

If you haven't already, solve exercise 1.2.

*Exercise 3.2: Extrapolation.*

In curve fitting, we often control how a fitted function extrapolates. For example, *kernel ridge regression* [43] will, for most kernels, extrapolate to 0 (see Fig. 3.8 for an example). Assume a generative model  $\mathbf{y} = f(\mathbf{x})$ ,

$$f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_D(\mathbf{x}) \end{pmatrix} \quad (3.10)$$

consisting of  $D$  functions  $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ .

1. Assume that functions  $f_i$  extrapolate to 0 for all  $i = 1, \dots, D$ , i.e., assume that  $f_i(\mathbf{x}) = 0$  for all  $\mathbf{x}$  that are more than a distance

$\tau$  away from the nearest training data. Show that the resulting manifold cannot be embedded.

*Exercise 3.3: Immersion & embedding.*

Consider a generative model  $\mathbf{y} = f(\mathbf{x})$ , where  $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$  and  $\mathbf{y} = (y_1, y_2, y_3) \in \mathbb{R}^3$ . Let

$$f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} 2x_1^2 + x_2^2 \\ x_1 \\ x_2 \end{pmatrix}. \quad (3.11)$$

1. Derive the Jacobian matrix of  $f$ .
2. Show that the generative model spans an immersed manifold.
3. Show that the generative model spans an embedded manifold.

*Exercise 3.4: Immersion only.*

Consider a generative model  $\mathbf{y} = f(\mathbf{x})$ , where  $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$  and  $\mathbf{y} = (y_1, y_2, y_3) \in \mathbb{R}^3$ . Let

$$f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \cos(x_1) \\ \sin(x_1) \\ \cos(x_2) \end{pmatrix}. \quad (3.15)$$

1. Derive the Jacobian matrix of  $f$ .
2. Show that the generative model spans an immersed manifold.
3. Show that the generative model does not span an embedded manifold.

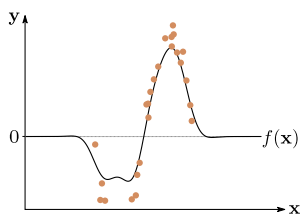


FIG. 3.8. The regression curve  $f(\mathbf{x})$  of a kernel ridge regressor extrapolates to zero.



## Chapter 4

### *Distances on manifolds*

*In which we start being able to tell what  
is close and what is far away.*

To build a statistical model, we need some way of measuring similarity. If we want to build statistical models of data distributed on a manifold, we then need a way of measuring on this manifold. This is where *Riemannian geometry* comes into play.

#### *4.1. Euclidean curve lengths*

The basic idea behind distances on manifolds is to define them as lengths of shortest paths. Hence, we need to agree on how to measure the length of a given curve, and we here start with the Euclidean case. The general case is then a trivial extension.

Let  $c : [0, 1] \rightarrow \mathbb{R}^D$  be a smooth curve in a  $D$ -dimensional space. We can then approximate this curve by a set of straight lines connecting points

$$\{c(0), c(t_1)\}, \{c(t_1), c(t_2)\}, \dots, \{c(t_N), c(1)\}, \quad (4.1)$$

where  $t_n \in ]0, 1[, n = 1, \dots, N$ . This approximation is shown in Fig. 4.1. We can then approximate the length of  $c$  as the sum of the length of these line segments, i.e.

$$\text{Length}(c) \approx \sum_{i=0}^N \|c(t_i) - c(t_{i+1})\|, \quad (4.2)$$

where we have defined  $t_0 = 0$  and  $t_{N+1} = 1$ . Increasing the number of line segments in the approximation then gives an increasingly better

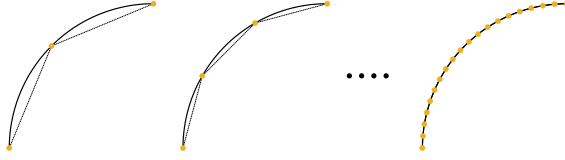


FIG. 4.1. The length of a curve (4.5) can be understood as “cutting up” the curve in small linear segments and adding up their lengths.

length approximation, and a natural definition of curve length is then to consider the limit  $N \rightarrow \infty$ ,

$$\text{Length}(c) = \lim_{N \rightarrow \infty} \sum_{i=0}^N \|c(t_i) - c(t_{i+1})\|. \quad (4.3)$$

A neat aspect of this definition is that it lends itself immediately to a numerical implementation by using Eq. 4.2.

Analytically, we can follow the limit in Eq. 4.3 and turn the sum into an integral,

$$\text{Length}(c) = \lim_{N \rightarrow \infty} \sum_{i=0}^N \left\| \frac{c(t_i) - c(t_{i+1})}{\Delta t} \right\| \Delta t \quad (4.4)$$

$$= \int_0^1 \left\| \frac{dc(t)}{dt} \right\| dt, \quad (4.5)$$

where  $\Delta t = |t_{i+1} - t_i|$ . This integral expression (4.5) is often easier to manipulate mathematically, as we do not have to worry about the limit.

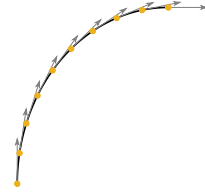


FIG. 4.2. The length of a curve can also be expressed by integrating the curve derivative, which avoids the use of a limit.

## 4.2. Reparametrization issues

A recurring problem that we will keep facing in various forms starts with the observation that a curve can be *reparametrized* without changing its length. What does that mean? Consider a curve  $c : [0, 1] \rightarrow \mathbb{R}^D$ , then its length is given by Eq. 4.5 above. Now we

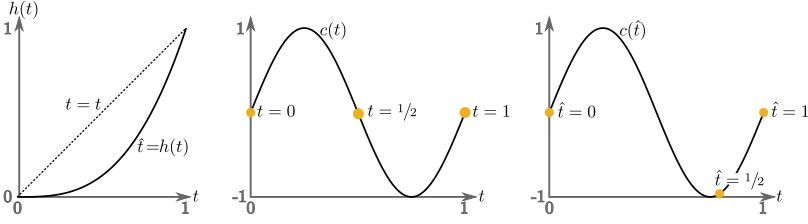


FIG. 4.3. A sine curve under two parametrizations. While the points generated along the curve are identical in the two parametrizations, we get different positions along the curve for a fixed input.

introduce a smooth monotonic function  $h : [0, 1] \rightarrow [0, 1]$  with the properties

$$h(0) = 0 \quad \text{and} \quad h(1) = 1. \quad (4.6)$$

The idea is that instead of parametrizing the curve  $c$  by  $t$ , we now parametrize it by  $t = h(s)$ . The first thing we observe (Fig. 4.3) is that  $\hat{c}(s) = c(h(s))$  generates the same curve as  $c(t)$ . The curve is, however, traversed at a different speed. If we think of the curve as a path along which to walk, then both  $c$  and  $\hat{c}$  define the same route, but we may reach different destinations at different times.

Since  $c$  and  $\hat{c}$  correspond to the same curve, we intuitively expect that they have equal lengths. To verify this, we compute the length of  $c$  as

$$\text{Length}(c) = \int_0^1 \left\| \frac{dc}{dt} \right\| dt \quad (\text{Def. of curve length}) \quad (4.7)$$

$$= \int_0^1 \left\| \left( \frac{dc}{dt} \right)_{t=h(s)} \right\| \frac{dh}{ds} ds \quad t \rightarrow h(s) \quad (4.8)$$

$$= \int_0^1 \left\| \left( \frac{dc}{dt} \right)_{t=h(s)} \frac{dh}{ds} \right\| ds \quad dh/ds > 0 \quad (4.9)$$

$$= \int_0^1 \left\| \frac{dc}{ds} \right\| ds \quad (\text{Chain rule}) \quad (4.10)$$

$$= \text{Length}(\hat{c}). \quad (4.11)$$

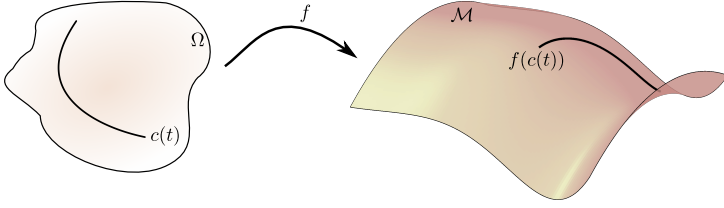


FIG. 4.4. The length of a curve on a manifold is simply the Euclidean length of the immersed curve.

As expected, the speed with which we walk along a path does not change the length of the path itself.

Why is this a problem? We will often find ourselves optimizing a curve to satisfy some criteria,<sup>1</sup> in which case it is rather annoying that multiple versions of the same curve exist, as this implies that an optimal curve is never unique. At best, an optimal curve will only be unique up to some reparametrization. Practically, this can make gradient-based optimization unstable, as every local optimum will be a plateau corresponding to different parametrizations of the same curve.

### 4.3. Measuring on immersed manifolds

Now we repeat the exercise for curves restricted to a manifold  $\mathcal{M} = f(\Omega)$ , where  $\Omega$  is a connected subset of  $\mathbb{R}^d$ , such that  $\mathcal{M}$  is an immersed manifold in  $\mathbb{R}^D$ .

Let  $c : [0, 1] \rightarrow \Omega$  be a smooth curve in  $\Omega$  such that  $f(c)$  is a smooth curve on  $\mathcal{M}$ . We then define the *curve length* of  $c$  to be the length of  $f(c)$ , i.e.,

$$\text{Length}_{\mathcal{M}}(c) \stackrel{\text{def}}{=} \text{Length}(f(c)) = \int_0^1 \left\| \frac{d}{dt} f(c_t) \right\| dt, \quad (4.12)$$

where we use the short-hand  $c_t = c(t)$ . This merely says, given a curve in  $\Omega$ , we measure its length by mapping the curve to  $\mathbb{R}^D$  and follow the usual Euclidean definition.

---

<sup>1</sup>e.g., we may seek a curve that is as short as possible

Given two points  $c_0, c_1 \in \Omega$ , we can define the shortest connecting path as<sup>2</sup>

$$\begin{aligned} c_{01} &= \underset{c}{\operatorname{argmin}} \operatorname{Length}_{\mathcal{M}}(c) \\ &\text{subject to } c(0) = c_0 \text{ and } c(1) = c_1. \end{aligned} \tag{4.13}$$

We call any local minimizer a *geodesic* as this is the usual convention. In practice, we tend to be interested in global minima, and will often implicitly assume that we can compute such a curve.

Given the shortest connecting curve between two points, it is natural to define the *geodesic distance* as the length of this curve,

$$\operatorname{dist}(c_0, c_1) = \operatorname{Length}_{\mathcal{M}}(c_{01}). \tag{4.14}$$

This definition is an example of the implicit assumption that  $c_{01}$  is a global minimizer.

We may ask if this geodesic distance is actually a proper distance function. That is, does it satisfy the intuitive criteria

$$\operatorname{dist}(x, y) = 0 \iff x = y \tag{4.15}$$

$$\operatorname{dist}(x, y) = \operatorname{dist}(y, x) \tag{4.16}$$

$$\operatorname{dist}(x, y) \leq \operatorname{dist}(x, z) + \operatorname{dist}(z, y) \tag{4.17}$$

for points  $x, y, z \in \Omega$ ? These three requirements are essentially all satisfied due to the definition of the geodesic as the minimal path. The first requirement is easy: if the geodesic has zero length, then its end-points must coincide and vice versa. The second requirement follows by noting that if the two distances are different, then one must be larger than the other. The larger geodesic cannot be the globally shortest path, as a shorter path exists. The third requirement<sup>3</sup> also follows by contradiction: assume that  $c_{xy}$  is the geodesic connecting  $x$  and  $y$  and that a shorter connecting path exists that goes through  $z$ . Then  $c_{xy}$  cannot be geodesic, and we have reached a contradiction.

---

<sup>2</sup>Technically, one should use an infimum rather than a minimum, but practically speaking we always seek a minimizer. Hence we here stick with the more intuitive definition.

<sup>3</sup>Also known as the triangle inequality.

#### 4.4. Reparametrization issues

We have previously seen that a given curve does not have a unique parametrization. Since the length of a curve does not depend on its parametrization, neither does the distance definition. The distance between two points is *invariant to reparametrizations* of the geodesic.

However, as touched upon in chapter 3, the manifold itself can also be reparametrized. As usual, let

$$\Omega \subset \mathbb{R}^d \tag{4.18}$$

$$f : \Omega \rightarrow \mathbb{R}^D, \quad d < D \tag{4.19}$$

such that our manifold is  $\mathcal{M} = f(\Omega)$ . Now consider a smooth invertible function  $h : \mathbb{R}^d \rightarrow \mathbb{R}^d$ , then we may define

$$\hat{\Omega} = h(\Omega) \subset \mathbb{R}^d \tag{4.20}$$

$$\hat{f}(\mathbf{x}) = f(h^{-1}(\mathbf{x})). \tag{4.21}$$

Then we have

$$\hat{f}(\hat{\Omega}) = f(h^{-1}(g(\Omega))) \tag{4.22}$$

$$= f(\Omega) \tag{4.23}$$

$$= \mathcal{M}. \tag{4.24}$$

In words,  $\hat{\Omega}$  and  $\hat{f}$  define the exact same manifold as  $\Omega$  and  $f$ , such that a given manifold can be parametrized differently.

When analyzing data, we often call upon the *manifold hypothesis*, which states that observed data is distributed near a low-dimensional manifold within the observation space. As an example, we may model our data  $\mathbf{y}$  as

$$\mathbf{y} = f(\mathbf{x}) + \epsilon, \tag{4.25}$$

where  $\epsilon$  is a random variable capturing the ‘off-manifold’ noise, and  $\mathbf{x}$  is a low-dimensional representation of  $\mathbf{y}$ . The reparametrization issue then implies that if some learning algorithm has recovered  $f$  alongside  $\mathbf{x}_{1:N} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  then these are not unique solutions. We may reparametrize by some  $h$  to recover an equally good fit to our data.

Using  $\hat{\mathbf{x}} = h(\mathbf{x})$  and  $\hat{f}(\hat{x}) = f(h^{-1}(\hat{x}))$  will give the exact same density estimate of our data. In the statistics literature, this is known as an *identifiability issue* as we cannot recover unique  $f$  and  $\mathbf{x}_{1:N}$  from the density of data alone [28]. This implies that if we analyze the estimated  $f$  and  $\mathbf{x}_{1:N}$ , then we need to take into account that these are by no means uniquely defined.

Here geometry comes in handy. While  $\mathbf{x}_{1:N}$  may not be identifiable, we can make pairwise distances between  $\mathbf{x}$ 's identifiable. Consider two points on the learned manifold  $\mathbf{x}_n, \mathbf{x}_m$ , approximately corresponding to observations  $\mathbf{y}_n \approx f(\mathbf{x}_n)$  and  $\mathbf{y}_m \approx f(\mathbf{x}_m)$ . If we compute the Euclidean distance between  $\mathbf{x}_n$  and  $\mathbf{x}_m$ , then this may<sup>4</sup> change under different parametrizations, i.e., in general

$$\|\mathbf{x}_n - \mathbf{x}_m\| \neq \|h(\mathbf{x}_n) - h(\mathbf{x}_m)\|. \quad (4.26)$$

In contrast, the geodesic distance does not depend on parametrization. To see this, let  $c$  be a smooth curve such that

$$c(0) = \mathbf{x}_n \quad \text{and} \quad c(1) = \mathbf{x}_m, \quad (4.27)$$

then the corresponding curve along the manifold is  $t \mapsto f(c(t))$ . If we reparametrize manifold by  $h$ , then  $c$  become  $t \mapsto h(c(t))$ , which will be immersed as  $t \mapsto f(h^{-1}(h(c(t)))) = f(c(t))$ . This implies that the length of a curve (4.5) does not depend on the parametrization of the manifold. Consequently, geodesic distances between points  $\mathbf{x}_n$  and  $\mathbf{x}_m$  are unique. In statistics jargon, we may say that the geodesic distance between  $\mathbf{x}_n$  and  $\mathbf{x}_m$  is identifiable, even if the points themselves are not.

## Exercises

### Exercise 4.1: Euclidean curve lengths.

Consider the curve  $c : [0, 1] \rightarrow \mathbb{R}^2$  defined as (see Fig. 4.5)

$$c_t = c(t) = \begin{pmatrix} 2t + 1 \\ -t^2 \end{pmatrix}. \quad (4.28)$$

---

<sup>4</sup>The exception is when  $h$  performs a rotation, which does preserve distances.

1. Derive an expression of the speed function  $t \mapsto \|\dot{c}_t\|$ .
2. Compute the Euclidean length of the curve. Hint:

$$\int \sqrt{1+t^2} dt = \frac{1}{2} \left( \sqrt{1+t^2} t + \sinh^{-1}(t) \right) + C, \quad (4.29)$$

for some constant  $C$ .

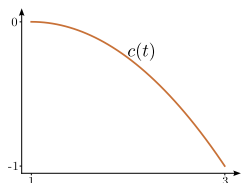


FIG. 4.5. The curve given by Eq. 4.28.

*Exercise 4.2: Numerical computation of Euclidean curve lengths.*

Consider the curve  $c : [0, 1] \rightarrow \mathbb{R}^2$  defined in Eq. 4.28.

1. Write a computer program that evaluates the length of the curve  $c$  using Eq. 4.2.
2. If you have completed exercise 4.1:
  - (a) Did the numerical and the analytical results agree?
  - (b) Use the analytic expression for  $\|\dot{c}_t\|$  to write a computer program that evaluates the length of the curve using Eq. 4.5. Note that you need to approximate the integral with a sum.

*Exercise 4.3: Metricity of geodesic distance.*

On page 33, a proof sketch is provided for why the geodesic distance (4.14) is a valid distance measure. This is somewhat ‘handwavy’.

1. Prove that Eq. 4.15 holds for the geodesic distance.
2. Prove that Eq. 4.16 holds for the geodesic distance.
3. Prove that Eq. 4.17 holds for the geodesic distance.

*Exercise 4.4: Manifold curve lengths.*

Consider the manifold spanned by the function  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$  defined as

$$f(x_1, x_2) = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{pmatrix}. \quad (4.38)$$



Let  $c_t = c(t) = \mathbf{v}t$  for a non-zero vector  $\mathbf{v} \in \mathbb{R}^2$ .

1. Derive an expression of the speed function  $t \mapsto \|\mathrm{d}/\mathrm{d}t f(c_t)\|$ .
2. Show that the length of  $c$  from  $t = 0$  to  $t = 1$  is

$$\text{Length}_{\mathcal{M}}(c) = \|\mathbf{v}\| \int_0^1 \sqrt{1 + 4\|\mathbf{v}\|^2 t^2} \mathrm{d}t. \quad (4.39)$$

*Exercise 4.5: Reparametrizations.*

Consider the manifold spanned by the function  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$  defined by Eq. 4.38 (previous exercise), and let  $\mathcal{M}_1 = f(\mathbb{R}^2)$  be the manifold consisting of all points that can be generated by  $f$ . Now consider the function  $h : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  defined as

$$h(x_1, x_2) = \begin{pmatrix} x_1^3 \\ x_2^3 \end{pmatrix}. \quad (4.46)$$

Next, we reparametrize our manifold as  $\mathcal{M}_2 = f(h(\mathbb{R}^2))$ .

1. Derive an alternative function  $\hat{f}$  for spanning the manifold such that  $f(\mathbf{x}) = \hat{f}(h(\mathbf{x}))$  for all  $\mathbf{x} \in \mathbb{R}^2$ .
2. Define  $\mathcal{M}_3 = \hat{f}(\mathbb{R}^2)$  and show that  $\mathcal{M}_1 = \mathcal{M}_2 = \mathcal{M}_3$ .

*Exercise 4.6: Numerical computation of autoencoder curve lengths.*

Consider an autoencoder (or a similar type of model) with a two-dimensional latent space.

1. Write a computer program that evaluates the length of any second-order polynomial curve  $c$  using Eq. 4.2. It is recommended that you write the code to support any callable curve  $c$ .

## Chapter 5

### *Riemannian metrics*

*In which we measure from the inside out.*

We can get a more intrinsic understanding of the length of a curve by applying the chain rule to Eq. 4.12

$$\text{Length}_{\mathcal{M}}(c) = \int_0^1 \left\| \frac{d}{dt} f(c_t) \right\| dt, \quad (5.1)$$

$$= \int_0^1 \| \mathbf{J}_{c_t} \dot{c}_t \| dt \quad (5.2)$$

$$= \int_0^1 \sqrt{\dot{c}_t^T \mathbf{J}_{c_t}^T \mathbf{J}_{c_t} \dot{c}_t} dt, \quad (5.3)$$

where  $\dot{c}_t = dc/dt|_{t=t}$  is the curve derivative () and  $\mathbf{J}_{c_t} = \partial f / \partial \mathbf{x} |_{\mathbf{x}=c_t}$  is the Jacobian of  $f$ . While this derivation is purely mechanical, it does allow for a more elegant view of the geometry of our models.

### *5.1. Tangent spaces*

The Jacobian  $\mathbf{J}_{\mathbf{x}}$  can be seen as a local linear approximation to  $f$ . Specifically, Taylor's theorem tells us that

$$f(\mathbf{x} + \epsilon) \approx f(\mathbf{x}) + \mathbf{J}_{\mathbf{x}} \epsilon \quad (5.4)$$

for sufficiently small  $\epsilon$ . Here it is worth recalling the dimensions of the variables at hand. Since  $f : \mathbb{R}^d \rightarrow \mathbb{R}^D$ , we have

$$\mathbf{J}_{\mathbf{x}} \in \mathbb{R}^{D \times d} \quad (5.5)$$

$$\mathbf{x}, \epsilon \in \mathbb{R}^d \quad (5.6)$$

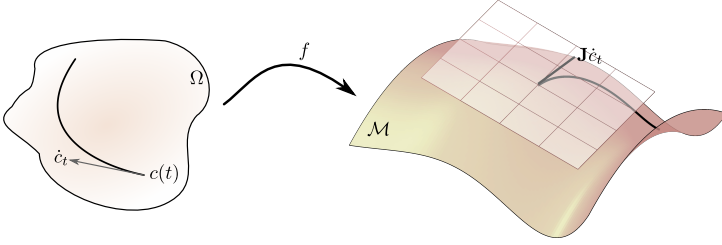


FIG. 5.1. A manifold is locally Euclidean, which is reflected by the tangent space of our manifold. For immersed manifolds, the Jacobian of  $f$  provides us with a basis for this tangent space.

$$\mathbf{y} = f(\mathbf{x}) \in \mathbb{R}^D. \quad (5.7)$$

The Jacobian  $\mathbf{J}_{\mathbf{x}}$  can then be understood to form a basis of a  $d$ -dimensional hyperplane in  $\mathbb{R}^D$  that is tangential to  $f(\mathbf{x})$ . We call this hyperplane the *tangent space* at  $\mathbf{x}$  and denote it  $T_{\mathbf{x}}$ . Returning to the viewpoint that manifolds are spaces that are locally Euclidean, we can think of the tangent space as the Euclidean interpretation of the manifold, which is valid in a local neighborhood around  $\mathbf{x}$ .

Here it is worth repeating that for a manifold to be ‘immersed’, we require that the Jacobian is full rank everywhere. That is, the tangent space of the manifold should have dimension  $d$  at all points on the manifold. If this is not the case, then it implies that the manifold locally collapses to have a dimension less than  $d$ .

## 5.2. Local inner products

Having direct access to a local Euclidean view of the manifold is very valuable, and we have already made use of this. When writing the length of a curve as in Eq. 5.2,

$$\text{Length}_{\mathcal{M}}(c) = \int_0^1 \|\mathbf{J}_{c_t} \dot{c}_t\| dt, \quad (5.8)$$

we can heuristically write this as

$$= \int_0^1 \|\mathbf{J}_{c_t} (\dot{c}_t dt)\|. \quad (5.9)$$

That is, we may think of  $\dot{c}_t dt$  as an infinitesimal vector, which we then express in the tangent space, where we measure its length. We emphasize the infinitesimal aspect as the Euclidean view is only valid locally. This construction informs us how we can measure the norm of an infinitesimal vector; the length of a curve is then found by integrating this infinitesimal norm.

Likewise, we can construct an inner product between two infinitesimal vectors. Assume that  $\delta_1$  and  $\delta_2$  are two such vectors, which we interpret as infinitesimal displacements from a point  $\mathbf{x}$ . We can then map both vectors to the tangent space at  $\mathbf{x}$  and compute the usual Euclidean inner product there,

$$\langle \delta_1, \delta_2 \rangle_{\mathbf{x}} = (\mathbf{J}_{\mathbf{x}} \delta_1)^\top (\mathbf{J}_{\mathbf{x}} \delta_2) = \delta_1^\top \mathbf{J}_{\mathbf{x}}^\top \mathbf{J}_{\mathbf{x}} \delta_2. \quad (5.10)$$

Notice how we place a subscript on the inner product  $\langle \cdot, \cdot \rangle_{\mathbf{x}}$  to denote that the inner product should be computed in the tangent space at  $\mathbf{x}$ . An alternative way to arrive at the same expression is to compute the Euclidean inner product between vectors  $(f(\mathbf{x} + \delta_1) - f(\mathbf{x}))$  and  $(f(\mathbf{x} + \delta_2) - f(\mathbf{x}))$ , which we can, again, do using Taylor's theorem

$$\langle f(\mathbf{x} + \delta_1) - f(\mathbf{x}), f(\mathbf{x} + \delta_2) - f(\mathbf{x}) \rangle \quad (5.11)$$

$$= (f(\mathbf{x}) + \mathbf{J}_{\mathbf{x}} \delta_1 - f(\mathbf{x}))^\top (f(\mathbf{x}) + \mathbf{J}_{\mathbf{x}} \delta_2 - f(\mathbf{x})) \quad (5.12)$$

$$= \delta_1^\top \mathbf{J}_{\mathbf{x}}^\top \mathbf{J}_{\mathbf{x}} \delta_2. \quad (5.13)$$

### 5.3. Local norms and angles

Given a vector  $\mathbf{v}$  in the tangent space at  $\mathbf{x}$ , we can measure its norm using the local inner product,

$$\|\mathbf{v}\|_{\mathbf{x}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\mathbf{x}}} \quad (5.14)$$

$$= \|\mathbf{J}_{\mathbf{x}} \mathbf{v}\|. \quad (5.15)$$

If two curves  $c_1$  and  $c_2$  on the manifold intersect at a point  $\mathbf{x}$  we can provide a notion of *angle* between the curves. Recall, that the angle between two vectors  $\mathbf{v}_1$  and  $\mathbf{v}_2$  is commonly defined as

$$\cos \theta = \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{\|\mathbf{v}_1\| \cdot \|\mathbf{v}_2\|}, \quad (5.16)$$

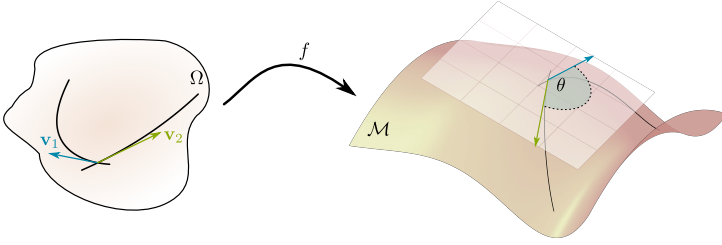


FIG. 5.2. The angle between two curves can be computed in the tangent space of the point of intersection as the angle between velocity vectors.

where  $\theta$  is the angle of interest. If we now pick vectors  $\mathbf{v}_1$  and  $\mathbf{v}_2$  as curve velocities of  $c_1$  and  $c_2$ , respectively, at the point of intersection, we can then define the angle between the curves as the angle between the velocity vectors. Figure 5.2 below illustrates this intuitive concept.

We can formalize this idea by first determining the point of intersection in terms of curve parameters. We let  $t_1$  and  $t_2$  be defined such that

$$c_1(t_1) = \mathbf{x} = c_2(t_2). \quad (5.17)$$

We need this notation to ensure that we compute velocity vectors at the correct points along the curves,

$$\mathbf{v}_1 = \dot{c}_1(t_1) \quad (5.18)$$

$$\mathbf{v}_2 = \dot{c}_2(t_2), \quad (5.19)$$

where  $\dot{c} = dc/dt$  is short-hand for curve derivatives. The angle between the curves then becomes

$$\theta = \cos^{-1} \left( \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle_{\mathbf{x}}}{\|\mathbf{v}_1\|_{\mathbf{x}} \cdot \|\mathbf{v}_2\|_{\mathbf{x}}} \right) \quad (5.20)$$

$$= \cos^{-1} \left( \frac{\mathbf{v}_1^T \mathbf{J}_{\mathbf{x}}^T \mathbf{J}_{\mathbf{x}} \mathbf{v}_2}{\|\mathbf{J}_{\mathbf{x}} \mathbf{v}_1\| \cdot \|\mathbf{J}_{\mathbf{x}} \mathbf{v}_2\|} \right). \quad (5.21)$$

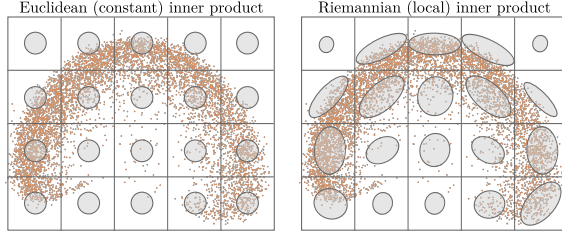


FIG. 5.3. *In a Euclidean view, the inner product structure is the same everywhere in space. In the Riemannian view, this changes locally. The inner product is such that local unit circles have an ellipsoidal shape.*

### 5.4. Riemannian metrics

The Jacobian  $\mathbf{J}$  appears practically everywhere we seek to make any sort of quantitative statement, such as a distance or an angle. You may have noticed that actually, the Jacobian tends to appear in the form  $\mathbf{J}^\top \mathbf{J}$ , e.g., as with inner products (as defined in Eq. 5.10)

$$\langle \delta_1, \delta_2 \rangle_{\mathbf{x}} = \delta_1^\top \mathbf{J}_{\mathbf{x}}^\top \mathbf{J}_{\mathbf{x}} \delta_2. \quad (5.22)$$

As it turns out, we rarely need the Jacobian directly, but rather  $\mathbf{J}^\top \mathbf{J}$ . We, therefore, give this matrix a name, call it the *metric*, and even give it its own symbol,

$$\mathbf{G}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^\top \mathbf{J}_{\mathbf{x}}, \quad (5.23)$$

where the subscript  $\mathbf{x}$  denotes that we are looking at the metric at the point  $\mathbf{x} \in \Omega$ . With this notation, we can now write inner products as

$$\langle \delta_1, \delta_2 \rangle_{\mathbf{x}} = \delta_1^\top \mathbf{G}_{\mathbf{x}} \delta_2. \quad (5.24)$$

One interpretation is that we locally define distances such that unit circles are not round but rather ellipsoidal (see fig. 5.3). The eigenvectors of  $\mathbf{G}_{\mathbf{x}}$  then provide the principal directions of these ellipsoids, much like how the eigenvectors of a covariance matrix give the principal components.

*Property 1: positive definiteness*

Since  $\mathbf{J} \in \mathbb{R}^{D \times d}$ , then  $\mathbf{G}$  is a  $d \times d$  symmetric matrix. When the manifold of interest is immersed, i.e., locally invertible, then  $\mathbf{J}$  has full rank and  $\mathbf{G}$  becomes positive definite<sup>1</sup> by construction. We can also show this directly by forming a singular value decomposition (SVD) of  $\mathbf{J}$  as

$$\mathbf{J} = \mathbf{U}\mathbf{S}\mathbf{V}^\top, \quad (5.25)$$

where  $\mathbf{U} \in \mathbb{R}^{D \times D}$ ,  $\mathbf{S} \in \mathbb{R}^{D \times d}$  and  $\mathbf{V} \in \mathbb{R}^{d \times d}$ .

Here  $\mathbf{U}$  and  $\mathbf{V}$  are rotations, i.e.  $\mathbf{U}^\top \mathbf{U} = \mathbf{I}$  and  $\mathbf{V}^\top \mathbf{V} = \mathbf{I}$ , and  $\mathbf{S}$  is a (rectangular) diagonal matrix with non-zero entries. This is just the usual definition of the SVD. Forming the metric, we now see

$$\mathbf{G} = \mathbf{J}^\top \mathbf{J} \quad (5.26)$$

$$= (\mathbf{U}\mathbf{S}\mathbf{V}^\top)^\top (\mathbf{U}\mathbf{S}\mathbf{V}^\top) \quad (5.27)$$

$$= \mathbf{V}\mathbf{S}^\top \mathbf{U}^\top \mathbf{U} \mathbf{S} \mathbf{V}^\top \quad (5.28)$$

$$= \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top, \quad (5.29)$$

where  $\mathbf{\Lambda} = \mathbf{S}^\top \mathbf{S} \in \mathbb{R}^{d \times d}$  is a (square) diagonal matrix with strictly positive entries. We recognize this as an eigenvalue decomposition of the metric, and since the eigenvalues are positive, we declare the metric to be positive definite.

This is important as now the inner product is also positive definite, which ensures nice things. For instance, the distance between two distinct points cannot be zero when the inner product is positive definite. Do note that the argument above rests on the assumption that the manifold is immersed, such that  $\mathbf{J}$  has full rank. Should this not be the case, then the local inner product is no longer positive definite and can consequently no longer be seen as an inner product.

*Property 2: basis independence*

One could certainly argue that since the metric is directly computed from the Jacobian, it is silly to introduce yet another term alongside new symbols. Is it just pointless obscurification?

---

<sup>1</sup>i.e., all eigenvalues are strictly positive

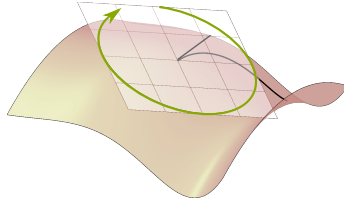


FIG. 5.4. *The particular choice of basis of our tangent space is rarely all that important. What matters is that we get norms and angles correct.*

One counterargument is that the Jacobian is a basis of the tangent space, but it is not the only possible basis. Let  $\mathbf{R}$  be a  $D \times D$  rotation matrix (i.e.,  $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$ ) such that  $\tilde{\mathbf{J}} = \mathbf{R}\mathbf{J}$  spans the same tangent space. The corresponding metric

$$\tilde{\mathbf{G}} = \tilde{\mathbf{J}}^\top \tilde{\mathbf{J}} = (\mathbf{R}\mathbf{J})^\top \mathbf{R}\mathbf{J} = \mathbf{J}^\top \mathbf{R}^\top \mathbf{R}\mathbf{J} = \mathbf{G} \quad (5.30)$$

is, however, identical to the previously defined metric. Working with the metric instead of the Jacobian, thus, allows us to not worry about the choice of basis. At times, this can be rather convenient. We will also later see that the metric allows for elegant interpretations of computations on manifolds.

## 5.5. Riemannian metrics and manifolds

We can view the metric as a function

$$\mathbf{G}_{\mathbf{x}} : \Omega \rightarrow \text{Sym}_+^{d \times d}, \quad (5.31)$$

where  $\text{Sym}_+^{d \times d}$  denote the set of real  $d \times d$  symmetric positive definite matrices. If  $\mathbf{G}_{\mathbf{x}}$  is a smooth function, then we say that it is a *Riemannian metric*. We further say that  $\mathcal{M}$  is a *Riemannian manifold* if it has an associated Riemannian metric.

It is worth remarking that whatever smoothness properties  $\mathbf{G}$  might possess comes directly from the smoothness properties of the Jacobian, which in turn comes from  $f$ . So, when we ask that the metric behaves smoothly we are merely asking that  $f$  is smooth.

But why do we care that the metric is smooth? The answer is simply that having access to derivatives is convenient. If we want to fit



a model on top of a manifold, it is practical that we can easily optimize model parameters. This, in turn, will often require (through the chain rule) that we can take derivatives of the metric.

### 5.6. Measuring on abstract manifolds

Once the metric has been defined, we can redefine local inner products, curve lengths, and the like in terms of the metric. To be explicit, we can define

$$\text{Inner product:} \quad \langle \delta_1, \delta_2 \rangle_{\mathbf{x}} = \delta_1^T \mathbf{G}_{\mathbf{x}} \delta_2 \quad (5.32)$$

$$\text{Norm:} \quad \|\mathbf{v}\|_{\mathbf{x}} = \sqrt{\mathbf{v}^T \mathbf{G}_{\mathbf{x}} \mathbf{v}} \quad (5.33)$$

$$\text{Angle:} \quad \theta = \cos^{-1} \left( \frac{\mathbf{v}_1^T \mathbf{G}_{\mathbf{x}} \mathbf{v}_2}{\sqrt{\mathbf{v}_1^T \mathbf{G}_{\mathbf{x}} \mathbf{v}_1} \cdot \sqrt{\mathbf{v}_2^T \mathbf{G}_{\mathbf{x}} \mathbf{v}_2}} \right) \quad (5.34)$$

$$\text{Curve length:} \quad \text{Length}_{\mathcal{M}}(c) = \int_0^1 \sqrt{\dot{c}_t^T \mathbf{G}_{c_t} \dot{c}_t} dt. \quad (5.35)$$

These only depend on the metric being used for computing local inner products.

Since we can compute all elementary quantities of interest from the metric alone, it is worth asking if we can forget all about  $f$  and  $\mathbf{J}$  and only care about the metric. This leads to the notion of *abstract manifolds*, which are simply manifolds that come with a metric, but where an explicit embedding (or immersion) is not known. Mathematically, these are often studied as they are both elegant abstractions, and often have less notional clutter than embedded and immersed manifolds.

Abstract manifolds may, however, also be very practical. At times, defining a metric to have desirable properties may simply be an easy way to ensure that your model behaves the way you want it. If, for example, we want a model where geodesics avoid particular regions, then we can equip this region with a metric that has larger numerical values than for other regions. Geodesics will then stay clear of this region as paths otherwise become excessively long. We shall return to such considerations in Chapter 11.

## Exercises

### Exercise 5.1: Spherical metric.

Let  $x_1 \in (0, \pi)$  and  $x_2 \in [0, 2\pi)$  be the latent coordinates of the manifold spanned by the function  $f : (0, \pi) \times [0, 2\pi) \rightarrow \mathbb{R}^3$

$$f(x_1, x_2) = \begin{pmatrix} \sin(x_1) \cos(x_2) \\ \sin(x_1) \sin(x_2) \\ \cos(x_1) \end{pmatrix}. \quad (5.36)$$

This will span the unit sphere.

1. Derive the Jacobian matrix of  $f$ .
2. Derive the metric associated with  $f$  according to Eq. 5.23.
3. Show that the metric is positive definite.

### Exercise 5.2: Quadratic metric.

Consider a two-dimensional abstract manifold with the metric

$$\mathbf{G}_{\mathbf{x}} = (1 + \|\mathbf{x}\|^2) \mathbf{I}, \quad \mathbf{x} \in \mathbb{R}^2. \quad (5.51)$$

Consider the points

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}. \quad (5.52)$$

1. Compute the local norms of the tangent vector  $\mathbf{v}_1 = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top$  assuming the point of tangency is  $\mathbf{x}_1$ ,  $\mathbf{x}_2$  and  $\mathbf{x}_3$ , respectively.
2. Compute the local angles between  $\mathbf{v}_1$  and  $\mathbf{v}_2 = \begin{pmatrix} 0 & 1 \end{pmatrix}^\top$  in the same three points of tangency.

### Exercise 5.3: Regressed metric.

Consider an abstract manifold generated a set of pre-determined positive definite matrices  $\{\mathbf{G}_k\}_{k=1}^K$ . Let the metric be given by [22]

$$\mathbf{G}_{\mathbf{x}} = \sum_{k=1}^K w_k(\mathbf{x}) \mathbf{G}_k, \quad (5.63)$$

where  $w_k : \mathbb{R}^d \rightarrow [0, 1]$  are positive functions that sum to one,  $\sum_k w_k(\mathbf{x}) = 1$  for all  $\mathbf{x} \in \mathbb{R}^d$ .

Consider the dataset available at

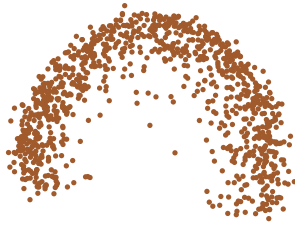


FIG. 5.5. The toy-banana dataset is a subset of 1-digits from MNIST expressed in its leading principal components and some example geodesics.

<http://hauberg.org/weekendwithbernie/toybanana.npy>

This consists of  $N = 992$  observations in  $\mathbb{R}^2$  (see Fig. 5.5).

1. Perform  $K$ -means clustering on the data to produce  $K = 5$  clusters. For each cluster, compute the covariance matrix of the data within the cluster.
2. Define the local metrics  $\mathbf{G}_k$  as the inverse covariance matrix of each cluster. Develop  $K$  positive functions  $w_k : \mathbb{R}^d \rightarrow [0, 1]$ , such that a function  $w_k$  output large values for values of  $\mathbf{x}$  near the data in cluster  $k$ , and small values otherwise. Be creative on how to estimate this function – there’s plenty of space to have fun here.
3. Plot the determinant of the regressed metric (5.63) on a grid alongside the data. How do you interpret the plot?

*Exercise 5.4: Autoencoder metric.*

Consider the autoencoder that you trained in exercise 1.2 (if you did not solve this exercise, not might be a good time).

1. Pick a random training point in the latent space. Evaluate the Riemannian metric (5.23) at the point. Hint: it is recommended to use tools from automatic differentiation to compute the Jacobian.
2. Compute the eigenvalues of the metric. Is the metric positive definite?

## Chapter 6

### *Examples of learned metrics*

*In which we take an exemplary intermission.*

So far, we have focused on the mathematical properties of manifolds and their metrics. We now change scenery and look at examples learned from data.

#### *6.1. Abstract density metrics*

Our general focus, thus far, has been on models where the manifold is defined through a mapping  $f : \mathbb{R}^d \rightarrow \mathbb{R}^D$ , but one can also learn the metric directly. Consider a data set  $\mathbf{y}_{1:N} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$ , where  $\mathbf{y}_n \in \mathbb{R}^D$ . We may then assign a smoothly changing metric  $\mathbf{G}_{\mathbf{y}}$  to each point in  $\mathbb{R}^D$ , and use the developed tools directly. The simplest approach is to pick the metric as

$$\mathbf{G}_{\mathbf{y}} = g(\mathbf{y})\mathbf{I}, \tag{6.1}$$

where  $g : \mathbb{R}^D \rightarrow \mathbb{R}_+$  is a learned function that controls the metric. If  $g$  is smooth, then  $\mathbf{G}_{\mathbf{y}}$  is also smooth. Since  $g$  is positive, then, by construction,  $\mathbf{G}_{\mathbf{y}}$  is positive definite, and we may view  $\mathbf{G}_{\mathbf{y}}$  as a Riemannian metric.

At this point, we may construct  $g$  to have whichever properties are useful for the application at hand. As an intuitive example, consider picking  $g$  to be inversely proportional to the data density [31], i.e.

$$g(\mathbf{y}) = \frac{1}{p(\mathbf{y})}. \tag{6.2}$$

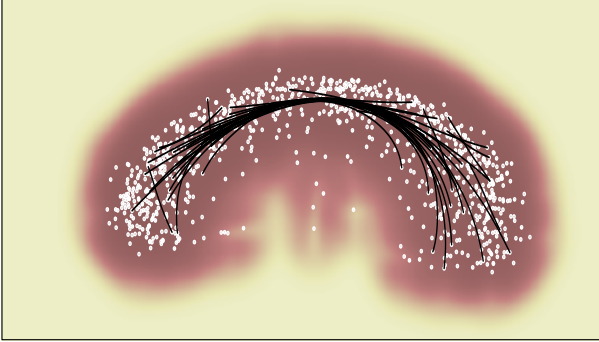


FIG. 6.1. *Direct learning of a Riemannian metric. The background color reflects the learned function  $g$  as in Eq. 6.2. As  $g$  takes small values near the data, geodesics are naturally drawn towards regions of high density.*

Here  $p(\mathbf{y})$  is any smooth estimate of the density, e.g., a *kernel density* estimate [19] or even a *deep generative model* [47]. This metric has the property that it takes on small values where we observe data, and high values otherwise. The length of a curve is, therefore, longer in regions far away from the data than near the data. The shortest paths are then naturally drawn toward the data. Figure 6.1 below provides an example of this behavior. Whether this is a useful property or not depends on the application.

Clearly, we need not restrict ourselves to the case of a scaled identity matrix (6.1) and can extend trivially to diagonal or general positive definite metrics. For examples of such, see [31, 22, 5].

## 6.2. Autoencoder metrics

The *autoencoder* [41] is a prototypical example of the manifolds discussed thus far. Briefly put, this model class defines a manifold through a *decoder* function  $f : \mathbb{R}^d \rightarrow \mathbb{R}^D$  and approximates the *projection* onto the manifold with an *encoder* function  $g : \mathbb{R}^D \rightarrow \mathbb{R}^d$ . This pair of functions is then jointly trained by minimizing the squared distance between

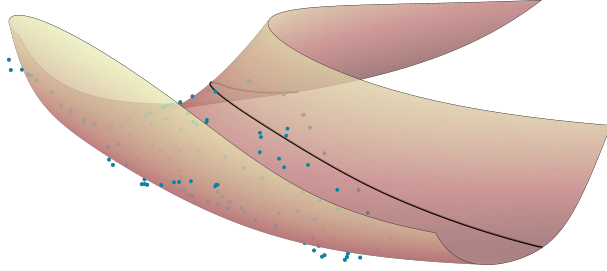


FIG. 6.2. A manifold learned from data (blue) and an example geodesic (black). Note how the geodesic avoids ‘climbing the steep hill’ of the manifold.

observations and their approximate projections onto the manifold,

$$\sum_{n=1}^N \|\mathbf{x}_n - f(g(\mathbf{x}_n))\|^2. \quad (6.3)$$

Figure 6.2 below gives a simple example where an autoencoder is used to approximate a two-dimensional manifold in  $\mathbb{R}^3$ . The figure further shows an example geodesic on this learned manifold. Notice how the geodesic avoids steep regions of the manifold as these result in very long curves. We will revisit autoencoders in Chapter 15 when discussing noisy manifolds.

### 6.3. Classification metrics

Riemannian metrics can, at times, also be used to shed light on the behavior of various predictive models. Consider, for example, a classifier  $f$  that maps a given input to a vector of probabilities over class membership, i.e.

$$f : \mathbb{R}^D \rightarrow C, \quad (6.4)$$

where  $C$  is a vector space of dimension  $K$  (number of classes) where all elements have positive entries that sum to 1. We can then assign a metric to the observation space with the regular pull-back construction (5.23)

$$\mathbf{G}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^{\top} \mathbf{J}_{\mathbf{x}}, \quad (6.5)$$

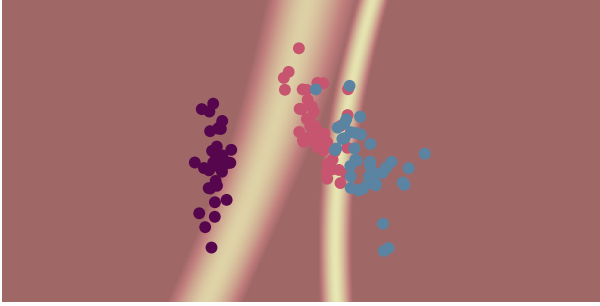


FIG. 6.3. The (square root determinant of the) pull-back metric associated with a neural network performing classification on a two-dimensional projection of the Iris dataset. The metric is then over the space of observed data.

where  $\mathbf{J}_{\mathbf{x}}$  is the Jacobian of  $f$  evaluated at  $\mathbf{x}$ . An important subtlety comes into play here: for  $\mathbf{G}_{\mathbf{x}}$  to be positive definite, we need the rank of  $\mathbf{J}_{\mathbf{x}}$  to equal the data dimension. Since the maximal rank of  $\mathbf{J}_{\mathbf{x}} \in \mathbb{R}^{D \times K}$  is  $\min(D, K - 1)$ , we see that the number of classes must be greater than the number of data dimensions for the metric to be positive definite.

Figure 6.3 below gives an example of such a classification metric. Here the observation space is two-dimensional, and the data can be partitioned into three classes. We see that the metric has significantly different behavior around the decision boundaries compared to the remainder of the observation space, which is indicative that the classifier tries to push the classes apart by ‘stretching’ the decision boundaries.

The above construction, however, breaks down when the dimension of the data exceeds the number of classes. Arguably, this is the most commonly occurring situation. The (partially) good news is that when the metric is positive semi-definite (some eigenvalues are zero), most of the geometric tools we have constructed still apply. In this case, we say that the metric is *pseudo-Riemannian* (or *semi-Riemannian*). In such models, one should be aware that distinct points may have a distance of 0, which does not match common intuition (*how can objects at distinct positions be at the same place?*). We will return to this question in Chapter 18.

## *Exercises*

### *Exercise 6.1: Density metric.*

Consider the density metric in Eq. 6.2. You will implement this for the data available at

`http://hauberg.org/weekendwithbernie/toybanana.npy`

1. *Implement a kernel density estimator for this dataset, i.e., approximate the data density with*

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \mathcal{N}(\mathbf{x}|\mathbf{x}_n, \sigma^2), \quad (6.6)$$

where  $x_n$  denotes the give observations and  $\sigma$  is a parameter you should choose.

2. *Implement the density metric.*
3. *Plot the determinant of the metric on a regular grid akin to Fig. 6.1.*

### *Exercise 6.2: Classification metric.*

We will not look at the degeneracies associated with classification models.

1. *Implement a neural network classifier for the MNIST dataset.*
2. *Implement the classification pull-back metric (6.5).*
3. *Empirically evaluate the rank of the metric at randomly chosen data points. What do you find?*



## Chapter 7

### *Geodesics*

*In which shortcuts are repeatedly taken.*

We have seen that we can define distances between two points as the length of the shortest path that connects them. But what about the shortest path itself? Is it ‘just’ an intermediate variable needed for computing distances?

Intuitively, the shortest path connecting two points is a natural choice for an *interpolant* (i.e., an interpolating curve) as it avoids unwarranted detours. Just like we often find the straight line interpolant to be natural in Euclidean spaces (where the straight line is indeed the shortest path).

#### 7.1. *What’s in a name?*

The shortest paths on a manifold are often called *geodesics*. The name originates from *geodesy*, which is the art of making measurements about the planet Earth. The word geodesic then refers to the shortest path between two points on Earth’s surface. Much of differential geometry comes from geodesy and *cartography*, i.e., the art of making maps, and often the terminology refers directly to these fields. On more general manifolds than the one reflecting Earth’s surface, we choose to reuse terminology.

#### 7.2. *Uniqueness?*

Let’s stay a tad bit longer on planet Earth, and for the sake of manageability, let’s assume that its surface is that of a sphere. On the sphere, geodesics (shortest paths) are circular arcs found through the

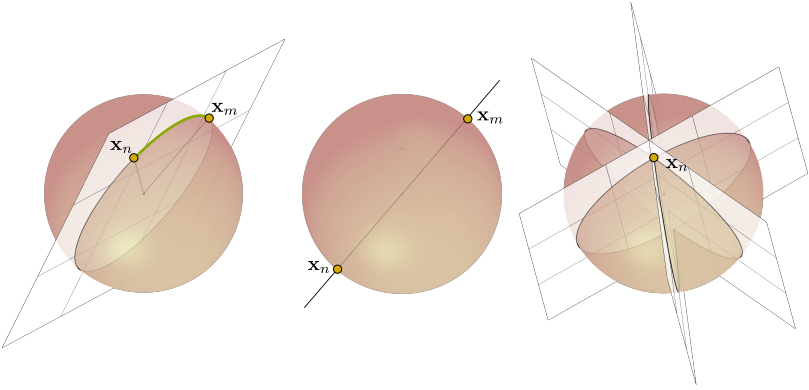


FIG. 7.1. Left: *Geodesics on a sphere may be found as the intersection of a plane and the sphere.* Center: *Antipodal points do not span a plane, and the geodesic is not unique.* Right: *In this case, we may pick any plane containing the antipodal points, such that we get infinitely many geodesics.*

intersection of the sphere and a plane going through the center of the sphere and the two points we seek to geodesically connect. This is illustrated in the left panel of Fig. 7.1. That is, given two points  $\mathbf{x}_n, \mathbf{x}_m$  and the center of the sphere  $\mathbf{o}$  we can span a two-dimensional plane going through these three points. The intersection of this plane and the sphere will define a circle within the plane. Geodesics on the sphere are arcs along this circle.

This construction, however, breaks down if the points  $\mathbf{x}_n$  and  $\mathbf{x}_m$  are *antipodal*, that is, they are opposite to each other on the sphere (think of the North and South poles). In this case, the three points  $\mathbf{x}_n, \mathbf{x}_m$ , and  $\mathbf{o}$  are linearly dependent and will only span a line rather than a plane (Fig. 7.1, center). We can, however, easily see that independently of which plane we choose that contains the three points, then the corresponding circular arc will have the same length. This implies that infinitely many geodesics exist that connect antipodal points on the sphere (Fig. 7.1, right). From this example, we conclude that in general, we cannot expect geodesics to be unique.

Since geodesics are defined as the shortest connecting curves,

$$\begin{aligned} c^* &= \underset{c}{\operatorname{argmin}} \operatorname{Length}_{\mathcal{M}}(c) \\ &\text{subject to } c(0) = \mathbf{x}_n \text{ and } c(1) = \mathbf{x}_m, \end{aligned} \tag{7.1}$$

it is not surprising that multiple geodesics may exist. We should in general expect several local optima arising from this optimization problem, and we will call *all* of the resulting curves geodesics.

### 7.3. Constant speed parametrizations

So, geodesics are not unique. To further add to the lack of non-uniqueness we have previously seen (page 30) that we can parametrize the same curve in a multitude of different ways. We can informally visualize the situation if we take an *optimization* perspective, and consider the geodesic definition of Eq. 7.1. Here we seek a curve of minimal length. We can think of the curve as having a set of parameters that determine its shape, e.g., the shape of a polynomial is determined by its coefficients. For any set of curve parameters, we can then evaluate the length of the corresponding curve, and we can then visualize the optimization landscape associated with the geodesic problem (7.1). Figure 7.2 provides a concept drawing of this landscape. Here the black curves correspond to different reparametrizations of different locally optimal solutions to the geodesic optimization problem. Here we are relying on a concept drawing rather than an actual numerical example as general smooth curves have infinitely many control parameters, which is at odds with the idea of making a two-dimensional plot. Nonetheless, it is valuable to attempt to imagine how messy the optimization landscape becomes when optima are never points, but always form continuums.

That geodesics need not be unique is fundamental: sometimes there is simply more than one good way to connect two points. That we cannot change. The reparametrization issue we can, in contrast, do something about. The solution is merely to introduce a convention that makes the reparametrization issue go away. If the same curve can be parametrized in different ways, we will simply pick one particular parametrization which we will then always use. Whatever choice we

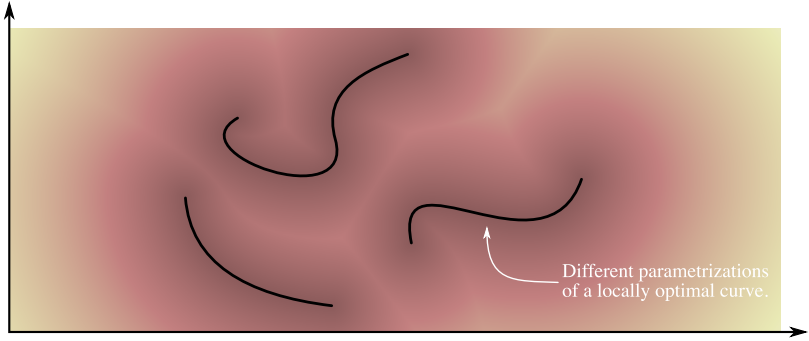


FIG. 7.2. The optimization landscape from curve length minimization. Here coordinates are different curve parameters. Due to reparametrization issues each local optimum forms a continuum (black curves) along which corresponding curves have the same shape and length.

make here will be arbitrary, so we might as well make a choice that is *simple*. Since parametrization changes the speed with which we traverse a curve, the simplest choice seems to be that we should traverse the curve with constant speed. That is if we have a curve  $c : [0, 1] \rightarrow \mathbb{R}^D$  with length

$$\text{Length}_{\mathcal{M}}[c] = \int_0^1 \|\dot{c}_t\| dt, \quad (7.2)$$

then its speed  $\|\dot{c}_t\|$  should be constant (independent of  $t$ ). If we let  $v = \|\dot{c}_t\|$  denote this constant, then the length simplifies to

$$\text{Length}_{\mathcal{M}}[c] = \int_0^1 v dt \quad (7.3)$$

$$= v \int_0^1 dt = v. \quad (7.4)$$

That is, the length of a constant speed curve can be found by computing the speed  $\|\dot{c}_t\|$  at any point rather than by evaluating an integral. So, not only can we now have a unique (by convention) parametrization of a curve, we also get the added benefit that length computations significantly simplify.

In practice, one can always reparametrize a given curve to have a constant speed, but rather than develop techniques for this, we will next see how we can get constant speed curves with no additional effort.

## 7.4. Curve energy

Instead of computing shortest paths by minimizing the length of a curve, we will now derive an alternative objective function to minimize. This will have the property that solution curves are the shortest paths, but it will also ensure that solution curves have a constant speed.

To arrive at this, we first recall the Cauchy-Schwarz inequality

$$|\langle u, v \rangle| \leq \|u\| \|v\|, \quad (7.5)$$

where the equality is satisfied if and only if  $u$  and  $v$  are parallel. This holds in any inner product space with the usual convention that  $\|u\| = \sqrt{\langle u, u \rangle}$ . We will use this to first bound the length of a curve, and for this, we will need an inner product between curves. The usual choice is

$$\langle u, v \rangle = \int_0^1 u_t v_t dt \quad (7.6)$$

for curves  $u, v : [0, 1] \rightarrow \mathbb{R}$ . In order to bound the length of a curve  $c$ , we introduce the curve  $u : [0, 1] \rightarrow \mathbb{R}$  that measures the speed of  $c$ ,

$$u_t = \|\dot{c}_t\|. \quad (7.7)$$

In order to apply the Cauchy-Schwarz inequality, we also introduce the seemingly silly constant curve  $v : [0, 1] \rightarrow \mathbb{R}$

$$v_t = 1. \quad (7.8)$$

With these definitions in place, we see that the length of the curve  $c$  can be written as an inner product,

$$\text{Length}_{\mathcal{M}}[c] = \int_0^1 \|\dot{c}_t\| dt = \langle u, v \rangle. \quad (7.9)$$

Then by the Cauchy-Schwarz inequality, we have

$$\text{Length}_{\mathcal{M}}[c] \leq \|u\| \|v\| \quad (7.10)$$

$$= \sqrt{\int_0^1 \|\dot{c}_t\|^2 dt} \sqrt{\int_0^1 1^2 dt} \quad (7.11)$$

$$= \sqrt{\int_0^1 \|\dot{c}_t\|^2 dt}. \quad (7.12)$$

This last integral turns out to be important, so we give it a name: the *energy* of a curve and denote it

$$\mathcal{E}[c] = \int_0^1 \|\dot{c}_t\|^2 dt. \quad (7.13)$$

If we square both sides of Eq. 7.12 we see that

$$\text{Length}_{\mathcal{M}}^2[c] \leq \mathcal{E}[c]. \quad (7.14)$$

This is a direct application of the Cauchy-Schwarz inequality. Here we know that equality is satisfied if and only if  $u$  and  $v$  are parallel, i.e., if they are proportional to each other,

$$u_t = \alpha v_t \quad \Leftrightarrow \quad \|\dot{c}_t\| = \alpha. \quad (7.15)$$

In other words, we achieve equality in Eq. 7.14 if and only if the curve speed  $\|\dot{c}_t\|$  is constant.

We now have all the ingredients to realize that if we minimize curve energy, then we get a curve of minimal length with constant speed. Since curve energy upper bounds squared curve length, we know that the smallest energy a curve can realize is the squared geodesic distance. We also know that a curve will only ever realize this minimal energy if it has a constant speed. From this, we see that by minimizing curve energy (7.13) we recover a geodesic with constant speed.

Contrasting energy minimization to length minimization we now get an optimization landscape that is significantly more well-behaved (Fig. 7.3 vs. Fig. 7.2).

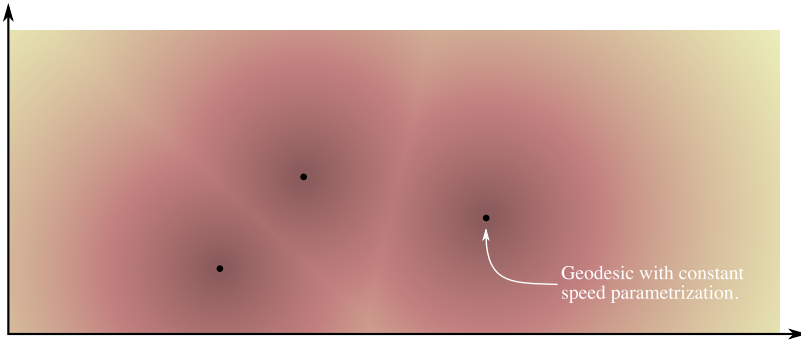


FIG. 7.3. The optimization landscape from curve energy minimization. This figure should be contrasted with that of length minimization (Fig. 7.2). Now optima are well isolated and correspond to constant speed curves.

## 7.5. Governing differential equations

Both computationally and mathematically we often need a richer language for understanding geodesics. We can get this through a system of *ordinary differential equations* (ODEs) that capture the behavior of geodesics.

A geodesic is a minimizer of curve energy (7.13), and informally this should imply that the derivative of energy with respect to the curve should be zero for geodesics. As a curve is not a vector, this statement is not quite true, and we must turn to the Euler-Lagrange equations for a more formal statement. The Euler-Lagrange equations state that integrals of the form

$$\int_0^1 E(t, c_t, \dot{c}_t) dt, \quad (7.16)$$

where  $E : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ , are minimized with respect to a curve  $c$  when

$$\frac{\partial E_t}{\partial c_t} = \frac{d}{dt} \frac{\partial E_t}{\partial \dot{c}_t}. \quad (7.17)$$

In our particular case of energy-minimizing curves, we would have

$$E_t = E(t, c_t, \dot{c}_t) = \dot{c}_t^T \mathbf{G}_{c_t} \dot{c}_t. \quad (7.18)$$

The corresponding Euler-Lagrange equations are then a system of ODEs that characterize geodesic curves. We will derive these in the next section, but first, present the result

$$[\ddot{c}_t]_m = - \sum_{j=1}^d \sum_{k=1}^d \Gamma_{jk}^m(c_t) [\dot{c}_t]_j [\dot{c}_t]_k \quad (7.19)$$

$$\Gamma_{jk}^m(c_t) = \frac{1}{2} \sum_{i=1}^d [\mathbf{G}_{c_t}^{-1}]_{mi} \left( \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} + \frac{d[\mathbf{G}_{c_t}]_{ik}}{d[c_t]_j} - \frac{d[\mathbf{G}_{c_t}]_{jk}}{d[c_t]_i} \right), \quad (7.20)$$

where  $[\cdot]_i$  denotes an indexing into the  $i^{\text{th}}$  dimension of a vector, and similarly with matrix indexing  $[\cdot]_{ij}$ . Clearly, this is a notional mess, but such is the situation.

The  $\Gamma_{jk}^m(c_t)$  coefficients are called the *Christoffel symbols*. Informally, they reflect how much the metric changes relative to the metric itself – Eq. 7.20 can be read as the derivative of the metric divided by the metric. This is indicative of how much the manifold curves at a given point. The Christoffel symbols carry a deeper geometric interpretation that we won't touch on here. It is, however, worth noting that they are symmetric,  $\Gamma_{jk}^m = \Gamma_{kj}^m$ , which can lead to computational savings.

## 7.6. Deriving the geodesic ODE



The derivation of the geodesic ODE (7.19) is purely mechanical and is, as indicated by the resulting equation, largely a matter of keeping track of indices. As such, the hurried reader is encouraged to skip this section. With such warnings out of the way, we may proceed.

The Euler-Lagrange equation (7.17) is easy to work with: evaluate the individual terms, and put them together. As we will be doing this in coordinates, we first write  $E_t$  in such,

$$E_t = \dot{c}_t^T \mathbf{G}_{c_t} \dot{c}_t = \sum_{j=1}^d \sum_{k=1}^d [\mathbf{G}_{c_t}]_{jk} [\dot{c}_t]_j [\dot{c}_t]_k. \quad (7.21)$$

Next, we write the left-hand side of the Euler-Lagrange equations as

$$\frac{dE_t}{d[c_t]_i} = \sum_{j=1}^d \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{jk}}{d[c_t]_i} [\dot{c}_t]_j [\dot{c}_t]_k. \quad (7.22)$$



To evaluate the right-hand side of the Euler-Lagrange equation, we first note that

$$\frac{\partial E_t}{\partial \dot{c}_t} = 2\mathbf{G}_{c_t}\dot{c}_t. \quad (7.23)$$

Differentiating this with respect to  $t$  then yields the right-hand side,

$$\frac{d}{dt} \frac{dE_t}{d[\dot{c}_t]_i} = \frac{d}{dt} \left( 2 \sum_{j=1}^d [\mathbf{G}_{c_t}]_{ij} [\dot{c}_t]_j \right) \quad (7.24)$$

$$= 2 \sum_{j=1}^d \left( \frac{d[\mathbf{G}_{c_t}]_{ij}}{dt} [\dot{c}_t]_j + [\mathbf{G}_{c_t}]_{ij} [\ddot{c}_t]_j \right) \quad (7.25)$$

$$= 2 \sum_{j=1}^d \left( \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} [\dot{c}_t]_k [\dot{c}_t]_j + [\mathbf{G}_{c_t}]_{ij} [\ddot{c}_t]_j \right). \quad (7.26)$$

We can now simply insert Eqs. 7.22 and 7.26 into the Euler-Lagrange equation (7.17) to get

$$\begin{aligned} \sum_{j=1}^d \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{jk}}{d[c_t]_i} [\dot{c}_t]_j [\dot{c}_t]_k \\ = 2 \sum_{j=1}^d \left( \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} [\dot{c}_t]_k [\dot{c}_t]_j + [\mathbf{G}_{c_t}]_{ij} [\ddot{c}_t]_j \right). \end{aligned} \quad (7.27)$$

Isolating the term with second-derivatives ( $\ddot{c}_t$ ) yields

$$\sum_{j=1}^d [\mathbf{G}_{c_t}]_{ij} [\ddot{c}_t]_j = -\frac{1}{2} \sum_{j=1}^d \sum_{k=1}^d \left( 2 \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} - \frac{d[\mathbf{G}_{c_t}]_{jk}}{d[c_t]_i} \right) [\dot{c}_t]_j [\dot{c}_t]_k. \quad (7.28)$$

The left-hand side is merely the matrix-vector product  $[\mathbf{G}_{c_t}\ddot{c}_t]_i$ , so we can multiply by the inverse metric to fully isolate  $\ddot{c}_t$

$$[\ddot{c}_t]_m = -\frac{1}{2} \sum_{i=1}^d [\mathbf{G}_{c_t}^{-1}]_{mi} \sum_{j=1}^d \sum_{k=1}^d \left( 2 \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} - \frac{d[\mathbf{G}_{c_t}]_{jk}}{d[c_t]_i} \right) [\dot{c}_t]_j [\dot{c}_t]_k.$$

To emphasize the symmetry inherent in these equations, we note that

$$2 \sum_{j=1}^d \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} [\dot{c}_t]_j [\dot{c}_t]_k \quad (7.29)$$

$$= \sum_{j=1}^d \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} [\dot{c}_t]_j [\dot{c}_t]_k + \sum_{j=1}^d \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} [\dot{c}_t]_j [\dot{c}_t]_k \quad (7.30)$$

$$= \sum_{j=1}^d \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} [\dot{c}_t]_j [\dot{c}_t]_k + \sum_{j=1}^d \sum_{k=1}^d \frac{d[\mathbf{G}_{c_t}]_{ik}}{d[c_t]_j} [\dot{c}_t]_k [\dot{c}_t]_j. \quad (7.31)$$

This is essentially nothing but index-swapping, but it allows us to write the geodesic ODE as

$$[\ddot{c}_t]_m = - \sum_{j=1}^d \sum_{k=1}^d \Gamma_{jk}^m(c_t) [\dot{c}_t]_j [\dot{c}_t]_k, \quad (7.32)$$

where the Christoffel symbols are

$$\Gamma_{jk}^m(c_t) = \frac{1}{2} \sum_{i=1}^d [\mathbf{G}_{c_t}^{-1}]_{mi} \left( \frac{d[\mathbf{G}_{c_t}]_{ij}}{d[c_t]_k} + \frac{d[\mathbf{G}_{c_t}]_{ik}}{d[c_t]_j} - \frac{d[\mathbf{G}_{c_t}]_{jk}}{d[c_t]_i} \right). \quad (7.33)$$

## 7.7. Alternative notation



In some computational environments, it is inconvenient that the geodesic ODE (7.19) uses many indices as a direct evaluation and then requires multiple loops over these indices. With a bit of manipulation, it is possible to express the geodesic ODE without the use of indices [6]:

$$\ddot{c}_t = -\frac{1}{2} \mathbf{G}_{c_t}^{-1} \left\{ 2(\mathbf{I} \otimes \dot{c}_t) \partial_{c_t} \text{vec}(\mathbf{G}_{c_t}) \dot{c}_t - \partial_{c_t} \text{vec}(\mathbf{G}_{c_t})^\top (\dot{c}_t \otimes \dot{c}_t) \right\}.$$

Here  $\text{vec}(\cdot)$  stacks the columns of a matrix into a vector, and  $\otimes$  is the Kronecker product. This does not exactly render the equation any easier to read. However, in some numerical environments it is more efficient to work with a vectorial, rather than coordinate-wise, expression.

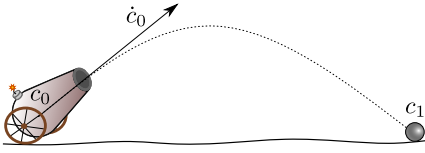


FIG. 7.4. The term shooting geodesics comes from analogy with computing the final position of a cannonball being shot from a cannon with known position and orientation. If you, like this author, are not too fond of shootings you can simply refer to this as the exponential map (see next chapter).

## 7.8. Two types of geodesics

The geodesic ODE (7.19) tells us how geodesics behave locally. This is a second-order ODE as it describes the behavior of the second derivative  $\ddot{c}_t$  of the geodesic  $c$ . As a curve satisfying the geodesic ODE is energy minimizing, we know that it must have a constant speed.

So far, we have treated geodesics as a means to computing distances between points  $c_0$  and  $c_1$ . The corresponding geodesic is then a solution to the geodesic ODE subject to the constraints that the initial point should be  $c_0$  and the terminal point should be  $c_1$ . This is known as a *two-point boundary value problem*, and little is known about such problems except that they may be difficult to solve.

A more positive story can be told for *initial value problems*. In general, when we specify the initial conditions of an ODE, the *Picard-Lindelöf theorem*<sup>1</sup> tell us that the resulting curve is unique. In terms of the geodesic ODE, an initial condition is a *starting point*  $c_0$  and an *initial velocity*  $\dot{c}_0$ . If we think of a geodesic path traversed by a person walking along the manifold, then we may say that if we know the starting point, initial direction, and how long said person should walk, then the traversed path is unique. Geodesics where we specify the initial point and velocity are often called *shooting geodesics* due to an analogy with how a cannonball flies as a function of the cannon's direction (Fig. 7.4).

---

<sup>1</sup>Readers unfamiliar with this theorem may consult Appendix A for a recap.

## Exercises

### Exercise 7.1: Euclidean metric.

Consider the Euclidean metric of  $\mathbb{R}^d$ , i.e.,  $\mathbf{G} = \mathbf{I}$ .

1. Derive the Christoffel symbols of this metric (Eq. 7.20).
2. Derive the geodesic ODE.
3. What is the geodesic that connects points  $\mathbf{x}_1$  and  $\mathbf{x}_2$ ?

### Exercise 7.2: Spherical metric.

Let  $x_1 \in (0, \pi)$  and  $x_2 \in [0, 2\pi)$  be the latent coordinates of the manifold spanned by the function  $f : (0, \pi) \times [0, 2\pi) \rightarrow \mathbb{R}^3$

$$f(x_1, x_2) = \begin{pmatrix} \sin(x_1) \cos(x_2) \\ \sin(x_1) \sin(x_2) \\ \cos(x_1) \end{pmatrix}. \quad (7.39)$$

This will span the unit sphere. In exercise 5.1 we derived that the metric in latent coordinates is

$$\mathbf{G}_{\mathbf{x}} = \begin{pmatrix} 1 & 0 \\ 0 & \sin^2(x_1) \end{pmatrix} \quad (7.40)$$

1. Derive the Christoffel symbols of this metric (Eq. 7.20). *Hint: note that some terms of the metric are zero, which renders some coefficients to be zero as well.*
2. Derive the geodesic ODE.

### Exercise 7.3: Quadratic metric.

Consider a two-dimensional abstract manifold with the metric

$$\mathbf{G}_{\mathbf{x}} = (1 + \|\mathbf{x}\|^2) \mathbf{I}, \quad \mathbf{x} \in \mathbb{R}^2. \quad (7.41)$$

1. Derive the Christoffel symbols of this metric (Eq. 7.20). *Hint: note that some terms of the metric are zero, which renders some coefficients to be zero as well.*
2. Derive the geodesic ODE.

3. Consider a geodesic  $c$  starting at  $c_0 = \mathbf{0}$  and initial velocity  $\dot{c}_0 = \mathbf{v}$ . What is the acceleration  $\ddot{c}_0$ ?

*Exercise 7.4: Isotropic metric.*

Consider the *isotropic metric* (i.e., it's a scaled identity matrix)

$$\mathbf{G}_{\mathbf{x}} = \alpha(c_t)\mathbf{I}, \quad \mathbf{x} \in \mathbb{R}^D. \quad (7.55)$$

1. Derive an expression for the curve energy under this metric.
2. Derive the Christoffel symbols of this metric (Eq. 7.20).
3. Derive the geodesic ODE.

## Chapter 8

### *Computing geodesics numerically*

*In which we actually do something practical.*

In practice, we are not satisfied with mathematical expressions for the geodesic curves: we want actual algorithms to compute them! Let's start with the bad news: *while many algorithms exist, none are sufficiently fast and stable to be considered the gold standard.* With that said, let's ignore all warnings and proceed ahead.

#### *8.1. Automatic differentiation*

To compute a geodesic, we will often need to evaluate many derivatives. For example, the geodesic ODE (7.19) requires the derivative of the metric. For many models, the metric is also composed of derivatives, e.g., the pull-back metric  $\mathbf{G} = \mathbf{J}^\top \mathbf{J}$  requires a Jacobian matrix. And so forth. Computing such derivatives is often feasible, but it is tedious ‘grunt work’.

Fortunately, tools for *automatic differentiation* [50] are plentiful and constantly evolving. These allow us to circumvent the manual calculations of derivatives, thereby both saving work and avoiding tedious-to-debug code.

At its core, an automatic differentiation system allows for computing vector products with the Jacobian of a given function. This comes in two forms, left- and right multiplication,

$$\frac{\partial f}{\partial x} \mathbf{v}_d = \mathbf{J} \mathbf{v}_d = \text{jvp}_f(\mathbf{v}_d) \quad (8.1)$$

$$\mathbf{v}_D \frac{\partial f}{\partial x} = \mathbf{v}_D \mathbf{J} = \text{vjp}_f(\mathbf{v}_D), \quad (8.2)$$

where  $\mathbf{v}_d \in \mathbb{R}^d$  and  $\mathbf{v}_D \in \mathbb{R}^D$  denotes arbitrary vectors of suitable dimension. The names `jvp` and `vjp` are short for *Jacobian-vector product* and *vector-Jacobian product*, respectively. It is exactly these two functions that form the core of systems of automatic differentiation.

Having access to these functions drastically reduces the amount of required work. For example, inner products under the metric  $\mathbf{G} = \mathbf{J}^\top \mathbf{J}$  can be evaluated as

$$\mathbf{u}^\top \mathbf{G} \mathbf{v} = (\mathbf{J} \mathbf{u})^\top (\mathbf{J} \mathbf{v}) = \text{jvp}_f(\mathbf{u})^\top \text{jvp}_f(\mathbf{v}). \quad (8.3)$$

This elementary construction is sufficiently flexible that we can compute gradients, Hessians, and so forth without manual derivations.

## 8.2. Direct length minimization

Arguably, the most obvious algorithm to compute a geodesic is to minimize the length of a curve connecting two points  $c_0$  and  $c_1$ . This, however, fails in a delightfully instructive way.

To see this failure, we first discretize the sought curve into a set of points  $\{\tilde{c}_i\}_{i=1}^{N-1}$  and create a piecewise linear curve consisting of the points  $\{\tilde{c}_0, \tilde{c}_1, \dots, \tilde{c}_{N-1}, \tilde{c}_N\}$ , where we have defined the end-points  $\tilde{c}_0 = c_0$  and  $\tilde{c}_N = c_1$ . Secondly, we construct *numerical curve length* `Curve length!numerical`,

$$\text{Numeric-Length}(\tilde{c}) = \sum_{i=1}^N \|f(\tilde{c}_i) - f(\tilde{c}_{i-1})\|. \quad (8.4)$$

Minimizing this objective with respect to the intermediate curve points, e.g., using gradient descent, should result in a working algorithm (but it doesn't).

Figure 8.1 illustrates the problem on a sphere. We provide an initial curve consisting of equidistant points along the true geodesic. We would expect that minimizing Eq. 8.4 would leave the initial curve unchanged, as it is already a geodesic. In practice, this does not happen. Instead, the intermediate curve points gradually move closer to the curve end-points, and the numerical curve becomes increasingly similar to a straight line.

In the example, the final discrete curve has a non-constant speed, leading to a poor numerical estimate of the curve length. This renders the approach practically useless, but also suggests a path forward: instead of minimizing length, we should minimize curve energy, as this ensures that the final solution has constant speed (thereby circumventing the failure mode).

### 8.3. Connecting geodesics

Given two points  $c_0$  and  $c_1$ , the simplest working algorithm to compute the connecting geodesic is *direct energy minimization*. This can be summarized as

*Alg. 8.3: Compute connecting geodesic*

1. parametrize a curve that connects  $c_0$  and  $c_1$ ,
2. numerically approximate the energy of the parametric curve, and finally
3. minimize the energy using gradient-based optimization.

This high-level approach can be realized in many ways, and we'll cover some step-by-step.

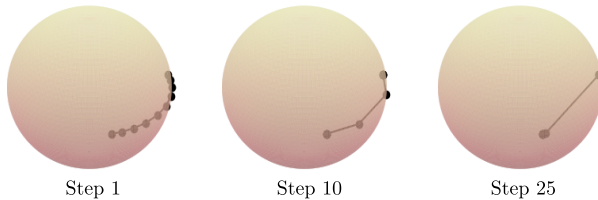


FIG. 8.1. *Optimization fails when we directly minimize the length of a discretized curve on an embedded manifold (here, a sphere).*



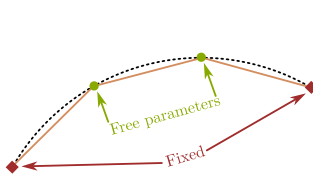


FIG. 8.2. A continuous curve (dashed, black) is approximated by a piecewise linear curve (brown). The endpoints (red diamonds) are considered fixed, while the green circles are considered free parameters that can move during optimization.

### Step 1, take 1: piecewise linear curves

The simplest way to parameterize a curve  $c : [0, 1] \rightarrow \mathbb{R}^d$  is through a piecewise linear function. We start with a partitioning of the interval  $[0, 1]$  with  $N + 1$  equidistant points,

$$t_i = \frac{i}{N}, \quad \text{for } i = 0, \dots, N. \quad (8.5)$$

For example, for  $N = 3$ , we have  $\{0, 1/3, 2/3, 1\}$ . For each  $t_i$ , we keep an associated position along the curve  $\tilde{c}_i \in \mathbb{R}^d$ . Since the end-points of the requested geodesic are known, we let  $\tilde{c}_0 = c_0$  and  $\tilde{c}_N = c_1$ . The remaining points  $\{\tilde{c}_i\}_{i=1}^{N-1}$  are considered *free parameters* of the curve representation. When computing geodesics, we will optimize with respect to these free parameters (see Fig. 8.2).

### Step 1, take 2: polynomial curves

On smooth manifolds, geodesics are smooth curves. Some readers may, therefore, object to the idea of using a piecewise linear curve to represent geodesics. Here we consider polynomials as an alternative, but any parametric curve (splines, Bezier curves, neural networks, etc.) would do. As a general strategy, consider decomposing a curve  $c$  into linear and nonlinear parts,

$$c(t) = (1 - t)c_0 + tc_1 + \tilde{c}(t), \quad (8.6)$$

where  $\tilde{c}$  is constructed to have endpoints at zero, i.e.,  $\tilde{c}(0) = \tilde{c}(1) = \mathbf{0}$ . This construction is illustrated in Fig. 8.3. This decomposition neatly allows you to change the curve endpoints with little effort.

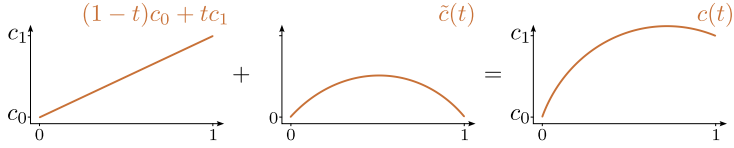


FIG. 8.3. A curve  $c(t)$  is split into a linear part,  $(1-t)c_0 + tc_1$ , and a nonlinear remainder  $\tilde{c}(t)$ . This has the benefit that the same parametrization of  $\tilde{c}(t)$  can be used independently of the geodesic endpoints.

If we model  $\tilde{c}$  as a  $K^{\text{th}}$  order polynomial, i.e.

$$\tilde{c}_i(t) = \sum_{k=1}^K w_{ik} t^k + w_{i0}, \quad (8.7)$$

then it is trivial to ensure that  $\tilde{c}_i(0) = \tilde{c}_i(1) = 0$ . Specifically, letting  $w_{i0} = 0$  and  $w_{iK} = -\sum_{k=1}^{K-1} w_{ik}$  ensures these boundary conditions.

Computing geodesics, thus, reduces to minimizing the energy of the curve  $c$  with respect to  $w_{ik}$  for  $i = 1, \dots, d$  and  $k = 1, \dots, K-1$ .

### Step 2, take 1: energy quadrature

Given a curve  $c$ , we seek to numerically approximate its energy

$$\mathcal{E}[c] = \int_0^1 \|\dot{c}_t\|_{\mathbf{G}_{c_t}}^2 dt \approx \tilde{\mathcal{E}}[c]. \quad (8.8)$$

The simplest approach is to numerically approximate the integral with a Riemann sum (quadrature), i.e.

$$\tilde{\mathcal{E}}[c] = \sum_{s=0}^S \|\dot{c}_s\|_{\mathbf{G}_{c_s}}^2 \Delta_s, \quad (8.9)$$

where  $c_s = c(s/S)$  are placed on a regular grid over the interval  $[0, 1]$  and  $\Delta_s = 1/S + 1$ .

Depending on how we have chosen to parametrize the curve  $c$ , we may compute the curve derivative  $\dot{c}$  analytically or using finite differences.

On abstract manifolds, where we have direct access to the metric  $\mathbf{G}$ , we can trivially evaluate the squared curve velocity as  $\|\dot{c}_s\|_{\mathbf{G}_{c_s}}^2 =$

$\dot{c}_s^\top \mathbf{G}_{c_s} \dot{c}_s$ . When working with pull-back metrics  $\mathbf{G} = \mathbf{J}^\top \mathbf{J}$  it is, instead, often more practical to recall that the curve energy can be written as

$$\mathcal{E} = \int_0^1 \left\| \frac{d}{dt} f(c_t) \right\|^2 dt. \quad (8.10)$$

Making a finite difference approximation to the derivative  $d/dt f(c_t) \approx (f(c_{t+h}) - f(c_t))/h$  followed by the usual numerical quadrature, we can arrive at the simple energy approximation,

$$\tilde{\mathcal{E}}[c] = \sum_{s=1}^S \|f(c_s) - f(c_{s-1})\|^2. \quad (8.11)$$

Compared to working directly with the metric  $\mathbf{G} = \mathbf{J}^\top \mathbf{J}$ , this approach has the advantage that we invoke the automatic differentiation engine (assuming we are clever enough to use such) one less time, which can provide significant reductions in both running time, memory use, and code complexity.

### *Step 2, take 2: Monte Carlo energy*

The energy quadrature (8.11) requires differentiating through  $S + 1$  calls to the function  $f$ , which can be memory intensive when  $f$  has many parameters (e.g., it is a deep neural network). This can, in turn, render the gradient-based optimization (step 3) computationally taxing. In such cases, it can be beneficial to rewrite the curve energy (8.10) as an expectation,

$$\mathcal{E} = \int_0^1 \left\| \frac{d}{dt} f(c_t) \right\|^2 dt = \mathbb{E}_{p(t)} \left[ \left\| \frac{d}{dt} f(c_t) \right\|^2 \right], \quad (8.12)$$

where  $p(t) = \mathcal{U}(0, 1)$  is the uniform distribution over  $[0, 1]$ . We can then use a Monte Carlo estimate of this expectation to arrive at a stochastic approximation of the curve energy

$$\tilde{\mathcal{E}}[c] = \frac{1}{S} \sum_{s=1}^S \left\| \frac{d}{dt} f(c_{t_s}) \right\|^2, \quad t_s \sim \mathcal{U}(0, 1). \quad (8.13)$$

If need be, we can make a finite difference approximation to the curve derivative  $d/dt f(c_t) \approx (f(c_{t+h}) - f(c_t))/h$ .

*Step 3: numerical optimization*

Once we have a parametric curve  $c_{\mathbf{w}}$  with free parameters  $\mathbf{w}$ , and a numerical estimator  $\tilde{\mathcal{E}}$  of the curve energy, we can find the geodesic by solving

$$w_{\text{opt}} = \underset{\mathbf{w}}{\operatorname{argmin}} \tilde{\mathcal{E}}[c_{\mathbf{w}}]. \quad (8.14)$$

Assuming we work in a computational environment that supports automatic differentiation, then it is trivial to compute derivatives with respect to  $\mathbf{w}$ .

The energy functional is convex near its optima, which implies that any reasonable optimization algorithm will recover a locally optimal curve. The numerical approximation of the energy is, however, not guaranteed to share this local convexity, but in practice, this optimization is stable when the numerical integrator is reasonably accurate.

*8.4. Shooting geodesics*

From a starting point  $c_0$  and an initial velocity  $\dot{c}_0$  we can derive the associated geodesic  $c$  by solving the geodesic ODE (7.19). In practice, we can achieve the same by solving the ODE numerically using any standard solver (e.g., Euler methods, variants of Runge-Kutta, and so forth). We will not review such solvers here<sup>1</sup> and instead assume that they are available through some library implementation.

Standard numerical ODE solvers are aimed at first-order problems, i.e., problems of the type  $\dot{c}_t = h(t, c_t)$  for some function  $h$ . The geodesic ODE (7.19) is, however, a second-order problem

$$[\ddot{c}_t]_j = h(t, c_t, \dot{c}_t) = - \sum_{i=1}^d \sum_{k=1}^d \Gamma_j^{ik}(c_t) [\dot{c}_t]_k [\dot{c}_t]_i, \quad (8.15)$$

where  $\Gamma_j^{ik}$  are the ODE coefficients (7.20).

We can, fortunately, easily turn our second-order ODE into a first-order problem. To do this, we introduce a curve  $k : t \mapsto (c_t, \dot{c}_t)$  and

---

<sup>1</sup>Interested readers can consult standard textbooks [49].

write the geodesic ODE accordingly. Note that if  $c$  is a curve in  $\mathbb{R}^d$  then  $k$  is in  $\mathbb{R}^{2d}$ . By construction

$$\dot{k}_t = \begin{pmatrix} \dot{c}_t \\ \ddot{c}_t \end{pmatrix} = \begin{pmatrix} \dot{c}_t \\ h(t, c_t, \dot{c}_t) \end{pmatrix} \quad \begin{array}{l} \leftarrow \text{part of } k_t \\ \leftarrow \text{the geodesic ODE.} \end{array} \quad (8.16)$$

This gives rise to the following high-level algorithm.

*Alg.8.4: Compute shooting geodesic*

**Input:**  $c_0, \dot{c}_0$ .

1. Define the ODE  $\dot{k} = (t, k_t) \mapsto \begin{pmatrix} \dot{c}_t \\ h(t, c_t, \dot{c}_t) \end{pmatrix}$ .
2. Define the initial condition  $k_0 = \begin{pmatrix} c_0 \\ \dot{c}_0 \end{pmatrix}$ .
3. Solve the ODE  $k_t = \text{odeint}(\dot{k}, k_0, t)$ .
4. Split the solution into  $c_t, \dot{c}_t \leftarrow k_t$ .

Here `odeint` is any numerical first-order ODE solver that takes as input

1. The ODE to be solved. This should be a function that takes inputs  $t$  and  $k_t = (c_t, \dot{c}_t)$ .
2. The initial condition  $k_0 = (c_0, \dot{c}_0) \in \mathbb{R}^{2d}$ .
3. The points  $t_1, \dots, t_n$  where the solution curve should be evaluated. Usually, these are equidistant points from 0 to 1.<sup>2</sup>

Such numerical ODE solvers are commonly available across different computational environments.

## 8.5. Connecting geodesics, again



Modern ODE solvers support automatic differentiation, such that we can differentiate the solution with respect to the initial condition [12]. That is, we can evaluate  $\partial c_t / \partial \dot{c}_0$ .

<sup>2</sup>`linspace(0, 1, 25)` in several programming environments.

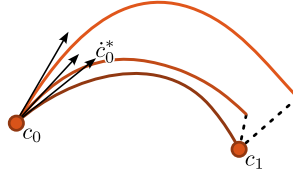


FIG. 8.4. The initial velocity of the geodesic ODE is optimized such that the resulting geodesic hits the right endpoint. The dashed lines indicate the optimization criteria.

Let  $c(t; \dot{c}_0)$  denote the numerical solution to the geodesic ODE starting at  $c_0$  with initial velocity  $\dot{c}_0$ . If we are seeking a geodesic with end-point  $c_1$ , then a solution is to solve

$$\dot{c}_0^* = \underset{\dot{c}_0}{\operatorname{argmin}} \|c_1 - c(1; \dot{c}_0)\|^2. \quad (8.17)$$

Using a numerical ODE solver that supports automatic differentiation, this equation can be immediately solved using standard gradient-based optimization. The solution will be an initial velocity such that the geodesic starting at  $c_0$  reaches  $c_1$ .

Conceptually this is quite elegant as both connecting and shooting geodesics are solved using the same driving building block: the ODE solver. In practice, the approach can, however, be both computationally expensive and numerically unstable. Expensive because the geodesic ODE requires derivatives of the metric, unlike direct energy minimization. Unstable, because differentiating an ODE requires solving another ODE, which often is ill-conditioned [12]. This approach for computing connecting geodesics is, thus, rarely recommendable.

## Exercises

### *Exercise 8.1: Curve parametrizations.*

Consider a two-dimensional abstract manifold with the metric

$$\mathbf{G}_{\mathbf{x}} = (1 + \|\mathbf{x}\|^2) \mathbf{I}, \quad \mathbf{x} \in \mathbb{R}^2. \quad (8.18)$$

1. Implement direct energy minimization for computing geodesics using piecewise straight lines to parameterize the solution curve.

2. *Extend the previous implementation to also support third-order polynomials to parametrize the solution curve.*

*Exercise 8.2: Shooting geodesics.*

Consider the same metric as the previous exercise.

1. *Implement an algorithm for computing shooting geodesics by solving the geodesic ODE using a first-order Euler method.*
2. *Extend the previous implementation to use more elaborate numerical ODE solvers. This can be done using a support library for your computational environment (e.g., `scipy` in Python).*

*Exercise 8.3: Density metrics.*

Consider the dataset available at

<http://hauberg.org/weekendwithbernie/toybanana.npy>

This consists of  $N = 992$  observations in  $\mathbb{R}^2$  (see Fig. 8.5). Consider the metric over  $\mathbb{R}^2$  defined as

$$\mathbf{G}_{\mathbf{x}} = \frac{1}{p(\mathbf{x}) + \epsilon} \quad (8.19)$$

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \mathcal{N}(\mathbf{x}|\mathbf{x}_n, \sigma^2 \mathbf{I}), \quad (8.20)$$

where  $\sigma = 0.1$ , and  $\epsilon = 10^{-4}$  avoids dividing by zero.

1. *Implement direct energy minimization for computing geodesics using piecewise straight lines to parameterize the solution curve.*

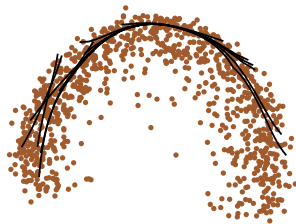


FIG. 8.5. The toy-banana dataset is a subset of 1-digits from MNIST expressed in its leading principal components and some example geodesics.

2. *Extend the previous implementation to also support third-order polynomials to parametrize the solution curve.*

*Exercise 8.4: Autoencoder geodesics.*

Consider a simple dataset (e.g., MNIST).

1. *Implement an autoencoder with at least one hidden layer in the decoder, and fit it to the chosen dataset.*
2. *Implement at least one algorithm for computing connecting geodesics.*
3. *How does the (empirical) computational cost of computing geodesics grow with the dimensionality of the latent representation?*



## Chapter 9

### *Arithmetic on Riemannian manifolds*

*In which we joyfully rediscover plus  
and minus.*

When building computational models over Riemannian manifolds we need basic operators akin to plus and minus for interacting with data. The most elementary operators over Riemannian manifolds are the logarithm and exponential maps, which exactly serve this purpose.

#### *9.1. Euclidean arithmetic*

We are used to interacting with vectors  $\mathbf{x}_n, \mathbf{x}_m \in \mathbb{R}^D$  by adding and subtracting them from each other. Before seeking to understand how we can interact with points on Riemannian manifolds, we first look at the Euclidean case. This will allow us to build up intuitions that we can generalize. The reader should be warned, that this section is beyond trivial, but it is so purposely.

##### *Shortest paths*

We first remind the reader that given two points in a Euclidean space, then the shortest connecting path is the straight line. This is an important analogy when considering the extension to Riemannian manifolds.

##### *Subtraction (minus)*

Consider two points,  $\mathbf{x}_n$  and  $\mathbf{x}_m$ , in a Euclidean space, their difference is given by  $\Delta = \mathbf{x}_n - \mathbf{x}_m$ . This is illustrated in Fig. 9.1 on the right.

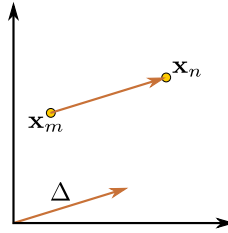


FIG. 9.1. *Subtraction in a Euclidean space is an operator that returns a vector representing the shortest path (straight line) connecting two points. Here,  $\mathbf{x}_n - \mathbf{x}_m = \Delta$ .*

The shortest path connecting  $\mathbf{x}_m$  to  $\mathbf{x}_n$  can be written as

$$\mathbf{c}(t) = \Delta t + \mathbf{x}_m, \quad t \in [0, 1]. \quad (9.1)$$

Here we have parametrized the shortest path over  $t \in [0, 1]$ , which is an arbitrary choice. We see that  $\Delta$  provides a representation of the shortest path from  $\mathbf{x}_m$  to  $\mathbf{x}_n$ , i.e., *subtraction is an operator that returns a vector that parametrizes the shortest path from one point to another.*

### Addition (plus)

Subtraction gives a vector representing the shortest path connecting two points, and addition is the inverse operation. Geometrically, this means that addition should be an operator that given a point  $\mathbf{x}_m$  and a vector  $\Delta$  should span the shortest path that connects the two points. Indeed, this is achieved by the addition in Eq. 9.1.

### Operators, points, and vectors

Thus far, we have made a subtle distinction between a *point*,  $\mathbf{x}_n$  or  $\mathbf{x}_m$ , and a *vector*  $\Delta$ . A point represents the *position* of an object, whereas a vector represents a *displacement*. We will now make this distinction painfully clear and let  $\mathcal{M}$  denote the set of all points, and let  $T$  denote the set of all displacements. In the Euclidean domain,  $\mathcal{M} = T = \mathbb{R}^D$  and this distinction are rather pointless. We will proceed nonetheless as it will soon carry value.

Now we can make the subtraction and addition operators more explicit by writing

$$- : \mathcal{M} \times \mathcal{M} \rightarrow T \quad (9.2)$$

$$+ : \mathcal{M} \times T \rightarrow \mathcal{M}. \quad (9.3)$$

That is, subtraction takes two points and returns a vector, while addition takes a point and a vector and returns a point.

### *Relation to distances*

As a final remark, we note that the vector  $\Delta = \mathbf{x}_n - \mathbf{x}_m$  not only tells us how to get from  $\mathbf{y}$  to  $\mathbf{x}$ , but also tells us how far  $\mathbf{x}_n$  is from  $\mathbf{x}_m$ . In particular, the distance between  $\mathbf{x}_n$  and  $\mathbf{x}_m$  is by definition  $\text{dist}(\mathbf{x}_n, \mathbf{x}_m) = \|\mathbf{x}_n - \mathbf{x}_m\| = \|\Delta\|$ .

## *9.2. Riemannian operators*

We now carry on to the Riemannian setting. Our goal will be to provide operators akin to subtraction and addition as described above.

### *Riemannian subtraction*

In Euclidean space, we have seen that subtraction is an operator that given two points returns a vector encoding the direction to get from one point to the other as fast as possible. The *logarithm map* serves the same purpose on Riemannian manifolds.

Given two points  $\mathbf{x}_n, \mathbf{x}_m \in \mathcal{M}$  on a Riemannian manifold, we let  $c_{mn}$  denote the shortest connecting path (geodesic). We have previously seen that this path can be recovered as the solution to a system of ordinary differential equations (7.19). The *Picard-Lindelöf theorem*<sup>1</sup> then imply that the geodesic is uniquely defined by its initial position and velocity. That is, the geodesic can be uniquely recovered from  $c_{mn}(0)$  and  $\partial_t c_{mn}(0)$  by solving the associated differential equations. Letting  $c_{mn}(0) = \mathbf{x}_m$  and  $\Delta = \partial_t c_{mn}(0)$ , we can interpret  $\Delta$  as the initial direction of the shortest path going from  $\mathbf{x}_m$  to  $\mathbf{x}_n$ . This vector then uniquely tells us how to get from  $\mathbf{x}_m$  to  $\mathbf{x}_n$  as fast as possible, and we can view  $\Delta$  as the Riemannian counterpart to the difference vector attained by Euclidean subtraction.

---

<sup>1</sup>See Appendix A if you are not familiar with this result.

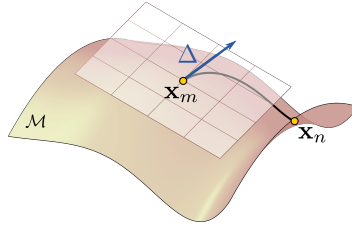


FIG. 9.2. The logarithm map (subtraction) returns the initial velocity of the connecting geodesic.

These considerations motivate the definition of the logarithm map (or *log-map* for short) as the function that given two points returns the initial velocity of the geodesic connecting the points. We write

$$\text{Log}_{\mathbf{x}_m}(\mathbf{x}_n) = \Delta = \partial_t c_{mn}(0), \quad (9.4)$$

and note that  $\Delta$  by definition belongs to the tangent space  $T_{\mathbf{x}_m}$ . If the geodesic is parametrized over the time interval  $[0, 1]$  with constant speed, then we see that its length is

$$\text{Length}_{\mathcal{M}}(c_{mn}) = \int_0^1 \|\partial_t c_{mn}(t)\| dt \quad (9.5)$$

$$= \|\partial_t c_{mn}(0)\| \int_0^1 dt \quad (9.6)$$

$$= \|\Delta\|. \quad (9.7)$$

That is, the norm of the vector returned by the logarithm map is the geodesic distance between  $\mathbf{x}_n$  and  $\mathbf{x}_m$ , and we have recovered the two characteristics of subtraction highlighted in the previous section. It should be noted that since  $\Delta$  is a member of the tangent space  $T_{\mathbf{x}_m}$ , then the norm of  $\Delta$  should be computed using the inner product (metric) associated with this tangent space.

### *Riemannian addition*

Having seen that the logarithm map performs an operation akin to subtraction, it is natural to consider the inverse operation. This is known as the *exponential map* and is defined as the solution to the geodesic ODE (7.19) subject to the initial conditions

$$c(0) = \mathbf{x}_m \quad (9.8)$$

$$\partial_t c(0) = \Delta. \quad (9.9)$$

This ODE is evaluated at time  $t = 1$  to match the parametrization used for the logarithm map. We write this as

$$\text{Exp}_{\mathbf{x}_m}(\Delta) = \mathbf{x}_n. \quad (9.10)$$

## Exercises

### Exercise 9.1: Spherical arithmetic.

On the unit sphere,  $\mathbb{S}^{D-1} = \{\mathbf{y} \in \mathbb{R}^D \mid \|\mathbf{y}\| = 1\}$ , two points  $\mathbf{y}_1$  and  $\mathbf{y}_2$  can be connected with the geodesic

$$c(t) = \cos\left(\frac{2t}{\pi}\right) \mathbf{y}_1 + \sin\left(\frac{2t}{\pi}\right) \mathbf{y}_2, \quad (9.11)$$

assuming the points are not on opposite poles. The geodesic distance between the points is the angle between the vectors  $\mathbf{y}_1$  and  $\mathbf{y}_2$ ,

$$\text{dist}(\mathbf{y}_1, \mathbf{y}_2) = \arccos(\mathbf{y}_1^\top \mathbf{y}_2). \quad (9.12)$$

1. Derive the expression for the logarithm map  $\text{Log}_{\mathbf{y}_1}(\mathbf{y}_2)$ .
2. Derive the expression for the exponential map  $\text{Exp}_{\mathbf{y}_1}(\mathbf{v})$  for some vector  $\mathbf{v} \in \mathbb{R}^D$  such that  $\mathbf{v}^\top \mathbf{y}_1 = 0$ .

### Exercise 9.2: Density metrics.

You will now extend the density metric from exercise 8.3 to support exponential and logarithm maps.

1. Implement a computer program that evaluates  $\text{Log}_{\mathbf{y}_1}(\mathbf{y}_2)$ .
2. Implement a computer program that evaluates  $\text{Exp}_{\mathbf{y}_1}(\mathbf{v})$ .

### Exercise 9.3: Autoencoders.

You will now extend the autoencoder from exercise 8.4 to support exponential and logarithm maps.

1. Implement a computer program that evaluates  $\text{Log}_{\mathbf{y}_1}(\mathbf{y}_2)$ .
2. Implement a computer program that evaluates  $\text{Exp}_{\mathbf{y}_1}(\mathbf{v})$ .

## Chapter 10

### *Integration on manifolds*

*In which the metric speaks volumes  
about space.*

The volume of an object is one of the most basic quantities we may compute. Of particular importance, volumes are useful for determining the probability of an event, which is paramount for any decision-making procedure.

To define a notion of the volume of a part of a manifold, we need *integration* over the said manifold. Consider, as usual, a manifold  $\mathcal{M} = f(\Omega)$ , where  $f : \mathbb{R}^d \rightarrow \mathbb{R}^D$  and  $\Omega \subset \mathbb{R}^d$ . If  $\Omega$  is compact then the manifold has a finite volume defined as

$$\text{vol}(\mathcal{M}) = \int_{\mathcal{M}} 1 \, dy. \tag{10.1}$$

In practice (and in particular when working numerically), it is much easier to compute this integral if we can phrase this over the low-dimensional  $\Omega$  rather than the high-dimensional  $\mathcal{M}$ . This is what we will set out to do.

#### *10.1. Ordinary integration*

Before extending our view of integration, let us briefly recap the basics of the *Riemann integral*.<sup>1</sup> Given a real function  $h : \mathbb{R} \rightarrow \mathbb{R}$ , the integral of  $h$  corresponds to the area under the curve traced out by  $h$ . The *Riemann sum* provides an approximation to this area by partitioning

---

<sup>1</sup>While the name might suggest that this integral is tied to Riemannian geometry, such is not the case. The name merely shows that Bernhard Riemann was a productive fellow.

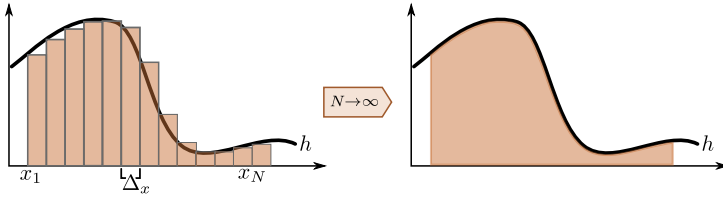


FIG. 10.1. The Riemann sum approximates the area under the curve by summing the volume of small segments that fit the curve. Note how the segment width  $\Delta_x$  depends on the first axis being the real line  $\mathbb{R}$ .

the first axis into segments of size  $\Delta_x$  and then summing the areas of the corresponding boxes that fit under the curve,

$$\text{Area} \approx \sum_{n=1}^N h(x_n) \Delta_x, \quad \text{where } x_1 < \dots < x_N. \quad (10.2)$$

Here  $\Delta_x = x_{n+1} - x_n$  is the width of each segment. We can then get an increasingly better approximation by increasing  $N$ , and the Riemann integral is the limit  $N \rightarrow \infty$ ,

$$\int_a^b h(x) dx = \lim_{N \rightarrow \infty} \sum_{n=1}^N h(x_n) \Delta_x. \quad (10.3)$$

The notation  $dx$  should be thought of as the limiting value of  $\Delta_x$ , i.e., it is the width of an infinitesimal segment.

## 10.2. Volume measures

Let us consider the more general task of integrating a function over a subset of a manifold. That is, we are interested in a function  $h : \mathcal{M} \rightarrow \mathbb{R}$ , which raises the question of how we are going to determine the width of the infinitesimal segment in the Riemann sum.

Let us assume that  $\mathcal{X} \subseteq \Omega$  is the region over which we want to integrate, and that  $h : \mathcal{M} \rightarrow \mathbb{R}$  is the *integrand*, i.e., the function which we seek to integrate. Then we can define the integral of interest as

$$\int_{f(\mathcal{X})} h(\mathbf{y}) d\mathbf{y}. \quad (10.4)$$

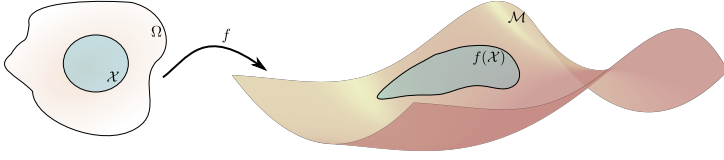


FIG. 10.2. When integrating a function over a subset of the manifold,  $f(\mathcal{X})$ , it is generally more practical to express it as an integral over  $\mathcal{X}$  instead.

By standard *integration by substitution* we may write this as

$$\int_{\mathcal{X}} h(f(\mathbf{x}))V(\mathbf{x})d\mathbf{x}, \quad (10.5)$$

where  $V(\mathbf{x})$  is the volume of the parallelogram spanned by the Jacobian of  $f$  at  $\mathbf{x}$ . This volume informs us of how different the volume of a small patch in  $\Omega$  is compared to the corresponding patch expressed on the manifold  $\mathcal{M}$ .

To quantify this change in volume, we write the Jacobian of  $f$  at  $\mathbf{x}$  in its *singular value decomposition*

$$\mathbf{J}_{\mathbf{x}} = \mathbf{U}_{\mathbf{x}}\mathbf{S}_{\mathbf{x}}\mathbf{V}_{\mathbf{x}}^{\top}, \quad (10.6)$$

where  $\mathbf{U}_{\mathbf{x}} \in \mathbb{R}^{D \times D}$  and  $\mathbf{V}_{\mathbf{x}} \in \mathbb{R}^{d \times d}$  are orthogonal matrices, and  $\mathbf{S}_{\mathbf{x}} \in \mathbb{R}^{D \times d}$  is a rectangular diagonal matrix with strictly positive entries.<sup>2</sup> Since orthogonal matrices (rotation matrices) do not influence the volume of the parallelogram spanned by the Jacobian, we observe

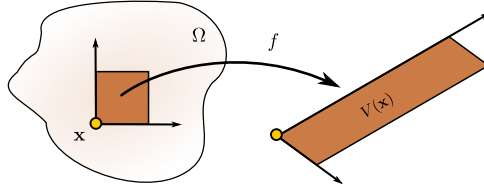


FIG. 10.3. The volume of an infinitesimal patch in  $\Omega$  changes when measured in ambient space.

<sup>2</sup>As the Jacobian is assumed to have full rank.



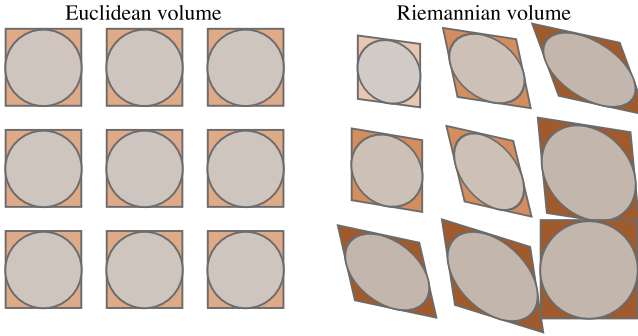


FIG. 10.4. *Integration is defined by partitioning space into small segments and the volume of each segment weighs its contribution to the total integral. In Euclidean space, this volume is constant, but for Riemannian models, we must take the ‘size’ of the local metric into account.*

that this must be the product of the diagonal of  $\mathbf{S}_x$ ,

$$V(\mathbf{x}) = \prod_{i=1}^d [\mathbf{S}_x]_{ii}. \quad (10.7)$$

Returning to the singular value decomposition of  $\mathbf{J}_x$  (10.6), we observe that the metric can be written

$$\mathbf{G}_x = \mathbf{J}_x^T \mathbf{J}_x = \mathbf{V}_x \mathbf{S}_x^T \mathbf{S}_x \mathbf{V}_x^T. \quad (10.8)$$

This is the eigendecomposition of  $\mathbf{G}_x$  where  $\mathbf{S}_x^T \mathbf{S}_x$  is a square diagonal matrix containing the eigenvalues of the metric. The determinant of  $\mathbf{G}_x$  is the product of these eigenvalues, so the metric determinant must be the square of the volume of the parallelogram spanned by  $\mathbf{J}_x$ ,

$$V^2(\mathbf{x}) = \det(\mathbf{G}_x) \Leftrightarrow V(\mathbf{x}) = \sqrt{\det(\mathbf{G}_x)}. \quad (10.9)$$

Combining these observations, we may write the expression of the integral (10.5) as

$$\int_{\mathcal{X}} h(f(\mathbf{x})) \sqrt{\det(\mathbf{G}_x)} d\mathbf{x}. \quad (10.10)$$

One common interpretation is that  $\sqrt{\det(\mathbf{G}_{\mathbf{x}})}d\mathbf{x}$  forms a *measure* over the manifold, which gives rise to the short-hand notation

$$\int_{\mathcal{X}} h(f(\mathbf{x}))\sqrt{\det(\mathbf{G}_{\mathbf{x}})}d\mathbf{x} = \int_{\mathcal{X}} h(f(\mathbf{x}))d\mathcal{M}(\mathbf{x}). \quad (10.11)$$

This nicely highlights that the metric not only tells us how to compute inner products but also how to integrate over the manifold. The metric truly provides an elegant summary of the quantitative aspects of the manifold.

### 10.3. Densities and probabilities

Having established how to integrate over a manifold, it is now easy to define probabilities over manifold-valued events. If  $\mathbf{x} \in \Omega$  is a random variable, we may give it a density  $p(\mathbf{x}) \geq 0$  under the requirement that it integrates to 1,

$$\int_{\Omega} p(\mathbf{x})d\mathcal{M}(\mathbf{x}) = 1. \quad (10.12)$$

The probability of an event is then defined as usual

$$\Pr(\mathbf{x} \in \mathcal{X}) = \int_{\mathcal{X}} p(\mathbf{x})d\mathcal{M}(\mathbf{x}). \quad (10.13)$$

An example density is a *uniform density* over a compact region  $\mathcal{X}$ , which is constant with respect to the measure  $d\mathcal{M}$ , i.e.

$$\mathcal{U}_{\mathcal{X}}(\mathbf{x}) = \begin{cases} \frac{1}{\int_{\mathcal{X}} d\mathcal{M}(\mathbf{x})} & \mathbf{x} \in \mathcal{X} \\ 0 & \text{otherwise.} \end{cases} \quad (10.14)$$

Something akin to a *normal distribution* can also be defined as

$$p(\mathbf{x}) = \frac{1}{Z(\mu, \sigma)} \exp\left(-\frac{\text{dist}^2(\mathbf{x}, \mu)}{2\sigma^2}\right), \quad (10.15)$$

where  $\mu$  and  $\sigma$  parametrize the distribution. The normalization constant  $Z(\mu, \sigma)$  is generally not known in closed form.<sup>3</sup> This is a common

---

<sup>3</sup>For an exception on the sphere, see [20].

issue for distributions on manifolds as integration almost always requires numerical approximations, such as *Monte Carlo* techniques.

A common workaround is to place Euclidean distributions in the tangent space of some a priori chosen point. This distribution can then be mapped to the manifold through the exponential map (Chapter 9). This leads to the so-called *wrapped distributions*. While they are fairly easy to apply, their practical behavior crucially depends on the chosen point of tangency, and it is common to observe notable distortion when mapping onto the manifold [23].

## 10.4. Expectations

If we have a random variable  $\mathbf{x} \in \Omega$  with density  $p(\mathbf{x})$ , and a function  $h : \Omega \rightarrow \mathbb{R}$  then we can define the expectation of  $h(\mathbf{x})$  as usual

$$\mathbb{E}[h(\mathbf{x})] = \int_{\Omega} h(\mathbf{x})p(\mathbf{x})d\mathcal{M}(\mathbf{x}). \quad (10.16)$$

For finite data, this is then approximated as

$$\mathbb{E}[h(\mathbf{x})] \approx \frac{1}{N} \sum_{n=1}^N h(\mathbf{x}_n)d\mathcal{M}(\mathbf{x}_n). \quad (10.17)$$

An annoying subtlety, however, prevents us from using this construction to define something as elementary as the *mean* of a random variable. If we just naively attempt to evaluate  $\mathbb{E}[\mathbf{x}]$ , then we end up with an integral over a manifold-valued variable rather than a real variable. This is undefined, and we have no standard extension of means to manifolds.

As we often find great value in summarizing an entire data set in a single observation, various generalizations of means are available. The most common choice is the *Fréchet mean*, which is defined as follows. We start by defining the *variance* with respect to a point  $\mu$  as the average squared distance to data,

$$\text{Var}[\mathbf{x}] = \mathbb{E}[\text{dist}^2(\mathbf{x}, \mu)]. \quad (10.18)$$

In Euclidean statistics, it is well-known that the mean is the unique minimizer of  $\text{Var}[\mathbf{x}]$ . The Fréchet mean for manifold-valued data is then

defined as the global minimizer of the variance. Note that multiple minima may exist, so this definition is not as well-behaved as its Euclidean counterpart. In practice, we tend to minimize the empirical variance using gradient-based optimization and then hope that the point of convergence is a reasonable summary statistic.

Finally, it should be noted that the Fréchet mean is generally not a maximum likelihood estimator. For example, the mean of the normal distribution in Eq. 10.15 influence the distribution's normalization constant. This implies that a simple variance minimization does not maximize likelihood – unlike in Euclidean settings.

## Exercises

### *Exercise 10.1: Spherical metric.*

Let  $x_1 \in (0, \pi)$  and  $x_2 \in (0, 2\pi)$  be the latent coordinates of the manifold spanned by the function  $f : (0, \pi) \times (0, 2\pi) \rightarrow \mathbb{R}^3$

$$f(x_1, x_2) = \begin{pmatrix} \sin(x_1) \cos(x_2) \\ \sin(x_1) \sin(x_2) \\ \cos(x_1) \end{pmatrix}. \quad (10.19)$$

This will span the unit sphere. In exercise 5.1, we derived that the metric in latent coordinates is

$$\mathbf{G}_{\mathbf{x}} = \begin{pmatrix} 1 & 0 \\ 0 & \sin^2(x_1) \end{pmatrix}. \quad (10.20)$$

1. *Derive the volume measure of the metric.*
2. *Integrate the function  $h(x_1, x_2) = x_2 \sin(x_1)$  with respect to the volume measure, i.e., compute  $\int_0^\pi \int_0^{2\pi} h(x_1, x_2) d\mathcal{M}(x_1, x_2)$ .*
3. *Write a computer program to numerically estimate the value of the integral.*

### *Exercise 10.2: Quadratic metric.*

Consider a two-dimensional abstract manifold with the metric

$$\mathbf{G}_{\mathbf{x}} = (1 + \|\mathbf{x}\|^2) \mathbf{I}, \quad \mathbf{x} \in \mathbb{R}^2. \quad (10.27)$$

1. Derive the volume measure of the metric.
2. Integrate the function  $h(x_1, x_2) = x_1$  with respect to the volume measure over the set  $[-1, 1]^2$ , i.e., compute

$$\int_{-1}^1 \int_{-1}^1 h(x_1, x_2) d\mathcal{M}(x_1, x_2)$$

3. Write a computer program to numerically estimate the value of the integral.

*Exercise 10.3: Normalizing flows.*

Consider a smooth embedding<sup>4</sup>  $f : \mathbb{R}^d \rightarrow \mathbb{R}^D$ , and let the random variable  $\mathbf{y}$  be distributed as  $p(\mathbf{y})$ . Assume that  $p(\mathbf{y})$  has support on the entirety of  $\mathbb{R}^D$ , i.e.,  $p(\mathbf{y}) > 0 \forall \mathbf{y}$ .

1. Let  $\mathbf{x} = f^{-1}(\mathbf{y})$  be a new random variable. What is the support of  $p(\mathbf{x})$ ?
2. Bringing the Euclidean metric from  $\mathbb{R}^D$  into  $\mathbb{R}^d$  gives the Riemannian metric  $\mathbf{G}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^T \mathbf{J}_{\mathbf{x}}$ . Use Eq. 10.10 to evaluate the probability  $\Pr(\mathbf{x} \in \hat{\mathcal{X}}) = \int_{f(\hat{\mathcal{X}})} p(\mathbf{y}) d\mathbf{y}$ .
3. Consider the case  $d = D$ . Show that the volume measure of the metric reduces to  $\sqrt{\det \mathbf{G}_{\mathbf{x}}} = \det \mathbf{J}_{\mathbf{x}}$ . Update the expression of  $\Pr(\mathbf{x} \in \hat{\mathcal{X}})$  to this case.

---

<sup>4</sup>If you forgot about embeddings, see page 26.

# Chapter 11

## Submanifolds



*In which recursion takes over  
(again).*

So far, we have constructed distance measures over manifolds by measuring curve lengths along the manifold. The length of such curves has then been measured using the Euclidean metric of the ambient space. On the other hand, we have seen how we can disregard the ambient space altogether once we have access to a Riemannian metric. This raises the natural question: *can our ambient space have a Riemannian metric?*

### 11.1. Composite functions

To start the discussion, let us first recall that we focus on manifolds that are spanned by a function  $f : \mathbb{R}^d \rightarrow \mathbb{R}^D$ ,

$$\mathcal{M} = f(\Omega), \quad \Omega \subseteq \mathbb{R}^d. \quad (11.1)$$

Let us now assume that  $f$  is a composite function, i.e.

$$f(\mathbf{x}) = f_2(f_1(\mathbf{x})), \quad (11.2)$$

where  $f_1 : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ ,  $f_2 : \mathbb{R}^{d'} \rightarrow \mathbb{R}^D$ , and  $d \leq d' \leq D$ .<sup>1</sup> The analysis we have carried out thus far tells us that then  $\Omega \subseteq \mathbb{R}^d$  is naturally equipped with a Riemannian metric  $\mathbf{G}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^T \mathbf{J}_{\mathbf{x}}$ . The same analysis, however, also tells us that  $\Omega' = f_1(\Omega) \subseteq \mathbb{R}^{d'}$  also has a Riemannian metric,

$$\mathbf{G}_{\mathbf{x}}^{(2)} = \mathbf{J}_{\mathbf{x}}^{(2)T} \mathbf{J}_{\mathbf{x}}^{(2)}, \quad (11.3)$$

---

<sup>1</sup>In a machine learning context, the obvious example of such a function is a multi-layer neural network.

where  $\mathbf{J}_{\mathbf{x}}^{(2)} = \partial f_2 / \partial f_1(\mathbf{x})$  is the Jacobian of  $f_2$ . In this case,  $\mathbf{G}_{\mathbf{x}}$  and  $\mathbf{G}_{\mathbf{x}}^{(2)}$  represent the Euclidean metric of  $\mathbb{R}^D$  brought into  $\mathbb{R}^d$  and  $\mathbb{R}^{d'}$ , respectively. A natural question is then if we can bring the metric  $\mathbf{G}_{\mathbf{x}}^{(2)}$  from  $\mathbb{R}^{d'}$  into  $\mathbb{R}^d$  and if doing so results in the metric  $\mathbf{G}_{\mathbf{x}}$ ?

To provide an answer, let us consider the manifold

$$\mathcal{M}_1 = f_1(\Omega) \subset \mathbb{R}^{d'}. \quad (11.4)$$

We will use the Riemannian metric  $\mathbf{G}_{\mathbf{x}}^{(2)}$  to measure curve lengths in  $\mathbb{R}^{d'}$ , i.e.

$$\text{Length}(c) = \int_0^1 \left\| \frac{d}{dt} f_1(c_t) \right\|_{\mathbf{G}_{c_t}^{(2)}} dt \quad (11.5)$$

$$= \int_0^1 \left\| \mathbf{J}_{c_t}^{(1)} \dot{c}_t \right\|_{\mathbf{G}_{c_t}^{(2)}} dt, \quad (11.6)$$

where  $\mathbf{J}_{\mathbf{x}}^{(1)} = \partial f_1 / \partial \mathbf{x}$  is the Jacobian of  $f_1$ . Inserting the definition of the norm (5.15) then gives

$$\text{Length}(c) = \int_0^1 \sqrt{\left( \mathbf{J}_{c_t}^{(1)} \dot{c}_t \right)^\top \mathbf{G}_{c_t}^{(2)} \left( \mathbf{J}_{c_t}^{(1)} \dot{c}_t \right)} dt \quad (11.7)$$

$$= \int_0^1 \sqrt{\dot{c}_t^\top \left( \mathbf{J}_{c_t}^{(1)\top} \mathbf{G}_{c_t}^{(2)} \mathbf{J}_{c_t}^{(1)} \right) \dot{c}_t} dt. \quad (11.8)$$

That is,  $\Omega$  is now equipped with the metric  $\mathbf{J}_{c_t}^{(1)\top} \mathbf{G}_{c_t}^{(2)} \mathbf{J}_{c_t}^{(1)}$ .

The Jacobian of  $f$  can be written in terms of the Jacobians of  $f_1$  and  $f_2$  as  $\mathbf{J}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^{(2)} \mathbf{J}_{\mathbf{x}}^{(1)}$ , such that our initial metric for  $\Omega$  is

$$\mathbf{G}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^\top \mathbf{J}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^{(1)\top} \mathbf{J}_{\mathbf{x}}^{(2)\top} \mathbf{J}_{\mathbf{x}}^{(2)} \mathbf{J}_{\mathbf{x}}^{(1)} \quad (11.9)$$

$$= \mathbf{J}_{c_t}^{(1)\top} \mathbf{G}_{c_t}^{(2)} \mathbf{J}_{c_t}^{(1)}, \quad (11.10)$$

which is the same as above. The unsurprising summary is that if  $f$  is a composite function, then we can bring the ambient metric through the individual functions to create intermediate metrics, and still get the original metric  $\mathbf{G}_{\mathbf{x}}$ .

## 11.2. Ambient metrics and submanifolds

Sometimes observed data naturally comes with a Riemannian metric rather than the Euclidean metric. We can then follow the same procedure as in the previous section to bring this Riemannian metric into our low-dimensional representation  $\Omega$ .

More precisely, let  $\mathcal{M}^{(D)}$  denote  $\mathbb{R}^D$  equipped with the Riemannian metric  $\mathbf{G}^{(D)}$ , and let

$$f : \Omega \rightarrow \mathcal{M}^{(D)}, \quad \Omega \subseteq \mathbb{R}^d \quad (11.11)$$

span a manifold  $\mathcal{M}^{(d)} = f(\Omega) \subseteq \mathcal{M}^{(D)}$ . We will then call  $\mathcal{M}^{(d)}$  a *submanifold* of  $\mathcal{M}^{(D)}$ . As a simple example, a curve on a manifold is a one-dimensional submanifold of the said manifold.

We can then bring the metric from  $\mathcal{M}^{(D)}$  into  $\Omega$  as

$$\mathbf{G}_{\mathbf{x}}^{(d)} = \mathbf{J}_{\mathbf{x}}^T \mathbf{G}_{\mathbf{x}}^{(D)} \mathbf{J}_{\mathbf{x}}, \quad (11.12)$$

where  $\mathbf{J}_{\mathbf{x}}$  is the Jacobian of  $f$ . Note that if  $\mathbf{G}^{(D)} = \mathbf{I}$ , then  $\mathcal{M}^{(D)}$  is just the ordinary Euclidean  $\mathbb{R}^D$  and we recover the metric used in the previous chapters.

## Exercises

### Exercise 11.1: Jacobians and metrics.

Consider the functions

$$f_1(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \quad f_2(\mathbf{x}) = \begin{pmatrix} \tanh([\mathbf{x}]_1) \\ \vdots \\ \tanh([\mathbf{x}]_D) \end{pmatrix}, \quad (11.13)$$

where  $\mathbf{W} \in \mathbb{R}^{D \times d}$  and  $\mathbf{b} \in \mathbb{R}^D$ .

1. Derive the Jacobians of  $f_1$ ,  $f_2$  and  $f_3 = f_2 \circ f_1$ .
2. Derive the pull-back metrics  $\mathbf{G}_1 = \mathbf{J}_{f_1}^T \mathbf{J}_{f_1}$ ,  $\mathbf{G}_2 = \mathbf{J}_{f_2}^T \mathbf{J}_{f_2}$ , and  $\mathbf{G}_3 = \mathbf{J}_{f_3}^T \mathbf{J}_{f_3}$ , and write  $\mathbf{G}_{f_3}$  in terms of  $\mathbf{G}_{f_2}$ .



*Exercise 11.2: Autoencoders.*

Consider the autoencoder from exercise 8.4.

1. *Compute the pull-back metric associated with any layer of the encoder–decoder pipeline. How are they related to each other?*
2. *Put a density metric (6.2) over the observation space (see exercise 8.3). How does this change the metric in the latent space?*

## Chapter 12

### *Basics of topology*



*In which we forget about appearances  
and go for inner qualities.*

So far, we haven't actually defined what is really meant by the word *manifold*. In the machine learning literature, the word manifold is loosely used to indicate a smoothly connected set of points that intrinsically has a lower dimension than the space in which they exist. In this chapter, we formalize the term but emphasize that for most purposes the informal view is to be preferred. Consequently, this chapter can also be skipped by the hurried reader.

#### *12.1. Topological spaces and manifolds*

*Topology* studies properties of abstract spaces that are preserved even when we bend or twist the space. Usually, we are concerned with properties such as *connectivity* (which points are connected) or *holes* (an object that prevents its neighborhood from shrinking into a single point). To formalize these notions, we need to define two function classes that are useful when characterizing the “bendings and twistings” of a space.

We start the discussion with the notion of a *topological space*, which is the most elementary space with which we will be concerned.

**Definition 1.** A topological space is a set of points  $\mathcal{S}$  together with a collection of open subsets  $\tau$  that satisfy

1. The empty set  $\emptyset$  is in  $\tau$ .
2.  $\mathcal{S}$  is in  $\tau$ .
3. The intersection of a finite number of sets in  $\tau$  is also in  $\tau$ .
4. The union of a finite number of sets in  $\tau$  is also in  $\tau$ .

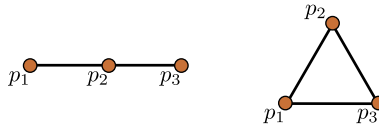


FIG. 12.1. The same points form different connectivity graphs under different topologies (see example 2).

We call  $\tau$  the topology of the set  $\mathcal{S}$ .

This is a rather abstract concept and is (for our purposes) best viewed as such. We will, however, consider two examples to show that the most common sets we encounter in machine learning are indeed topological spaces.

**Example 1** (Euclidean topology). Let  $\mathcal{S}$  be the  $D$ -dimensional Euclidean space  $\mathbb{R}^D$ , and let the topology  $\tau$  be the collection of open balls of  $\mathbb{R}^D$ . Clearly, this topology satisfies definition 1, such that  $\mathbb{R}^D$  is a topological space.

**Example 2** (Point topology). Consider 3 points  $\mathcal{S} = \{p_1, p_2, p_3\}$  with any specific coordinate representation. Let the topology of this set be  $\tau_1 = \{\emptyset, \mathcal{S}, \{p_1, p_2\}, \{p_2, p_3\}, \{p_2\}\}$ . This is easily verified to satisfy definition 1 such that  $(\mathcal{S}, \tau_1)$  is a topological space. We can interpret this topology as defining connectivity by saying that two points  $p_i, p_j$  are connected if they both appear in all subsets containing either. For example, the point  $p_1$  appears in the subsets  $\{p_1, p_2\}$ ,  $\{p_1, p_2, p_3\}$ , and  $p_2$  appear in both, so we say that  $p_1$  and  $p_2$  are connected. From this, we can construct the left graph of Fig. 12.1. If we instead consider the trivial topology  $\tau_2 = \{\emptyset, \mathcal{S}\}$ , then all points are connected to each other (right graph of Fig. 12.1).

The last example shows how we can use the topology of a set of points to encode properties, such as connectivity, without giving the points actual coordinates. This allows us to study the structure of a space without embedding it into an ambient space to observe its shape.

One way to understand the structure of a space is to compare it with another space that is already well-understood. For example, we might ask if a newly encountered space has a topology that is similar to that of the sphere. We often answer this question by deforming one

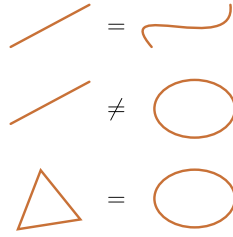


FIG. 12.2. *Seemingly different spaces can be topologically equal. The figure shows that a line segment is topologically identical to a curve, but not a circle. A triangle, on the other hand, is identical to a circle.*

space continuously to become another. To formalize this, we need a well-behaved class of functions for carrying out this deformation.

**Definition 2.** A homeomorphism is a continuous function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  between two topological spaces, which is bijective and has a continuous inverse  $f^{-1}$ .

We then say that two spaces are topologically identical if they can be related through a homeomorphism; Fig. 12.2 provides a few examples. In the top row, we can bijectively deform the straight line to a curve, but (center row) we cannot deform to a circle, because when the curve end-points meet to close the circle, we lose the ability to invert the deformation at this point. In practice, homeomorphisms can be implemented with neural network architectures from *normalizing flows* or *diffusion models* [27, 29, 12].

Of particular interest to us are topological spaces that locally are Euclidean. Such a space is called a *manifold*.

**Definition 3.** A manifold  $\mathcal{M}$  is a topological space  $(\mathcal{M}, \tau)$  if there is a non-negative integer  $d$  such that every point in  $\mathcal{M}$  has a neighborhood that is homeomorphic to an open subset of  $\mathbb{R}^d$  (usually a  $d$ -ball). In this case,  $\mathcal{M}$  is said to have dimension  $d$ .

Manifolds are rather convenient since their local Euclidean nature implies that we can, in principle, locally assign coordinates to points on the manifold. Given a local neighborhood around a point, we can assign coordinates to this neighborhood using a *chart*.

**Definition 4.** A chart  $\{\phi, U\}$  consist of a homeomorphism  $\phi : U \subset \mathcal{M} \rightarrow \phi(U) \subset \mathbb{R}^d$  acting on an open set  $U \subset \mathcal{M}$ . Given  $\mathbf{x} \in U$ , the vector  $\phi(\mathbf{x})$  is said to be coordinates of  $\mathbf{x}$ .

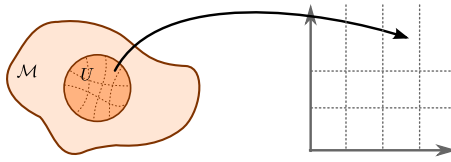


FIG. 12.3. A chart is a function  $\phi$  that assigns Euclidean coordinates to points in a local neighborhood  $U$  on the manifold  $\mathcal{M}$ .

An *atlas* is then a collection of charts, such that any point on  $\mathcal{M}$  can be assigned coordinates. Terminology is borrowed from cartography, which seeks to draw maps of the world. Each page of a book of maps (an *atlas*) is a flat (Euclidean) picture of a region of the world. Like different pages may contain overlapping regions, different charts may also assign different coordinates to the same point on a manifold.

## 12.2. Smooth manifolds

In machine learning, we often need to optimize a loss function in order to fit a model to data. For such purposes, it is practical if the spaces in which we operate are smooth, such that we can compute gradients. Manifolds need not be smooth, which motivates us to restrict the class of spaces under consideration to those that are smooth. The strategy is simple: where a manifold is only locally homeomorphic to a Euclidean space, we define smooth manifolds as those that are smoothly related to a Euclidean space (compare with definition 3).

**Definition 5.** A smooth manifold  $\mathcal{M}$  is a topological space  $(\mathcal{M}, \tau)$  if there is a non-negative integer  $d$  such that every point in  $\mathcal{M}$  has a neighborhood that is related to an open subset of  $\mathbb{R}^d$  (usually a  $d$ -ball) through a smooth homeomorphism. In this case,  $\mathcal{M}$  is said to have dimension  $d$ .

It directly follows that charts on a smooth manifold are smooth functions. This allows us to compute derivatives of functions defined on a smooth manifold. Let  $f_a : U_a \rightarrow \mathbb{R}$  be a collection of smooth functions for each subset  $U_a \subset \mathcal{M}$ . Then we can compute derivatives, as usual, within the coordinates of each subset  $\phi_a(U_a)$ , and between coordinates by differentiating through  $\phi_b \circ \phi_a^{-1}$ , which is possible as

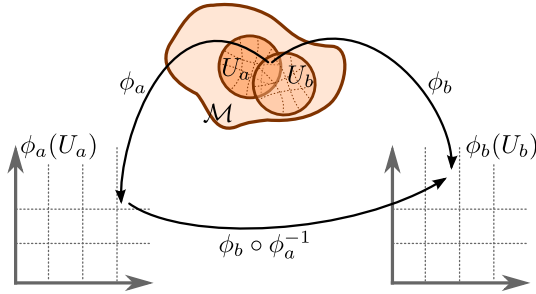


FIG. 12.4. When the charts are smooth, we can compute derivatives across local coordinates.

the individual functions are smooth. This concept of mapping between coordinate charts is shown in Fig. 12.4. In practice, we rarely need to carry out these calculations and therefore skip detailed examples.

### 12.3. Riemannian manifolds

A *Riemannian manifold* is then simply a smooth manifold which is equipped with a Riemannian metric. That is, a smooth manifold is a way to describe a set, while a Riemannian manifold is such a set along with the additional ability to quantify stuff, e.g., distances, angles, and volumes.

When the target is to analyze data, we almost always want some notion of a metric. In manifold learning, we often take this metric from the observation space since we understand this choice, and (honestly) it is many times the only choice we can think of.

## Exercises

### Exercise 12.1: Multi-charted autoencoders.

To illustrate the concept of a chart, we will consider a simple multi-

charted autoencoder to be trained on the MNIST dataset. Let

$$f(\mathbf{x}) = \sum_{k=1}^K r_k(\mathbf{x}) f_k(\mathbf{x}) \quad (12.1)$$

be a manifold consisting of  $K$  decoders  $f_k : \mathbb{R}^d \rightarrow \mathbb{R}^D$  that are weighted by  $r_k : \mathbb{R}^d \rightarrow [0, 1]$ . Further impose that

$$\sum_{k=1}^K r_k(\mathbf{x}) = 1 \quad \forall \mathbf{x}. \quad (12.2)$$

We can interpret the  $K$  decoders as being different charts, which we then smoothly combine to ensure that the resulting model is a smooth manifold.

1. *First, we will train such a model on MNIST. Let  $g(\mathbf{y}) : \mathbb{R}^D \rightarrow \mathbb{R}^d \times [0, 1]^K$  be an encoder neural network that emits both a latent position and a probability  $r_k$  that the  $k^{\text{th}}$  decoder should be used to recreate the data. Train the encoder alongside all decoders by minimizing the sum of squared reconstruction errors.*
2. *This model can be viewed as a mixture of decoders. Derive and implement an EM algorithm to train the model [8].*
3. Investigate if the learned model recovers meaningful clusters of the data.

## Chapter 13

### *Statistical identifiability*

*In which we're all unique.*

The opening motivation of this book was to ask whether we can arrive at a statistically identifiable way to interact with the latent variables of a generative model. We have almost achieved this, since we have shown that Riemannian pullback metrics are invariant under reparametrizations. While this factors out reparametrizations of the latent variable space, it is, technically, not a statement about *identifiability* as that requires reasoning about densities. This chapter connects the dots to show that pullback metrics indeed solve the identifiability problem.

#### *13.1. Identifiability and indeterminacy*

We consider the fairly broad family of latent variable models where observations are generated as samples from a latent distribution  $p(\mathbf{x})$  that are mapped to observation space and corrupted by observation noise. Specifically, we write

$$\mathbf{x} \sim p(\mathbf{x}) \quad (\text{latent sampling}) \quad (13.1)$$

$$\hat{\mathbf{y}} = f(\mathbf{x}) \quad (\text{decoding}) \quad (13.2)$$

$$\mathbf{y} = \eta(\hat{\mathbf{y}}) \quad (\text{noising}) \quad (13.3)$$

Here,  $\eta : \mathbb{R}^D \rightarrow \mathbb{R}^D$  corrupts a ‘clean observation’  $\hat{\mathbf{y}}$  through an i.i.d. noise process, e.g., by adding Gaussian noise.

This process results in a distribution  $p(\mathbf{y})$ , which, in practice, is determined by a set of model parameters  $\theta$ . Usually, the latent sampling distribution  $p(\mathbf{x})$  is a priori specified, e.g., a standard Gaussian distribution.



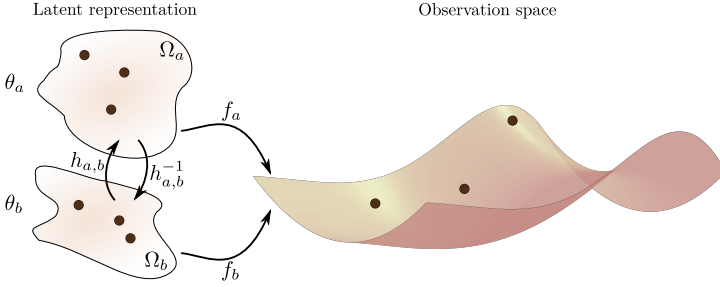


FIG. 13.2. Two different models  $\theta_a$  and  $\theta_b$  have different latent representations  $\Omega$  and decoders  $f$ . They can, however, always be linked by an indeterminacy transformation  $h_{a,b}$ .

We say that a model is *identifiable* if the mapping  $\theta \mapsto p(\mathbf{y})$  is one-to-one, i.e., if the generative distribution  $p(\mathbf{y})$  is uniquely determined by the parameter vector  $\theta$ . In practice, this is rarely the case, e.g., we have already seen that we can often reparametrize the latent space and adapt the decoder  $f$  without changing the generative process.

For a given parameter vector  $\theta$ , we can then ask which other parameters yield the same generative distribution. Notation-wise, we write  $p_{\theta_a}(\mathbf{y}) = p_{\theta_b}(\mathbf{y})$  if the parameters  $\theta_a$  and  $\theta_b$  result in identical distributions, and let the set of equal-model parameters be

$$[\theta] = \{\theta' : p_{\theta}(\mathbf{y}) = p_{\theta'}(\mathbf{y})\}. \quad (13.4)$$

In this notation, the identifiability question is then to determine when  $[\theta]$  only contains *one* element.

Given two parameters  $\theta_a$  and  $\theta_b$  with generative distributions  $p_a = p_b$ , an *indeterminacy transformation* is a measurable function  $h_{a,b} : \Omega_a \rightarrow \Omega_b$  such that  $p_a = p_b$  and  $f_a \circ h_{a,b}^{-1} = f_b$ ; c.f. Fig. 13.2. These indeterminacy transformations are, thus, reparametrizations of the latent space. Neatly, Xi and Bloem-Reddy [51] has shown that the set of all indeterminacy transformations fully determines the equivalence

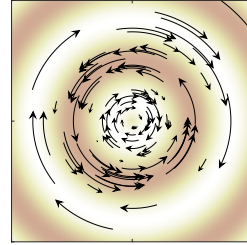


FIG. 13.1. An example of a reparametrization given by a function  $h : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ . This figure is a repetition of Fig. 1.3.

class  $[\theta]$ . In other words, all elements of  $[\theta]$  are due to indeterminacy transformations, and we only have to reason about those to answer questions of identifiability.

### 13.2. Assumptions

To characterize the set of indeterminacy transformations, we place some restrictions on our generative process. We consider decoders that are smooth functions  $f : \Omega \rightarrow \mathbb{R}^D$  such that

**A1**  $\Omega$  is compact;

**A2**  $f$  is injective;

**A3** the Jacobian of  $f$  has full column rank;

**A4** for any decoders induced by  $[\theta]$ , we have  $f_a(\Omega) = f_b(\Omega) = \mathcal{M}$ .

The first assumption is purely technical and can be interpreted as us only considering a compact subset of the latent space. The last two assumptions are quite broad, and they allow us to leverage the differential geometric tools from previous chapters. The second assumption is, however, restrictive. If the set of possible decoders is, e.g., given by a wide family of neural networks, then it is difficult to provide guarantees about injectivity. We will nonetheless proceed following Syrota et al. [44].

### 13.3. Generator transformations

Since two models  $\theta_a, \theta_b \in [\theta]$  span the same manifold (by assumption A4), we know that pullback metrics lead to identical geodesic distances. Specifically, if we let

$$\mathbf{G}_a(\mathbf{x}) = \mathbf{J}_a^\top \mathbf{J}_a \quad \text{and} \quad \mathbf{G}_b(\mathbf{x}) = \mathbf{J}_b^\top \mathbf{J}_b, \quad (13.5)$$

denote the pullback metrics associated with the latent spaces of the two models, then

$$\text{dist}_a \left( \mathbf{x}_1^{(a)}, \mathbf{x}_2^{(a)} \right) = \text{dist}_b \left( \mathbf{x}_1^{(b)}, \mathbf{x}_2^{(b)} \right). \quad (13.6)$$

Here,  $\mathbf{x}_i^{(a)}$  and  $\mathbf{x}_i^{(b)}$  denotes the latent representation of the clean observation  $\hat{\mathbf{y}}_i$  in both models.

If we know the indeterminacy transformation  $h_{a,b} : \Omega_a \rightarrow \Omega_b$ , we can write the two pullback metrics in terms of each other by recalling that  $f_a \circ h_{a,b}^{-1} = f_b$ . Hence,

$$\mathbf{J}_a = \mathbf{J}_b \frac{\partial h_{a,b}}{\partial \mathbf{x}} \Rightarrow \mathbf{G}_a = \frac{\partial h_{a,b}}{\partial \mathbf{x}}^\top \mathbf{G}_b \frac{\partial h_{a,b}}{\partial \mathbf{x}} \quad (13.7)$$

$$\mathbf{J}_b = \mathbf{J}_a \frac{\partial h_{a,b}^{-1}}{\partial \mathbf{x}} \Rightarrow \mathbf{G}_b = \frac{\partial h_{a,b}^{-1}}{\partial \mathbf{x}}^\top \mathbf{G}_a \frac{\partial h_{a,b}^{-1}}{\partial \mathbf{x}}. \quad (13.8)$$

The mapping  $f_a \circ h_{a,b}^{-1} = f_b$  is called a *generator transform*. This tells us how, given an indeterminacy transformation, we can link two models geometrically. In practice, we, however, don't know the indeterminacy transformations associated with a model. Excitingly, Xi and Bloem-Reddy [51] shows that any indeterminacy transformation must be almost everywhere equal to the above construction, implying that indeterminacy transformations and generator transforms are two sides to the same coin.

This is sufficient to see that geometric properties such as lengths of curves, distances, angles, volumes, and so forth are identifiable.

### 13.4. Back to map-making

In Chapter 2, we saw that Gauss's *Theorema Egregium* implied that because planet Earth is a curved object, it is impossible to make a flat map thereof. We are now equipped to make similar statements for the identifiability of latent variables.

Specifically, if we assume a Euclidean metric on the latent space  $\Omega$ , then we can only identify the latent variables if the decoder parametrizes a *flat manifold*. Otherwise, the indeterminacy transformations would change the metric to be non-Euclidean.

This suggests that the quest for an identifiable *Euclidean* latent space is futile except for the simplest of models.

## Exercises

### Exercise 13.1: Probabilistic PCA.

Consider the generative model [46]

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}) \quad \mathbf{x} \in \mathbb{R}^d \quad (13.9)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{A}\mathbf{x} + \mathbf{b}, \sigma^2\mathbf{I}) \quad \mathbf{y} \in \mathbb{R}^D \quad (13.10)$$

where  $\mathbf{A} \in \mathbb{R}^{D \times d}$  and  $\mathbf{b} \in \mathbb{R}^D$ .

1. From this model description, explicate the three generative steps in eqs. 13.1–13.3.
2. Let  $\Omega = \{\mathbf{x} \mid \|\mathbf{x}\| < 10\}$  be a compact subset of  $\mathbb{R}^d$ . Provide conditions under which assumptions A2–A4 hold.
3. Derive the pullback metric and the associated geodesic distance under this model.

### Exercise 13.2: Variational autoencoders.

Consider the generative model [26, 37]

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}) \quad \mathbf{x} \in \mathbb{R}^d \quad (13.11)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{x}|f(\mathbf{x}), \sigma^2\mathbf{I}) \quad \mathbf{y} \in \mathbb{R}^D \quad (13.12)$$

where  $f: \mathbb{R}^d \rightarrow \mathbb{R}^D$  is the neural network

$$f(\mathbf{x}) = \tanh(\mathbf{W}_2 \tanh(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2). \quad (13.13)$$

Here  $\mathbf{W}_1 \in \mathbb{R}^{H \times d}$ ,  $\mathbf{W}_2 \in \mathbb{R}^{D \times H}$ ,  $\mathbf{b}_1 \in \mathbb{R}^H$ ,  $\mathbf{b}_2 \in \mathbb{R}^D$  with the condition that  $D > H > d$ .

1. From this model description, explicate the three generative steps in eqs. 13.1–13.3.
2. Let  $\Omega = \{\mathbf{x} \mid \|\mathbf{x}\| < 10\}$  be a compact subset of  $\mathbb{R}^d$ . Provide conditions under which assumptions A2–A4 hold.
3. Derive the pullback metric under this model.

## Chapter 14

### *Statistical manifolds*

*In which we are probably  
in the ambient space.*

So far, we have considered manifolds spanned by functions  $f : \mathbb{R}^d \rightarrow \mathbb{R}^D$  that map from one real space to another. Often, the ambient space has more structure than a generic real space  $\mathbb{R}^D$ .

In this chapter, we touch on the case where  $f$  outputs probability distributions. This is an ever-present case in machine learning, e.g., a classifier emits *class probabilities*, corresponding to a categorical distribution. Likewise, regression models can often be seen as emitting Gaussian distributions.

The sought-after geometry is known as *information geometry*, and the individual manifolds are commonly called *statistical manifolds*.

#### *14.1. The space of distributions*

We seek to characterize the geometry of the space of probability distributions. The idea is simple: we consider distributions  $p(\mathbf{y}|\theta)$  that are governed by a set of parameters  $\theta$ . The space of distributions will then simply be the space in which  $\theta$  resides.

For example, if  $p(\mathbf{y}|\theta)$  is the univariate Gaussian  $\mathcal{N}(\mu, \sigma^2)$  then the parameters are  $\theta = (\mu, \sigma)$ . The Gaussian distribution is a strictly positive function

$$\mathcal{N}(\mathbf{y}|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|\mathbf{y} - \mu\|^2}{2\sigma^2}\right) \quad (14.1)$$

that integrates to one. We can think of such functions as points in the infinite-dimensional space of continuous functions. This set of points

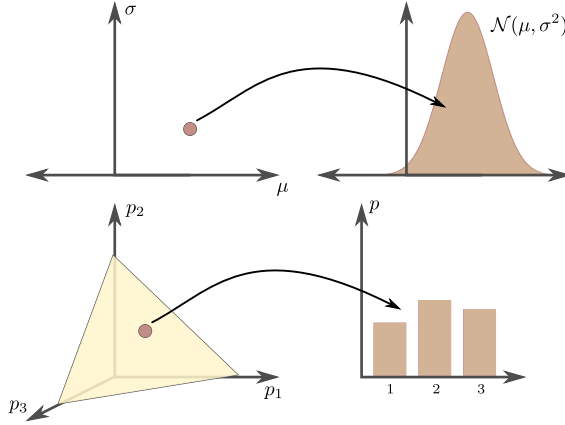


FIG. 14.1. *Examples of the space of distributions. Top: The space of normal distributions  $\mathcal{N}(\mu, \sigma^2)$  can, e.g., be represented as  $(\mu, \sigma) \in \mathbb{R} \times \mathbb{R}_+$ . Bottom: The space of categorical distributions can be seen as points on the unit simplex.*

has an intricate geometry that we seek to understand. Carrying out the analysis within the infinite-dimensional space of functions is, however, too daunting a task, and we instead work in the space of parameters. In the Gaussian case, this is  $\mathbb{R} \times \mathbb{R}_+$  since  $\mu$  is real-valued and  $\sigma$  is positive. We refer to this parameter space as *the space of distributions* and denote it  $\mathcal{H}$ . It is this space that we seek to give a geometry.

Similarly, in the multivariate case where  $p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mu, \Sigma)$ , we have  $\theta = (\mu, \Sigma)$ . Since  $\mu \in \mathbb{R}^D$  and (the covariance is positive definite)  $\Sigma \in \text{Sym}_+^{D \times D}$ , we have that the space of such distributions is  $\mathcal{H} = \mathbb{R}^D \times \text{Sym}_+^{D \times D}$ .

As another example, the *categorical* distribution over  $D$  possible classes has parameters  $\theta = (p_1, \dots, p_D)$ , where  $p_d$  is the probability of the  $d^{\text{th}}$  class. The space of distributions  $\mathcal{H}$  is then the unit simplex (the set of positive numbers summing to one).

## 14.2. Naive distances between distributions

Having a space of distributions  $\mathcal{H}$ , we now seek a distance measure on this space, i.e., a way to determine the similarity between two

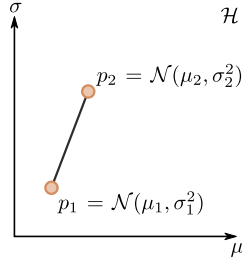


FIG. 14.2. The Euclidean distance between two univariate Gaussian parametrized as  $\theta = (\mu, \sigma)$ . There is much arbitrariness in this choice of distance measure.

distributions.

An obvious candidate distance measure is the Euclidean distance over  $\mathcal{H}$ . For example, if we have two univariate Gaussian distributions,  $p_1 = \mathcal{N}(\mu_1, \sigma_1^2)$  and  $p_2 = \mathcal{N}(\mu_2, \sigma_2^2)$ , and parameterize the distributions as  $\theta = (\mu, \sigma)$ , then the Euclidean distance becomes

$$\text{dist}(p_1, p_2) = \sqrt{(\mu_1 - \mu_2)^2 + (\sigma_1 - \sigma_2)^2}. \quad (14.2)$$

However, we could have equally well chosen to parametrize the distributions as  $\theta = (\mu, \sigma^2)$ , which would result in a different Euclidean distance. Like so many times before, we see that how we parametrize things changes how we measure.

Arguably, it would be desirable if we had access to a distance measure over the space of distributions that did not depend on how we had chosen to parametrize the distributions. A *reparametrization invariant* distance measure, that is.

### 14.3. Divergences are not distances

Fortunately, we already have a notion of similarity between distributions that is invariant to how we parametrize the distribution: *statistical divergences*. The most celebrated such *divergence* is the *Kullback–Leibler (KL) divergence*, defined as

$$\text{KL}(p||q) = \int \log \left( \frac{p(\mathbf{y})}{q(\mathbf{y})} \right) p(\mathbf{y}) d\mathbf{y} = \mathbb{E}_{p(\mathbf{y})} \left[ \log \left( \frac{p(\mathbf{y})}{q(\mathbf{y})} \right) \right] \quad (14.3)$$

for two distributions  $p(\mathbf{y})$  and  $q(\mathbf{y})$ . Since this divergence only relies on the densities, and not their parameterizations, it is by construction

reparametrization invariant. We can also think of the divergence as being defined in the infinite-dimensional function space in which the density resides, rather than in the parameter space.

We can easily observe that the KL-divergence is positive for distinct distributions using *Jensen's inequality*<sup>1</sup>

$$\text{KL}(p\|q) = - \int \log \left( \frac{q(\mathbf{y})}{p(\mathbf{y})} \right) p(\mathbf{y}) d\mathbf{y} \quad (14.4)$$

$$\geq - \log \underbrace{\int \frac{q(\mathbf{y})}{p(\mathbf{y})} p(\mathbf{y}) d\mathbf{y}}_{=1} = 0. \quad (14.5)$$

By Jensen's inequality, we also see that equality only holds when  $p(\mathbf{y}) = q(\mathbf{y})$  for all  $\mathbf{y}$ . The divergence, thus, shares some properties with distance functions.

The KL-divergence is, however, not a distance measure as it is generally *not* symmetric, i.e.,  $\text{KL}(p\|q) \neq \text{KL}(q\|p)$ . This prevents us from directly using the KL-divergence as a distance function over the space of distributions.

#### 14.4. Curve lengths and energies, revisited

It turns out that we actually can use the KL-divergence to arrive at a distance measure in the space of distributions. To see this, recall the initial definition of the length of a curve (4.3), in the space of distributions. We repeat it here in a slightly modified form

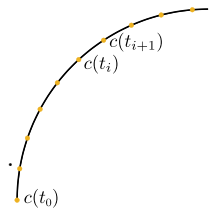
$$\text{Length}(c) = \lim_{N \rightarrow \infty} \sum_{i=0}^N \text{dist}(p(\mathbf{y}|c(t_i)), p(\mathbf{y}|c(t_{i+1}))).$$

Similarly, for the *energy* of a curve

$$\mathcal{E}(c) = \lim_{N \rightarrow \infty} \sum_{i=0}^N \text{dist}^2(p(\mathbf{y}|c(t_i)), p(\mathbf{y}|c(t_{i+1}))).$$

We, thus, see that we only need our choice of similarity measure between distributions to be a distance function *locally*. If we have such, we can proceed as in earlier chapters to arrive at a suitable geometry.

<sup>1</sup>For a concave function  $\phi$ ,  $\phi(\mathbb{E}[\mathbf{y}]) \geq \mathbb{E}[\phi(\mathbf{y})]$ .





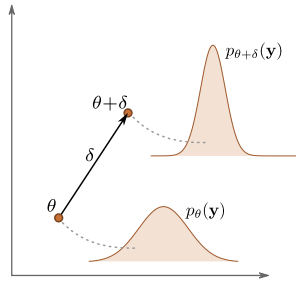


FIG. 14.3. The Fisher-Rao metric compares distributions locally through the KL-divergence resulting in a Riemannian metric.

### 14.5. The Fisher-Rao metric

We can analyze the local behavior of the KL-divergence through a Taylor expansion. To fix notation, let  $p(\mathbf{y}) = p(\mathbf{y}|\theta)$  be a distribution parametrized by  $\theta$ , and let  $q(\mathbf{y}) = p(\mathbf{y}|\theta + \delta)$  be an infinitesimal displacement with respect to  $\theta$ . We will then make a second-order Taylor expansion around  $\theta$ ,

$$\text{KL}(p\|q) = -\frac{1}{2}\delta^\top \underbrace{\int p(\mathbf{y}) \nabla_\theta^2 \log p(\mathbf{y}) d\mathbf{y}}_{\mathbf{F}} \delta + \mathcal{O}(\|\delta\|^3), \quad (14.6)$$

where  $\nabla_\theta^2$  denotes the Hessian. We detail this computation in exercise 14.2. Infinitesimally, we, thus, see that the KL-divergence is a simple quadratic equation,  $\text{KL}(p\|q) = \delta^\top \mathbf{F} \delta$ .

The matrix  $\mathbf{F}$  can be rewritten in the more elegant form as (derived in exercise 14.2)

$$\mathbf{F} = -\frac{1}{2} \int p(\mathbf{y}) \nabla^2 \log p(\mathbf{y}) d\mathbf{y} \quad (14.7)$$

$$= \frac{1}{2} \int p(\mathbf{y}) \nabla_\theta \log p(\mathbf{y}) \nabla_\theta \log p(\mathbf{y})^\top d\mathbf{y} \quad (14.8)$$

That is, locally, the KL-divergence is a quadratic function given as the expectation of the outer product of the gradient of  $\log p(\mathbf{y})$  by itself. By construction, the matrix  $\mathbf{F}$  is positive definite, such that we can interpret  $\mathbf{F}$  as a Riemannian metric over the space of distributions. This metric is commonly known as the *Fisher-Rao metric*.

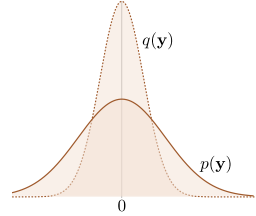
In summary, the KL-divergence is not a distance function between distributions, but locally it is. Even better, *locally, the KL-divergence is a Riemannian metric*.

A desirable property of the Fisher–Rao metric is that it does not depend on how we parametrize  $p(\mathbf{y})$ . For example, if  $p(\mathbf{y})$  is a normal distribution, it does not matter if we use a *mean–variance* or *mean–precision* parametrization. Why does the parametrization not matter? Because Fisher–Rao measures changes in the KL divergence.

### 14.6. Should we have distances?

The KL-divergence is inherently an asymmetric measure, i.e.,  $\text{KL}(p\|q) \neq \text{KL}(q\|p)$ . While this property can be annoying when we seek a distance measure, it is fundamentally a sound property.

Consider two Gaussian distributions over  $\mathbf{y} \in \mathbb{R}$ ,  $p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|0, \sigma_p^2)$  and  $q(\mathbf{y}) = \mathcal{N}(\mathbf{y}|0, \sigma_q^2)$ , where  $\sigma_p \gg \sigma_q$ ; see Fig. 14.4. If we draw samples from  $p(\mathbf{y})$  and measure their log-likelihood under  $q(\mathbf{y})$ , we get a notably different quantity compared to sampling from  $q(\mathbf{y})$  and measuring under  $\log p(\mathbf{y})$ , i.e.,



$$\mathbb{E}_{\mathbf{y} \sim p(\mathbf{y})}[\log q(\mathbf{y})] \neq \mathbb{E}_{\mathbf{y} \sim q(\mathbf{y})}[\log p(\mathbf{y})]. \quad (14.9)$$

FIG. 14.4. Two zero-centered Gaussians.

In the particular case, samples from  $q(\mathbf{y})$  would be very likely under  $p(\mathbf{y})$ , while the opposite would not be true. Fundamentally, the relationship between  $p(\mathbf{y})$  and  $q(\mathbf{y})$  is asymmetric, suggesting that perhaps we should be careful about placing (symmetric) distance functions over the space of distributions.

In practice, there will be cases where having a distance function is both suitable and practical, and there will be cases where such are misleading.

### 14.7. Natural gradients

A commonly applied tool from information geometry is the *natural gradient*. This is a gradient of some loss with respect to the parameters

of a statistical model. Unlike the traditional Euclidean gradient, this, however, takes the geometry of the space of distributions into account, which often leads to improvements in optimization.

We commonly use gradients to minimize some loss function  $L$  with respect to some parameters. The simplest algorithm, *gradient descent*, merely takes a small step in the direction of the gradient,

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \frac{\partial L}{\partial \theta^{(t)}}, \quad (14.10)$$

where  $\alpha$  controls the step size.

One motivation for this algorithm is that the gradient is the direction of *steepest descent*, i.e., the gradient points in the direction where the loss will change the most. To see this, we first make a Taylor expansion of the loss,

$$L(\theta + \delta) \approx L(\theta) + \frac{\partial L}{\partial \theta}^\top \delta. \quad (14.11)$$

The quantity  $\partial L / \partial \theta^\top \delta$ , thus, locally measures how much our loss changes if we move in the direction  $\delta$ .

The direction of steepest descent can then be found by minimizing  $\partial L / \partial \theta^\top \delta$ . To make our conclusion invariant to the scale of  $\delta$  (we are only interested in the *direction*), we impose the constraint that  $\|\delta\| = c$  for some small constant  $c$ . The direction of steepest descent can then be found by solving

$$\min_{\delta} \frac{\partial L}{\partial \theta}^\top \delta \quad \text{subject to} \quad \|\delta\| = c. \quad (14.12)$$

In exercise 14.4, we derive that the answer is  $\delta \propto -\partial L / \partial \theta$ . Informally, we can arrive at this answer by noting that

$$\frac{\partial L}{\partial \theta}^\top \delta = \left\| \frac{\partial L}{\partial \theta} \right\| \|\delta\| \cos \psi, \quad (14.13)$$

where  $\psi$  is the angle between  $\partial L / \partial \theta$  and  $\delta$ . To make Eq. 14.13 as small as possible, we should pick  $\psi$  such that  $\cos \psi = -1$ , implying that  $\delta \propto -\partial L / \partial \theta$ .

In the above, we impose the constraint  $\|\delta\| = c$  to focus our attention on the direction of  $\delta$ . Earlier in this chapter, we have,

however, seen that the Fisher–Rao metric  $\mathbf{F}$  provides a more natural norm when engaging with probabilistic models. It, thus, might be more natural to instead impose the constraint  $\|\delta\|_{\mathbf{F}} = c$ . Then, the direction of steepest descent becomes

$$\delta \propto -\mathbf{F}^{-1} \frac{\partial L}{\partial \theta}. \quad (14.14)$$

The proof of this result appears in exercise 14.4. The quantity  $\mathbf{F}^{-1} \partial L / \partial \theta$  is known as the *natural gradient*.

Is this gradient ‘natural’? The answer is clearly subjective, but the natural gradient is a good choice when we want to measure the loss reduction through the KL-divergence. For most maximum likelihood problems, this should be the case.

Computationally, natural gradient descent can be daunting as we both need to compute and invert the Fisher–Rao metric. Sometimes this can be carried out efficiently, and in other cases approximations of the Fisher–Rao may be required.

## 14.8. Exponential families

We have already seen that we can compute the Fisher–Rao metric through the Hessian of the KL-divergence. For some distributions, we can get more explicit expressions, which can reduce the required computations.

A fairly broad class of distributions is known as the *exponential family*, which takes the form

$$p(\mathbf{y}|\theta) = h(\mathbf{y}) \exp(\eta(\theta)^\top T(\mathbf{y}) - \psi(\theta)), \quad (14.15)$$

where  $\mathbf{y}$  is the modeled variable,  $\theta$  are the parameters of the distribution, and  $h(\mathbf{y})$  must be non-negative.

As an example, consider the Gaussian distribution

$$p(\mathbf{y}|\theta) = \mathcal{N}(\mathbf{y}|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mathbf{y} - \mu)^2}{2\sigma^2}\right). \quad (14.16)$$

Here, the parameters are  $\theta = (\mu, \sigma^2)$ . We can bring the Gaussian on the form of an exponential family by expanding the square and moving

terms around,

$$\mathcal{N}(\mathbf{y}|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\mathbf{y}^2 + \mu^2 - 2y\mu}{2\sigma^2} - \log(\sigma)\right) \quad (14.17)$$

$$= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\mathbf{y}^2}{2\sigma^2} - \frac{\mu^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \log(\sigma)\right). \quad (14.18)$$

With this rewrite in hand, we define

$$h(\mathbf{y}) = \frac{1}{\sqrt{2\pi}} \quad (14.19)$$

$$\eta(\theta) = (\mu/\sigma^2, -1/2\sigma^2) \quad (14.20)$$

$$T(\mathbf{y}) = (\mathbf{y}, \mathbf{y}^2) \quad (14.21)$$

$$\psi(\theta) = \mu^2/2\sigma^2 + \log(\sigma) \quad (14.22)$$

to note that the Gaussian distribution is a part of the exponential family.

The exponential family is a broad class of distributions that captures many commonly used models. Interestingly, the associated Fisher–Rao metric can be written fairly explicitly [24],

$$\mathbf{F}_{\text{ef}}(\theta) = \left(\frac{\partial \eta}{\partial \theta}\right)^\top \left(\frac{\partial \mu}{\partial \theta}\right), \quad (14.23)$$

where  $\mu(\theta) = \mathbb{E}[T(\mathbf{y}) \mid \theta]$ . When  $\mu$  is computable, either in closed-form or using Monte Carlo, we can, thus, use automatic differentiation to evaluate the Fisher–Rao metric.

This result can also be used to directly evaluate curve energies

$$\mathcal{E}[c] = \int_0^1 \dot{c}_t^\top \mathbf{F}_{\text{ef}}(c_t) \dot{c}_t dt \quad (14.24)$$

$$= \int_0^1 \dot{c}_t^\top \left(\frac{\partial \eta(c_t)}{\partial c_t}\right)^\top \left(\frac{\partial \mu(c_t)}{\partial c_t}\right) \dot{c}_t dt \quad (14.25)$$

$$= \int_0^1 \left(\frac{d\eta(c_t)}{dt}\right)^\top \left(\frac{d\mu(c_t)}{dt}\right) dt, \quad (14.26)$$

which can often be efficiently evaluated using automatic differentiation. Similar calculations gives the following expression for curve length

$$\text{Length}[c] = \int_0^1 \sqrt{\left(\frac{d\eta(c_t)}{dt}\right)^\top \left(\frac{d\mu(c_t)}{dt}\right)} dt. \quad (14.27)$$

## Exercises

### Exercise 14.1: Choice of parametrization.

Consider two normal distributions

$$\mathcal{N}_1 = \mathcal{N}(\mu_1, \sigma_1^2) \quad \text{and} \quad \mathcal{N}_2 = \mathcal{N}(\mu_2, \sigma_2^2), \quad (14.28)$$

where

$$\mu_1 = 0, \quad \mu_2 = 1, \quad \sigma_1 = 1, \quad \sigma_2 = 2. \quad (14.29)$$

1. *Compute the Euclidean distance between these distributions in the following parametrizations*

$$\theta = (\mu, \sigma) \quad (14.30)$$

$$\theta = (\mu, \sigma^2) \quad (14.31)$$

$$\theta = (\mu, \sigma^{-2}) \quad (14.32)$$

$$\theta = \left( \frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2} \right). \quad (14.33)$$

The latter is known as the ‘*natural parametrization*’ of the Gaussian.

2. *Does one parametrization seem more natural (or better) than another (this question is open-ended)?*
3. *Find parametrization of normal distributions such that the associated pull-back metric corresponds to that of the randomly projected manifold (15.8).*

### Exercise 14.2: Deriving Fisher–Rao.

Consider the distribution  $p_\theta(\mathbf{y})$  parametrized by  $\theta$ , and let  $p_{\theta+\delta}(\mathbf{y})$  be an infinitesimal displacement with respect to  $\theta$ . The KL-divergence between these is

$$\begin{aligned} \text{KL}(p_\theta \| p_{\theta+\delta}) &= \int \log(p_\theta(\mathbf{y})) p_\theta(\mathbf{y}) d\mathbf{y} \\ &\quad - \int \log(p_{\theta+\delta}(\mathbf{y})) p_\theta(\mathbf{y}) d\mathbf{y}. \end{aligned} \quad (14.38)$$

1. Make a second-order Taylor expansion of  $\log p_\theta(\mathbf{y})$  with respect to  $\theta$ .
2. Use the Taylor expansion to approximate  $p_{\theta+\delta}$  to derive Eq. 14.6.
3. Show that the Hessian with respect to  $\theta$  of  $\log p_\theta(\mathbf{y})$  is

$$\begin{aligned}\nabla_\theta^2 \log p_\theta(\mathbf{y}) &= \frac{1}{p_\theta(\mathbf{y})} \nabla_\theta^2 p_\theta(\mathbf{y}) \\ &\quad - \nabla_\theta \log p_\theta(\mathbf{y}) \nabla_\theta \log p_\theta(\mathbf{y})^\top.\end{aligned}\tag{14.39}$$

4. Finally, show that the Fisher–Rao metric is

$$\mathbf{G} = \frac{1}{2} \int p_\theta(\mathbf{y}) \nabla_\theta \log p_\theta(\mathbf{y}) \nabla_\theta \log p_\theta(\mathbf{y})^\top d\mathbf{y}.\tag{14.40}$$

*Exercise 14.3: A Poisson example.*

Consider the Poisson distribution over the set of non-negative integers  $k \in \mathbb{N}_0$  with probability mass function

$$P(k|\lambda) = \frac{\exp(-\lambda)\lambda^k}{k!}.\tag{14.46}$$

Here  $\lambda \in (0, \infty)$  is the so-called ‘rate’ parameter of the distribution.

As the distribution is over  $\mathbb{N}_0$ , the KL-divergence takes the form

$$\text{KL}(P(k|\lambda_1) \| P(k|\lambda_2)) = \mathbb{E}_{P(k|\lambda_1)} \left[ \log \frac{P(k|\lambda_1)}{P(k|\lambda_2)} \right],\tag{14.47}$$

where

$$\mathbb{E}_{P(k|\lambda_1)}[h(k)] = \sum_{k=0}^{\infty} h(k) P(k|\lambda_1)\tag{14.48}$$

denotes the expectation with respect to the distribution  $P(k|\lambda_1)$ .

- Derive a closed-form expression for the KL-divergence (14.47) between two Poisson distributions with parameters  $\lambda_1$  and  $\lambda_2$ , respectively. The final expression should depend only  $\lambda_1$  and  $\lambda_2$ . Hint: it may be useful to know that  $\mathbb{E}_{P(k|\lambda)}[k] = \lambda$ .

- Derive an expression for the Fisher–Rao metric for the Poisson distribution over the parameter  $\lambda \in (0, \infty)$ .

*Exercise 14.4: Natural gradients.*

In this exercise, we derive the natural gradient as the direction of steepest descent under the Fisher–Rao metric. We first establish the proof technique by showing that the Euclidean gradient is the steepest descent direction under the Euclidean metric.

1. The direction of steepest descent under the Euclidean metric is the solution to

$$\min_{\delta} \frac{\partial L}{\partial \theta}^{\top} \delta \quad \text{subject to} \quad \|\delta\| = c. \quad (14.49)$$

Turn the constrained optimization problem into an unconstrained one using the technique of Lagrange multipliers. Use this to show that the optimal direction is

$$\delta = \frac{c}{\|\partial L / \partial \theta\|} \frac{\partial L}{\partial \theta}. \quad (14.50)$$

From this, conclude that the negative gradient is the direction of steepest Euclidean descent.

2. Next, consider the natural gradient. This is the solution to the optimization problem

$$\min_{\delta} \frac{\partial L}{\partial \theta}^{\top} \delta \quad \text{subject to} \quad \delta^{\top} \mathbf{F} \delta = c^2. \quad (14.51)$$

Use the technique of Lagrange multipliers to show that the direction of steepest descent is then

$$\delta \propto -\mathbf{F}^{-1} \partial L / \partial \theta, \quad (14.52)$$

i.e., the natural gradient.

*Exercise 14.5: Deriving the Gaussian KL-divergence.*

Consider two Gaussian distributions over  $\mathbf{y} \in \mathbb{R}^D$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mu_p, \Sigma_p) \quad \text{and} \quad q(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mu_q, \sigma_q). \quad (14.53)$$



1. Assume that  $\Sigma_p = \Sigma_q = \mathbf{I}$  and derive an expression for the KL-divergence  $\text{KL}(p||q)$ .
2. Assume that  $\Sigma_p = \Sigma_q$  and derive an expression for the KL-divergence  $\text{KL}(p||q)$ .
3. Assume that  $\Sigma_p \neq \Sigma_q$  and derive an expression for the KL-divergence  $\text{KL}(p||q)$ .

*Exercise 14.6: The Gaussian Fisher–Rao metric.*

Consider a Gaussian distributions over  $\mathbf{y} \in \mathbb{R}$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mu, \sigma^2). \quad (14.54)$$

1. Derive Fisher–Rao metric over the  $(\mu, \sigma)$  space.
2. Derive the Fisher–Rao metric over the natural parameters of the distribution.

*Exercise 14.7: Asymmetry of the KL-divergence.*

The KL-divergence is inherently an asymmetric measure, i.e.,  $\text{KL}(p||q) \neq \text{KL}(q||p)$ . While this property can be annoying when we seek a distance measure, it is fundamentally a sound property. Consider two Gaussian distributions over  $\mathbf{y} \in \mathbb{R}$

$$p(\mathbf{y}) = \mathcal{N}(0, \sigma_p^2) \quad \text{and} \quad q(\mathbf{y}) = \mathcal{N}(0, \sigma_q^2), \quad (14.55)$$

where  $\sigma_p \gg \sigma_q$ .

1. Pick values of  $\sigma_p$  and  $\sigma_q$  and numerically draw 100 samples from each distribution. Evaluate the log-likelihood under  $p$  of samples from  $q$  and vice versa. Do samples from one distribution have a higher likelihood than the other? Is this to be expected?

## Chapter 15

### *Noisy manifolds*

*In which data ruins the party  
only to randomly save it again.*

From a statistics or machine learning perspective, our interest is in manifolds that are estimated from data. As this data is practically always finite in amount and subject to observation noise, we should start off with the acknowledgement that the estimated manifold is just that: *an estimate*. But how does this imperfection reveal itself in the geometry of the manifold?

#### *15.1. Manifold learning, revisited*

Assume we have some data  $\mathbf{y}_{1:N} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$  in  $\mathbb{R}^D$  and that we want to fit a  $d$ -dimensional manifold to this data. As mentioned in chapter 3, a common strategy, e.g., employed by autoencoders [41], is to find a function  $f$  that minimizes the squared distance between observations and projections thereof onto the manifold:

$$f^* = \operatorname{argmin}_f \sum_{n=1}^N \|f(\mathbf{x}_n) - \mathbf{y}_n\|^2, \quad (15.1)$$

where  $\mathbf{x}_n$  is chosen such that  $f(\mathbf{x}_n)$  is the projection<sup>1</sup> of  $\mathbf{y}_n$  onto the manifold spanned by  $f$ :

$$\mathbf{x}_n = \operatorname{proj}_{\mathcal{M}}(\mathbf{y}_n) = \operatorname{argmin}_{\mathbf{x}} \|f(\mathbf{x}) - \mathbf{y}_n\|^2. \quad (15.2)$$

---

<sup>1</sup>Technically, an autoencoder does not optimize the representations  $\mathbf{x}_{1:N}$ , but rather optimizes a neural network that predicts the representation  $\mathbf{x}_n$  from an observation  $\mathbf{y}_n$ . This is irrelevant from our perspective as it is merely an approximation.

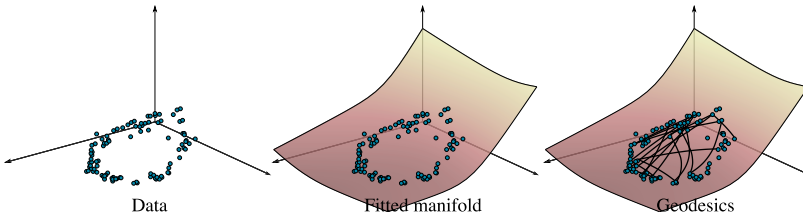


FIG. 15.1. *Fitting a manifold to data gives a smooth function that well-interpolates the data. The geometry of the learned manifold does, however, not reflect the structure of the observed data.*

Combining Eqs. 15.1 and 15.2 into one optimization problem gives

$$f^*, \mathbf{x}_1^*, \dots, \mathbf{x}_n^* = \operatorname{argmin}_{f, \mathbf{x}_{1:N}} \sum_{n=1}^N \|f(\mathbf{x}_n) - \mathbf{y}_n\|^2. \quad (15.3)$$

In the above, we write that we minimize our loss function with respect to the function  $f$ . In practice, this means that we minimize with respect to the unspecified parameters governing  $f$ , e.g., if  $f$  is a polynomial, the parameters are the coefficients, or if  $f$  is a neural network, the parameters are the weights.

Figure 15.1 gives an example of this in practice. We start from some data with a circular structure in  $\mathbb{R}^3$  to which a function  $f$  is fitted. Due to either implicit or explicit *regularization*, the result is quite smooth.<sup>2</sup> In the right-most panel of the figure, we observe that geodesics on the manifold leverage this smooth structure to ‘cut across’ the hole of the circular structure. Geodesics fail to capture the structure hidden in the data.

This example highlights an unfortunate situation: we want our models to be smooth, as experience suggests that such models are more reliable, but then geodesics follow the structure induced by the smoothness rather than that of the actual data. This suggests that the Riemannian geometry induced by  $f$  is a poor match for the geometry that governs the data. A rather depressing finding...

---

<sup>2</sup>By *smooth* we refer to how much the manifold ‘wiggles’, i.e., if the spectrum of the underlying function  $f$  is dominated by low-frequency components.

## 15.2. Gaussian noisy manifolds

Perhaps the result of Fig. 15.1 is unsurprising. After all, we are trying to estimate a function  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$  that deforms  $\mathbb{R}^2$  to match a circle. This must imply that parts of  $\mathbb{R}^2$  will be mapped to regions of  $\mathbb{R}^3$  that are not near data. The lack of data should imply that the function  $f$  will be poorly estimated in such regions. We, therefore, turn our attention to other models that seek to estimate the quality of the fit.

Instead of working with a deterministic model  $\mathbf{y} = f(\mathbf{x})$ , we now consider a probabilistic model where we have additional uncertainty.<sup>3</sup> We will consider the additive Gaussian noise model where

$$\mathbf{y} = f(\mathbf{x}) + \epsilon \sigma(\mathbf{x}), \quad (15.4)$$

where the noise  $\epsilon \in \mathbb{R}^{D \times D}$  is a diagonal matrix with standard Gaussian entries, i.e.,  $[\epsilon]_{ii} \sim \mathcal{N}(0, 1)$ . Here  $\sigma(\mathbf{x}) \in \mathbb{R}^D$  is a vector of standard deviations that may change smoothly with  $\mathbf{x}$ . We can alternatively write this model as a conditional distribution

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|f(\mathbf{x}), \text{diag}(\sigma^2(\mathbf{x}))), \quad (15.5)$$

where  $\text{diag} : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times D}$  create a diagonal matrix from a vector, and  $\mathcal{N}$  denotes the density of the Gaussian distribution, i.e.,

$$\mathcal{N}(\mathbf{y}|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^D \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mu)^\top \Sigma^{-1}(\mathbf{y} - \mu)\right).$$

Examples of such models include the *generative topographic map* [9] and the *variational autoencoder* [26, 37].

## 15.3. Randomly projected manifolds

Intuitively, we see that the model in Eq. 15.4 spans a ‘manifold plus noise’, so it seems natural that our geometric tools should still apply. *But how?* We can opt to disregard the noise and focus on the mean, but then we recover the problems highlighted in the discussion around

---

<sup>3</sup>For now we focus on the case of *noisy data*. The case of truly uncertain models will be discussed in the next chapter.

Fig. 15.1, and nothing will be gained by the probabilistic extension. Instead, we will take the view that our noisy manifold is a *random projection* of a deterministic manifold.

To figure out what that means, let us rewrite Eq. 15.4 in matrix form,

$$\mathbf{y} = f(\mathbf{x}) + \epsilon \sigma(\mathbf{x}) \quad (15.6)$$

$$= \begin{pmatrix} \mathbf{I} & \epsilon \end{pmatrix} \begin{pmatrix} f(\mathbf{x}) \\ \sigma(\mathbf{x}) \end{pmatrix} = \mathbf{P}h(\mathbf{x}). \quad (15.7)$$

Here  $\mathbf{P} = (\mathbf{I}, \epsilon) \in \mathbb{R}^{D \times 2D}$  is a *random matrix*<sup>4</sup> which we can choose to think of as performing a projection from  $\mathbb{R}^{2D}$  onto  $\mathbb{R}^D$ . We can, thus, think of  $h(\mathbf{x}) = (f(\mathbf{x}); \sigma(\mathbf{x}))$  as spanning a  $d$ -dimensional manifold in  $\mathbb{R}^{2D}$ , which is then randomly projected by  $\mathbf{P}$  to form  $\mathbf{y}$ .

The idea of  $\mathbf{y}$  appearing through a random projection is interesting because it gives us access to a (deterministic) manifold  $h(\Omega) \in \mathbb{R}^{2D}$ , which we can treat just as we have done in previous chapters. This gives us access to a theoretical toolbox that we can use for analyzing *noisy manifolds* of the form found in Eq. 15.4.

Practically speaking, this result tells us that we should simply concatenate  $f(\mathbf{x}) \in \mathbb{R}^D$  and  $\sigma(\mathbf{x}) \in \mathbb{R}^D$  to form a vector in  $\mathbb{R}^{2D}$ , and treat the corresponding map  $\mathbb{R}^d \mapsto \mathbb{R}^{2D}$  as spanning a manifold.

## 15.4. An example

To illustrate that the randomly projected manifold is useful, we set out to solve a simple task. We randomly sample points on a circle in  $\mathbb{R}^2$  and pass these through a nonlinear map that embeds the circle in  $\mathbb{R}^{1000}$ . The result is a collection of points along a ‘bent circle’ in a high-dimensional space. To these points, we add Gaussian noise and fit a randomly projected manifold of the type in Eq. 15.4.

This procedure gives a low-dimensional representation, which we can give a Riemannian metric. The panels of Fig. 15.2 show this representation equipped with different metrics. The left panel disregards the noise and considers the metric associated with  $f$  alone, i.e.,  $\mathbf{G}_{\mathbf{x}} = \mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x})$ . The white curves correspond to geodesics between

---

<sup>4</sup>That’s just a fancy way of saying a matrix with random entries.

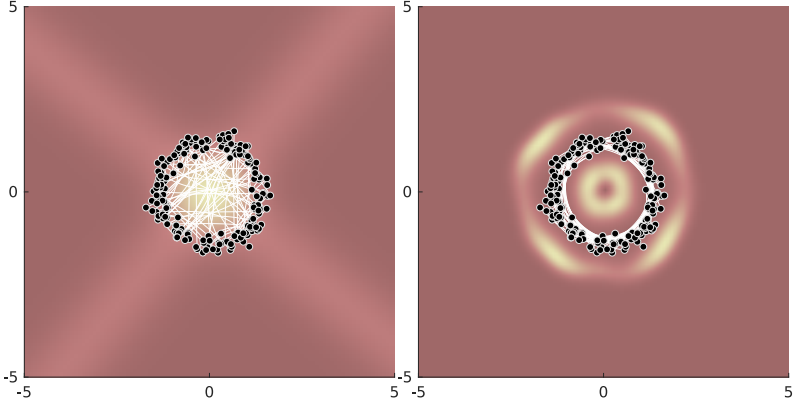


FIG. 15.2. Data distributed near a circle in a high-dimensional space is represented by a low-dimensional manifold. Left: geodesics under the metric associated with the mean of the estimated manifold. Right: geodesics under a metric that takes the uncertainty of the manifold into account. In both plots, the background color represents the volume measure associated with the metric.

randomly chosen points; it is quite evident that geodesics repeatedly move across the ‘hole’ in the circular structure, which suggests that the geometry does a poor job of reflecting the structure of the underlying data. This is in line with Fig. 15.1. The right panel of Fig. 15.2 on the other hand considers the Riemannian metric associated with  $h(\mathbf{x}) = (f(\mathbf{x}); \sigma(\mathbf{x}))$ , which take the form (derived in exercise 15.1)

$$\mathbf{G}_{\mathbf{x}} = \mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x}) + \mathbf{J}_\sigma(\mathbf{x})^\top \mathbf{J}_\sigma(\mathbf{x}). \quad (15.8)$$

From the figure, it is evident that geodesics under this metric follow the circular structure of the data, which suggests that the metric does a better job of truthfully respecting the data.

*Why does the uncertainty make such a difference?* The difference between the panels of Fig. 15.2 is the choice of metric, and the difference between these is the term  $\mathbf{J}_\sigma(\mathbf{x})^\top \mathbf{J}_\sigma(\mathbf{x})$  in Eq. 15.8. Clearly, this makes an important improvement to the metric. In this particular example,  $\sigma$  behaves such that it produces smaller values near the observed data than it does far away from the data. This rather natural behavior implies that the Jacobian of  $\sigma$  takes large values when  $\sigma$  changes from small to large values. Curves that pass through such regions then

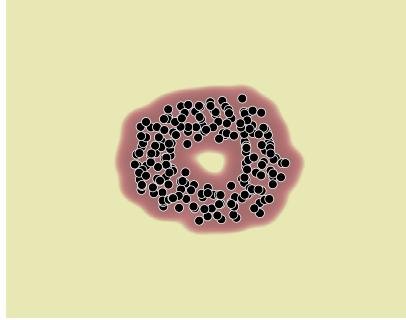


FIG. 15.3. *The summed per-dimension variance plotted in the latent representation space, i.e.,  $\sum_{d=1}^D \sigma^2(\mathbf{x})$ , as background. Dots correspond to latent data representations.*

become rather long, and the shortest curves (geodesics) naturally try to avoid these regions. In the right panel of Fig. 15.2, the expensive regions appear as the yellow ‘walls’ that surround the data. On randomly projected manifolds, uncertainty, thus, pushes geodesics toward the observed data.

### 15.5. Statistical submanifolds

The idea of *projected manifolds* rely crucially on the the decomposition in Eq. 15.6, which we here repeat

$$\mathbf{y} = f(\mathbf{x}) + \epsilon \sigma(\mathbf{x}). \quad (15.9)$$

The Gaussian distribution can always be decomposed as a sum of a deterministic mean and a stochastic deviation, but only a few other distributions share this decomposability. Most likelihoods  $p(\mathbf{y}|\mathbf{x})$  used for generative models, in particular, cannot be decomposed in this way.<sup>5</sup> This prevents us from applying the idea of randomly projected manifolds.

So far, our perspective has been that we have a map  $f$  from the latent representations into the space of *observations*. Our studied

---

<sup>5</sup>e.g., the *categorical* or *multinomial* distribution cannot be as simply decomposed.

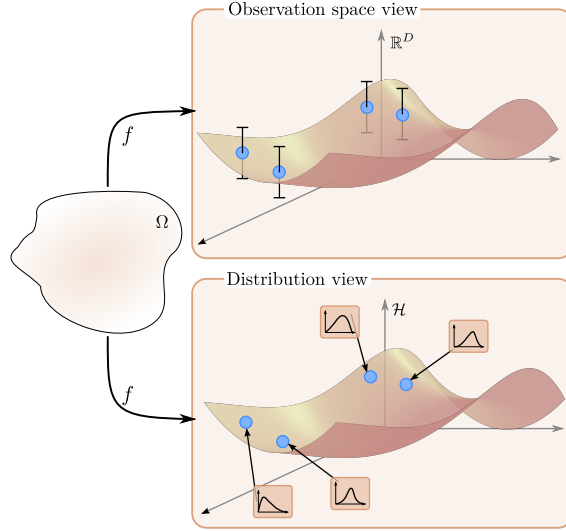


FIG. 15.4. *Two different perspectives leading to different latent geometries. In both views, we have latent representations in  $\Omega$  that are mapped through a function  $f$ . The observation space view (top) maps latent representations to the space of observations. To account for noise, this mapping must be stochastic, forcing us to consider projected manifolds. In the distribution view (bottom), we map to the abstract space of distributions  $p(\mathbf{y}|\mathbf{x})$ . In this space, one point corresponds to a distribution, thereby alleviating the need to model data noise.*

geometries then appear by pulling the metric from the observation space into the latent representation space.

We can construct an alternative geometry by foregoing this view and instead thinking of a map between latent representations  $\mathbf{x}$  and conditional distributions  $p(\mathbf{y}|\mathbf{x})$ . For example, the Gaussian model (15.5) can be viewed as

$$f : \mathbf{x} \mapsto \mathcal{N}(\mathbf{y}|f(\mathbf{x}), \text{diag}(\sigma^2(\mathbf{x}))). \quad (15.10)$$

Figure 15.4 illustrates the conceptual difference.



## 15.6. Pulling back Fisher–Rao

The information geometry studied in Chapter 14 provides us with the tools to bring the geometry of distributions into the latent representation space. Recall that we consider a manifold spanned by  $f$  within the space of distributions, i.e.

$$f : \Omega \rightarrow \mathcal{H}, \quad (15.11)$$

$$f : \mathbf{x} \mapsto p_\theta(\mathbf{y}|\mathbf{x}). \quad (15.12)$$

We have seen that we can equip the space of distributions  $\mathcal{H}$  with the Fisher–Rao metric  $\mathbf{F}_\mathbf{y}$ . We can then bring this to the latent representation space as

$$\mathbf{G}_\mathbf{x} = \mathbf{J}_\mathbf{x}^\top \mathbf{F}_{f(\mathbf{x})} \mathbf{J}_\mathbf{x}. \quad (15.13)$$

In practice, the Fisher–Rao metric can be computationally expensive to evaluate, especially when the conditional distribution  $p_\theta(\mathbf{y}|\mathbf{x})$  has a high-dimensional parameter vector  $\theta$ . Fortunately, many computations can be carried out without instantiating the metric.

The energy of a curve, which is used for computing geodesics, can be approximated using only evaluations of KL-divergences,

$$\mathcal{E}(c) = \int_0^1 \dot{c}_t^\top \mathbf{J}_{c_t}^\top \mathbf{F}_{c_t} \mathbf{J}_{c_t} \dot{c}_t dt \quad (15.14)$$

$$\lim_{N \rightarrow \infty} \sum_{i=0}^N \text{KL}(f(c(t_i)) \parallel f(c(t_{i+1}))) \quad (15.15)$$

$$\approx \sum_{i=0}^N \text{KL}(f(c(t_i)) \parallel f(c(t_{i+1}))), \quad (15.16)$$

which, for most distributions, can be done efficiently.

Unlike the randomly projected manifolds (15.8), this metric applies to practically all models as it merely relies on the availability of a local KL-divergence.

## *Exercises*

### *Exercise 15.1: Metric of randomly projected manifold.*

Consider the manifold spanned by the mapping  $f : \mathbb{R}^d \rightarrow \mathbb{R}^{2D}$  defined as

$$h(\mathbf{x}) = \begin{pmatrix} f(\mathbf{x}) \\ \sigma(\mathbf{x}). \end{pmatrix} \quad (15.17)$$

1. *Derive the pull-back metric of this manifold.*

### *Exercise 15.2: Gaussian variational autoencoder.*

Consider the MNIST dataset.

1. *Train a variational autoencoder with a Gaussian likelihood [26] on the MNIST data.*
2. *Implement a computer program to compute geodesics under the randomly projected metric (15.8) with the model's decoder.*
3. *Implement a computer program to compute geodesics under the pull-back of the Fisher–Rao metric associated with the likelihood.*
4. *How does the variance network  $\sigma^2(\mathbf{x})$  influence results?*

### *Exercise 15.3: Bernoulli variational autoencoder.*

Consider the MNIST dataset.

1. *Train a variational autoencoder with a Bernoulli likelihood [26] on the MNIST data.*
2. *Implement a computer program to compute geodesics under the pull-back of the Fisher–Rao metric associated with the likelihood.*
3. *If you did the previous exercise, how do the results compare?*

## Chapter 16

### *Expected Gaussian manifolds*



*In which great expectations  
aren't all that great.*

We have considered manifolds expressed by functions  $f$ . When these functions are estimated from data, they should, however, inherently be treated as stochastic objects. It then makes sense to talk of a *random function*. For a parametric function, we can make a random version of said function by replacing its parameters with random variables, e.g.,

$$f(\mathbf{x}) = a\mathbf{x}^2 + b\mathbf{x} + c \quad (16.1)$$

becomes random if we assign a density to the parameters  $(a, b, c)$ . The figure in the margin shows an example where  $(a, b, c)$  is normally distributed. Extending this view to neural networks gives rise to *Bayesian neural networks*, which is one way to be stochastic about large parametrized models.

Statistically, the most common way to talk about random functions is through the language of stochastic processes. This is beyond the scope of this text, but we briefly discuss the special case of *Gaussian processes* and the manifolds they span.

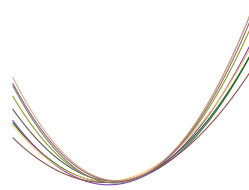


FIG. 16.1. A second-order polynomial with normally distributed coefficients.

#### *16.1. Gaussian polynomials*

What matters most about a function is the value it takes for a given input. Random functions are then naturally described by considering the distribution of function values for a given input, i.e., we write

$$\mathbf{y} \sim p(\mathbf{y}|\mathbf{x}) \quad (16.2)$$

instead of  $\mathbf{y} = f(\mathbf{x})$ . For example, if we choose independent normally distributed coefficients for the polynomial in Eq. 16.1,

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \bar{a} \\ \bar{b} \\ \bar{c} \end{pmatrix}, \begin{pmatrix} \sigma_a^2 & 0 & 0 \\ 0 & \sigma_b^2 & 0 \\ 0 & 0 & \sigma_c^2 \end{pmatrix} \right) \quad (16.3)$$

we get that

$$p(\mathbf{y}|\mathbf{x}) = \begin{pmatrix} a \\ b \\ c \end{pmatrix}^\top \begin{pmatrix} \mathbf{x}^2 \\ \mathbf{x} \\ 1 \end{pmatrix} \quad (16.4)$$

$$= \mathcal{N}(\mathbf{y} \mid \bar{a}\mathbf{x}^2 + \bar{b}\mathbf{x} + \bar{c}, \sigma_a^2\mathbf{x}^4 + \sigma_b^2\mathbf{x}^2 + \sigma_c^2). \quad (16.5)$$

To derive this, we used that for any normally distributed vector  $\mathbf{v}$

$$\mathbf{v} \sim \mathcal{N}(\mu, \Sigma) \quad \Rightarrow \quad \mathbf{A}\mathbf{v} \sim \mathcal{N}(\mathbf{A}\mu, \mathbf{A}^\top \Sigma \mathbf{A}), \quad (16.6)$$

which holds for any matrix  $\mathbf{A}$  of suitable dimension. We will use this property of normal distributions throughout the chapter, so keep it always fresh in mind.

The random function in Eq. 16.5 can be thought of as defining a *mean function*  $\mu(\mathbf{x}) = \bar{a}\mathbf{x}^2 + \bar{b}\mathbf{x} + \bar{c}$  and a *variance function*  $\sigma^2(\mathbf{x}) = \sigma_a^2\mathbf{x}^4 + \sigma_b^2\mathbf{x}^2 + \sigma_c^2$ , which is illustrated in the left panel of Fig. 16.2. While intuitively appealing, this view is, however, lacking.

Sampling function values from Eq. 16.5 corresponds to evaluating the mean function and adding Gaussian noise (scaled according to

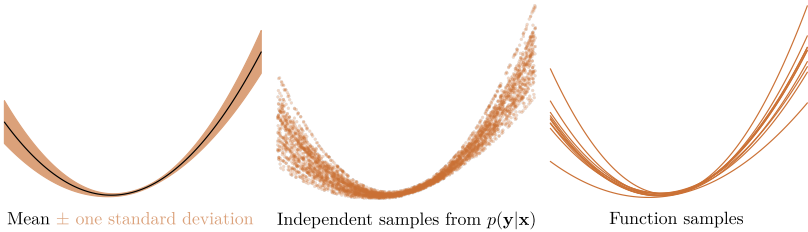
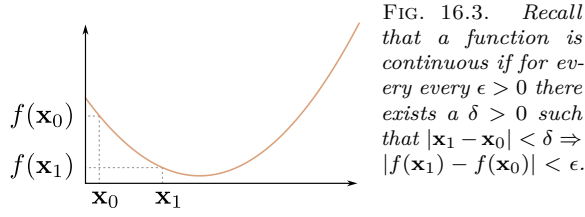


FIG. 16.2. Normally distributed polynomial coefficients lead to polynomial mean and variance functions (left panel). Independently sampling  $p(\mathbf{y}|\mathbf{x})$  gives discontinuous functions (center panel), unlike randomly sampled coefficients (right panel).



the variance function), i.e.,  $\mathbf{y} = \mu(\mathbf{x}) + \sigma(\mathbf{x})\epsilon$ , where  $\epsilon \sim \mathcal{N}(0, 1)$ . This does *not* result in smooth function samples as the added noise is not continuous with respect to  $\mathbf{x}$ . At the same time, we note that sampling the polynomial coefficients yields smooth functions that can be evaluated for any inputs  $\mathbf{x}$ . For our purposes, continuity and smoothness is important as it is required to define metrics.

The root of this discrepancy is *covariance* (or lack thereof). Consider evaluating a smooth function at two nearby input points; then, the associated function values should also be near each other. Similarly, a smooth stochastic function should produce highly correlated values of nearby inputs.

Returning to the stochastic polynomial from Eq. 16.5, we can write the *joint distribution* of two function values as

$$p(\mathbf{y}_1, \mathbf{y}_0 | \mathbf{x}_1, \mathbf{x}_0) = \begin{pmatrix} a \\ b \\ c \end{pmatrix}^T \begin{pmatrix} \mathbf{x}_1^2 & \mathbf{x}_0^2 \\ \mathbf{x}_1 & \mathbf{x}_0 \\ 1 & 1 \end{pmatrix} \quad (16.7)$$

$$= \mathcal{N}(\mathbf{y}_1, \mathbf{y}_0 | \mu, \Sigma), \quad (16.8)$$

where (using Eq. 16.6 as before)

$$\mu = \begin{pmatrix} \bar{a}\mathbf{x}_1^2 + \bar{b}\mathbf{x}_1 + \bar{c} \\ \bar{a}\mathbf{x}_0^2 + \bar{b}\mathbf{x}_0 + \bar{c} \end{pmatrix} \quad (16.9)$$

$$\Sigma = \begin{pmatrix} \sigma_a^2\mathbf{x}_1^4 + \sigma_b^2\mathbf{x}_1^2 + \sigma_c^2 & \sigma_a^2\mathbf{x}_1^2\mathbf{x}_0^2 + \sigma_b^2\mathbf{x}_1\mathbf{x}_0 + \sigma_c^2 \\ \sigma_a^2\mathbf{x}_1^2\mathbf{x}_0^2 + \sigma_b^2\mathbf{x}_1\mathbf{x}_0 + \sigma_c^2 & \sigma_a^2\mathbf{x}_0^4 + \sigma_b^2\mathbf{x}_0^2 + \sigma_c^2 \end{pmatrix}. \quad (16.10)$$

Since the off-diagonal elements of the covariance  $\Sigma$  are nonzero, the function values  $\mathbf{y}_1$  and  $\mathbf{y}_0$  affect each other (they are correlated). We will later see that this allows us to talk about the smoothness of the

stochastic function. But first, we both simplify and generalize the notation.

## 16.2. Gaussian processes

The covariance associated with the stochastic polynomial (Eq. 16.10) is a rather lengthy expression considering the model's simplicity. However, we note that all entries in this covariance matrix can be written as

$$[\Sigma]_{ij} = \sigma_a^2 \mathbf{x}_i^2 \mathbf{x}_j^2 + \sigma_b^2 \mathbf{x}_i \mathbf{x}_j + \sigma_c^2, \quad (16.11)$$

which allows us to introduce the short notation  $[\Sigma]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ . We call  $k$  a *covariance function* because it parametrizes the covariance between arbitrary points.

We then say that the stochastic polynomial (16.5) is a *Gaussian process* with mean function  $\mu$  and covariance function  $k$ , and use the short-hand notation  $\mathbf{y} \sim \mathcal{GP}(\mu, k)$ .

Formally, a Gaussian process  $\mathcal{GP}(\mu, k)$  is a collection of random variables  $\{\mathbf{y}_i\}$  such that any finite subset thereof follows a normal distribution,

$$\begin{pmatrix} \mathbf{y}_0 \\ \vdots \\ \mathbf{y}_n \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \mu(\mathbf{x}_0) \\ \vdots \\ \mu(\mathbf{x}_n) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \right). \quad (16.12)$$

In most uses of Gaussian processes, we specify the covariance function directly rather than, e.g., starting with a polynomial. The only requirement of this covariance function is that any resulting covariance matrix must be positive definite (or semi-definite, depending on how strict one wants to be).

## 16.3. Gaussian processes models

Most often, Gaussian processes are used to specify priors over functions of interest. Given data, we then seek the data-conditioned prediction for new inputs. Let's consider the case of *regression*. Here we are given input-output pairs  $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$  and seek a function  $f$  such that

$\mathbf{y}_n \approx f(\mathbf{x}_n)$ . We can now assume a Gaussian process prior over the unknown function  $f$ ,

$$f \sim \mathcal{GP}(\mu, k). \quad (16.13)$$

Our target is then the conditional distribution  $p(\mathbf{y}|\mathbf{x}, \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N)$ , where  $\mathbf{x}$  is the point where we seek to evaluate  $f$  and  $\mathbf{y}$  is the associated function value.

Arriving at this condition is merely a matter of applying the *Gaussian conditioning* identity

$$\begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{21} \\ \Sigma_{12} & \Sigma_{22} \end{pmatrix} \right) \quad \Rightarrow \quad (16.14)$$

$$\mathbf{v}_2 | \mathbf{v}_1 \sim \mathcal{N} \left( \mu_2 + \Sigma_{12} \Sigma_{11}^{-1} (\mathbf{v}_1 - \mu_1), \Sigma_{22} - \Sigma_{12} \Sigma_{11}^{-1} \Sigma_{21} \right). \quad (16.15)$$

To apply this identity, we first write the *joint distribution* of  $(\mathbf{x}, \mathbf{y})$  and  $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ ,

$$\begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \mu(\mathbf{x}_1) \\ \vdots \\ \mu(\mathbf{x}_N) \\ \mu(\mathbf{x}) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_1) & | & k(\mathbf{x}, \mathbf{x}_1) \\ \vdots & \ddots & \vdots & | & \vdots \\ k(\mathbf{x}_1, \mathbf{x}_N) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) & | & k(\mathbf{x}, \mathbf{x}_N) \\ \hline k(\mathbf{x}_1, \mathbf{x}) & \cdots & k(\mathbf{x}_N, \mathbf{x}) & | & k(\mathbf{x}, \mathbf{x}) \end{pmatrix} \right)$$

where the vertical and horizontal lines split the equation into terms corresponding to Eq. 16.14. Consequently, the conditioning used for regression results in a normal distribution. This implies that the conditional is also a Gaussian process, which can be written as

$$\mathbf{y} | \mathbf{x}, \mathbf{x}_{1:N}, \mathbf{y}_{1:N} \sim \mathcal{GP}(\hat{\mu}, \hat{k}), \quad (16.16)$$

where the cumbersome expression for the mean and covariance functions follows from Eq. 16.15.

Figure 16.4 illustrates how increasing the number of conditioning points results in a more certain estimate of the underlying function. Here, the prior Gaussian process is chosen to have a mean function which is constantly zero. The covariance function can be chosen as any positive definite function, and we here picked the so-called *squared*

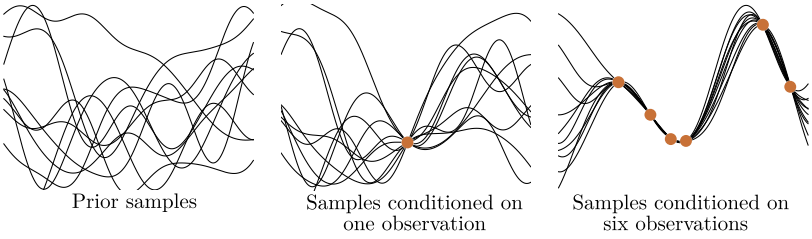


FIG. 16.4. *Samples from a Gaussian process conditioned on zero, one and six observations. As the number of conditioning points increases, the estimated function becomes more certain.*

exponential covariance function<sup>1</sup>

$$k(\mathbf{x}_i, \mathbf{x}_j) = \theta \exp(-\lambda \|\mathbf{x}_i - \mathbf{x}_j\|^2). \quad (16.17)$$

Here  $\theta$  and  $\lambda$  are parameters that change the behavior of the Gaussian process. There is a rich selection of similar functions available in the literature, and we refer to Rasmussen and Williams [36] for a more in-depth discussion.

## 16.4. Derivatives of Gaussian processes

We aim to understand the geometry of manifolds spanned by Gaussian processes, i.e., we consider the ‘usual’ setup where the manifold is given by

$$\mathcal{M} = f(\Omega), \quad \Omega \subset \mathbb{R}^d. \quad (16.18)$$

Except now  $f$  is not deterministically given, but rather follows a Gaussian process  $f \sim \mathcal{GP}(\mu, k)$ . For this to work out, we know that we will require access to the derivatives of  $f$ , e.g., the Jacobian determines the metric. But how do we talk about derivatives of stochastic processes?

Recall that the derivative of a function can be defined as

$$\frac{df}{d\mathbf{x}} = \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h} = \lim_{h \rightarrow 0} \Delta(h). \quad (16.19)$$

<sup>1</sup>This covariance function is also known as the *Gaussian* covariance or the *exponentiated quadratic*.



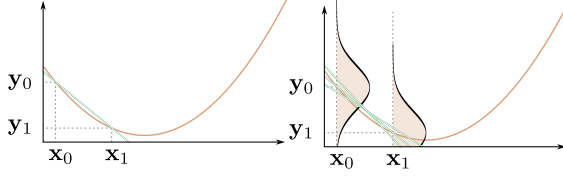


FIG. 16.5. Left: the deterministic derivative is the slope of a local linear approximation (green). Right: when function values are random, so is the derivative.

We can follow the same strategy in the stochastic case and first define the difference

$$\Delta(h) = \begin{pmatrix} 1/h \\ -1/h \end{pmatrix}^\top \begin{pmatrix} f(\mathbf{x} + h) \\ f(\mathbf{x}) \end{pmatrix}. \quad (16.20)$$

When  $f \sim \mathcal{GP}(\mu, k)$ , we have that

$$\begin{pmatrix} f(\mathbf{x} + h) \\ f(\mathbf{x}) \end{pmatrix} \sim \mathcal{N}(\mathbf{m}, \Sigma), \quad (16.21)$$

where

$$\mathbf{m} = \begin{pmatrix} \mu(\mathbf{x} + h) \\ \mu(\mathbf{x}) \end{pmatrix}, \quad (16.22)$$

$$\Sigma = \begin{pmatrix} k(\mathbf{x} + h, \mathbf{x} + h) & k(\mathbf{x} + h, \mathbf{x}) \\ k(\mathbf{x} + h, \mathbf{x}) & k(\mathbf{x}, \mathbf{x}) \end{pmatrix}, \quad (16.23)$$

where we have exploited the symmetry  $k(\mathbf{x} + h, \mathbf{x}) = k(\mathbf{x}, \mathbf{x} + h)$ . We can then compute the difference using Eq. 16.6 as

$$\Delta(h) \sim \mathcal{N}\left(\begin{pmatrix} 1/h \\ -1/h \end{pmatrix}^\top \mathbf{m}, \begin{pmatrix} 1/h \\ -1/h \end{pmatrix}^\top \Sigma \begin{pmatrix} 1/h \\ -1/h \end{pmatrix}\right) \quad (16.24)$$

$$= \mathcal{N}(\Delta_\mu(h), \Delta_k^2(h)), \quad (16.25)$$

where we have defined

$$\Delta_\mu(h) = \frac{\mu(\mathbf{x} + h) - \mu(\mathbf{x})}{h}, \quad (16.26)$$

$$\Delta_k^2(h) = \frac{k(\mathbf{x} + h, \mathbf{x} + h) + k(\mathbf{x}, \mathbf{x}) - 2k(\mathbf{x} + h, \mathbf{x})}{h^2}. \quad (16.27)$$

Now we can take the limit  $h \rightarrow 0$ , which gives

$$\lim_{h \rightarrow 0} \Delta_\mu(h) = \frac{d\mu(\mathbf{x})}{d\mathbf{x}} \quad (16.28)$$

$$\lim_{h \rightarrow 0} \Delta_k^2(h) = \frac{d^2 k(\mathbf{x}_1, \mathbf{x}_2)}{d\mathbf{x}_1 d\mathbf{x}_2}. \quad (16.29)$$

Intuitively, it is natural that the covariance is *twice* differentiated as it describes how *pairs* of points interact.

The conclusion is, thus, rather intuitive: the derivative of a Gaussian process is another Gaussian process with the once-differentiated mean function and the twice-differentiated covariance function,

$$\frac{df}{d\mathbf{x}} \sim \mathcal{GP}\left(\frac{d\mu(\mathbf{x})}{d\mathbf{x}}, \frac{d^2 k(\mathbf{x}_1, \mathbf{x}_2)}{d\mathbf{x}_1 d\mathbf{x}_2}\right). \quad (16.30)$$

Recall Eq. 16.6 saying that  $\mathbf{v} \sim \mathcal{N}(\mu, \Sigma) \Rightarrow \mathbf{A}\mathbf{v} \sim \mathcal{N}(\mathbf{A}\mu, \mathbf{A}^\top \Sigma \mathbf{A})$  for any suitably-sized matrix  $\mathbf{A}$ . Eq. 16.30 can be seen as a generalization in the sense that differentiation is a linear operator, and we can heuristically write

$$\mathbf{v} \sim \mathcal{GP}(\mu, k) \quad \Rightarrow \quad \partial \mathbf{v} \sim \mathcal{GP}(\partial \mu, \partial k \partial), \quad (16.31)$$

though we note that this is imprecise and should only be used as an intuitive guide.

## 16.5. Gaussian process manifolds

Now we are finally ready to discuss the geometry of manifolds spanned by Gaussian processes. To simplify notation, we write  $f$  component-wise as

$$f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_D(\mathbf{x}) \end{pmatrix}, \quad (16.32)$$

and then we assume that each function  $f_i \sim \mathcal{GP}(\mu_i, k_i)$  is conditionally independent of each other, i.e.,  $f_i(\mathbf{x}) \perp f_j(\mathbf{x})$  for  $i \neq j$ . This is a

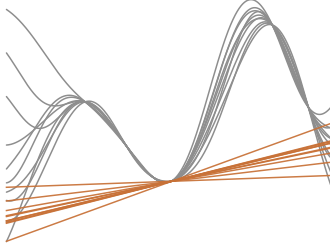


FIG. 16.6. *Samples from a Gaussian process (gray) alongside the corresponding derivative process (brown).*

somewhat crude assumption, but it drastically limits notation, as we can disregard how different functions covary.

We have previously seen that constructing tangent spaces is a good starting point. These are spanned by the Jacobian of  $f$ , which we write as

$$\mathbf{J}_{\mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial f_D}{\partial \mathbf{x}} \end{pmatrix} \in \mathbb{R}^{D \times d}, \quad (16.33)$$

where each row follows a Gaussian process

$$\frac{\partial f_i}{\partial \mathbf{x}} \sim \mathcal{GP} \left( \frac{\partial \mu_i}{\partial \mathbf{x}}, \frac{\partial^2 k_i}{\partial^2 \mathbf{x}} \right). \quad (16.34)$$

Here we use a slightly condensed notation; see Eq. 16.30 for the full notational gory.

## 16.6. Tangent vectors

Since the Jacobian of a Gaussian process is another Gaussian process, we see that the tangent space of our random manifold  $\mathcal{M}$  is a Gaussian process. Given a (deterministic) latent position  $\mathbf{x} \in \mathbb{R}^d$  and an (infinitesimal) displacement  $\Delta \mathbf{x}$ , the corresponding tangent vector to

the manifold is then

$$\mathbf{J}_\mathbf{x}\Delta\mathbf{x} + \mu(\mathbf{x}) \sim \mathcal{N}\left(\mu(\mathbf{x}) + \sum_{i=1}^D \frac{\partial\mu_i(\mathbf{x})}{\partial\mathbf{x}} [\Delta\mathbf{x}]_i, \sum_{i=1}^D \Delta\mathbf{x}^\top \frac{\partial^2 k_i}{\partial^2 \mathbf{x}} \Delta\mathbf{x}\right). \quad (16.35)$$

Figure 16.6 shows samples from a Gaussian process and the corresponding tangent spaces. The distribution of these tangents is then given by Eq. 16.35.

From the distribution of tangent vectors, we can now ask *how long is  $\Delta\mathbf{x}$* ? As a starting point, we will write the local squared norm of a displacement

$$\|\Delta\mathbf{x}\|_{\mathbf{G}_\mathbf{x}}^2 = \|\mathbf{J}_\mathbf{x}\Delta\mathbf{x}\|^2. \quad (16.36)$$

This squared length is a random variable. To characterize this variable, we first compute its mean by recalling that

$$\mathbb{E}[\mathbf{v}^\top \mathbf{v}] = \mathbb{E}[\mathbf{v}]^\top \mathbb{E}[\mathbf{v}] + \text{Tr}(\text{cov}(\mathbf{v})). \quad (16.37)$$

Choosing  $\mathbf{v} = \mathbf{J}_\mathbf{x}\Delta\mathbf{x}$  then gives

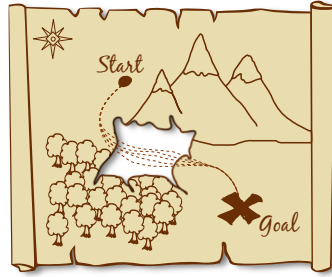
$$\begin{aligned} \mathbb{E} [\|\mathbf{J}_\mathbf{x}\Delta\mathbf{x}\|^2] &= \left\| \sum_{i=1}^D \frac{\partial\mu_i(\mathbf{x})}{\partial\mathbf{x}}^\top [\Delta\mathbf{x}]_i \right\|^2 \\ &\quad \text{(contribution of mean)} \\ &\quad + \Delta\mathbf{x}^\top \sum_{i=1}^D \text{Tr} \left( \frac{\partial^2 k_i}{\partial^2 \mathbf{x}} \right) \Delta\mathbf{x}. \\ &\quad \text{(contribution of covariance)} \end{aligned} \quad (16.38)$$

Assuming we can differentiate the mean and covariance function (which we usually can), we can evaluate the expected local squared length of a displacement  $\Delta\mathbf{x}$ .

A closer inspection of Eq. 16.38 sheds light on how uncertainty impacts geometry. In expectation, the squared norm of a tangent vector consists of two terms. The first is merely the norm of the tangent vector associated with the *mean manifold* spanned by  $\mu$ . The second

(and possibly more surprising) term is the trace of the covariance. This term is strictly positive (since the covariance is positive definite), implying that uncertainty increases local norms in expectation. We further see that local expected norms grow with the uncertainty.

This link between uncertainty and geometry has an intuitive explanation. Imagine navigating according to a map. If this map is less precise (more uncertain) in some regions, then we should not expect our navigation to be optimal, and we may take an unnecessary detour. Uncertainty gives longer routes (distances). The treasure map illustrates the situation.



### 16.7. Expected energies

The notion of expected local squared norms naturally generalizes into expectations of curve energies. Consider a latent curve  $c : [0, 1] \rightarrow \Omega \subseteq \mathbb{R}^d$ . Recall that the energy of this curve is given by

$$\mathcal{E}[c] = \int_0^1 \|\dot{c}_t\|_{\mathbf{G}_{c_t}}^2 dt \quad (16.39)$$

$$= \int_0^1 \|\mathbf{J}_{c_t} \dot{c}_t\|^2 dt. \quad (16.40)$$

This energy is stochastic, so it does not lend itself to direct minimization as we did for deterministic manifolds. A natural first choice is to look at curves of minimal *expected energy*,

$$\bar{\mathcal{E}}[c] = \mathbb{E} \int_0^1 \|\mathbf{J}_{c_t} \dot{c}_t\|^2 dt \quad (16.41)$$

$$= \int_0^1 \mathbb{E} [\|\mathbf{J}_{c_t} \dot{c}_t\|^2] dt, \quad (16.42)$$

where the integrand can be evaluated with Eq. 16.38. Using the numerical techniques from Chapter 8, we can directly minimize this expected energy.

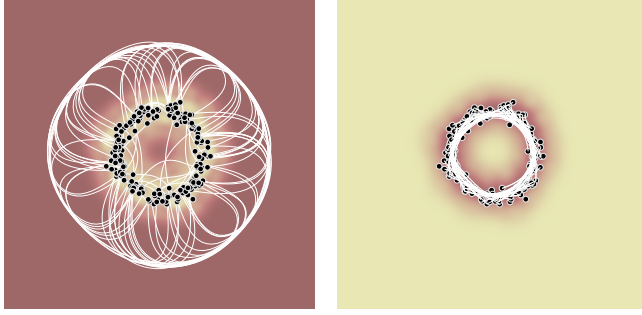


FIG. 16.7. *Example-geodesics according to the mean manifold and the expected energy. In the left panel, dark brown regions all map to the same point in  $\mathbb{R}^D$ , such that distances are zero in this region.*

Figure 16.7 shows example geodesics in the latent space of a manifold estimated from data with a circular structure. The top panel shows geodesics on the *mean manifold* spanned by  $\mu$ . Due to how the Gaussian process is estimated,  $\mu$  extrapolates to zero. This implies that all latent regions far from the data map to the origin of  $\mathbb{R}^D$ . Geodesics then have minimal energy if going through this origin, as all data is easily connected there. In contrast, the bottom panel shows geodesics according to the *expected energy*. These naturally avoid regions of high uncertainty as these have high expected energy (due to Eq. 16.38).

## 16.8. Expected metrics

But which manifold are we actually working on when minimizing expected energies? Since

$$\|\mathbf{J}_{c_t} \dot{c}_t\|^2 = \dot{c}_t^\top \mathbf{J}_{c_t}^\top \mathbf{J}_{c_t} \dot{c}_t, \quad (16.43)$$

we can write the expected energy as

$$\bar{\mathcal{E}}[c] = \int_0^1 \mathbb{E} [\|\mathbf{J}_{c_t} \dot{c}_t\|^2] dt \quad (16.44)$$

$$= \int_0^1 \mathbb{E} [\dot{c}_t^\top \mathbf{J}_{c_t}^\top \mathbf{J}_{c_t} \dot{c}_t] dt \quad (16.45)$$

$$= \int_0^1 \dot{c}_t^\top \mathbb{E} [\mathbf{J}_{c_t}^\top \mathbf{J}_{c_t}] \dot{c}_t dt. \quad (16.46)$$

This is merely the usual curve energy under the *expected metric* given by

$$\bar{\mathbf{G}} = \mathbb{E} [\mathbf{G}] = \mathbb{E} [\mathbf{J}^\top \mathbf{J}]. \quad (16.47)$$

Working with the expected energy, thus, corresponds to picking the expected metric. Since this is deterministic, the tools from previous chapters directly apply.

To gain a bit more insight into the expected metric, we can write it in terms of the underlying Gaussian process. For convenience, we repeat that the rows of the Jacobian are assumed to follow a Gaussian process,

$$\frac{\partial f_i}{\partial \mathbf{x}} \sim \mathcal{GP} \left( \frac{\partial \mu_i}{\partial \mathbf{x}}, \frac{\partial^2 k_i}{\partial^2 \mathbf{x}} \right). \quad (16.48)$$

Since  $\mathbf{J}^\top \mathbf{J} = \sum_i \frac{\partial f_i}{\partial \mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}}^\top$ , we can write the expected metric as

$$\bar{\mathbf{G}} = \sum_{i=1}^D \mathbb{E} \left[ \frac{\partial f_i}{\partial \mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}}^\top \right] \quad (16.49)$$

$$= \sum_{i=1}^D \frac{\partial \mu_i}{\partial \mathbf{x}} \frac{\partial \mu_i}{\partial \mathbf{x}}^\top + \sum_{i=1}^D \frac{\partial^2 k_i}{\partial^2 \mathbf{x}}. \quad (16.50)$$

Just as with the expected local norm (16.38), the expected metric decomposes into two terms. The first is the metric of the mean manifold spanned by  $\mu$ , while the second term depends on the covariance of the Jacobian. As the uncertainty of the Jacobian grows, we see that the expected metric becomes larger. This implies that geodesics are naturally attracted to regions where there is little uncertainty. In practice, Gaussian process models tend to have low uncertainty near training data, such that geodesics will be drawn towards the data on which the model is trained.

In most practical scenarios, it is highly beneficial that geodesics stay in regions of high data density, as this gives more trustworthy paths between points. In this regard, the expected metric is a significantly better choice than the metric of the manifold spanned by the mean function  $\mu$ .

## 16.9. Expected length

In Chapter 7 we saw that any energy-minimizing curve was also a length-minimizing. This result allows us to compute geodesics without worrying about curve parametrizations. Unfortunately, this is not the case when taking expectations.

Similarly to Chapter 7, we define two curves  $u_t = \mathbb{E}[\|\dot{c}_t\|]$  and  $v_t = 1$ . Again, we use Cauchy-Schwartz's inequality  $|\langle u, v \rangle|^2 \leq \|u\|^2 \cdot \|v\|^2$  to give

$$\left( \int_0^1 \mathbb{E}[\|\dot{c}_t\|] dt \right)^2 \leq \int_0^1 \mathbb{E}[\|\dot{c}_t\|^2] dt \cdot \int_0^1 dt \quad (16.51)$$

$$= \int_0^1 \mathbb{E}[\|\dot{c}_t\|^2] dt. \quad (16.52)$$

We then write the *expected length* as

$$\bar{\mathcal{L}}[c] = \mathbb{E} \left[ \int_0^1 \|\dot{c}_t\| dt \right] = \int_0^1 \mathbb{E}[\|\dot{c}_t\|] dt \quad (16.53)$$

such that

$$\int_0^1 \mathbb{E}[\|\dot{c}_t\|^2] dt \geq \bar{\mathcal{L}}^2[c]. \quad (16.54)$$

Equality is achieved when  $u_t$  and  $v_t$  are parallel, that is when  $\mathbb{E}[\|\dot{c}_t\|]$  is constant. We can always reparametrize  $c$  to have a constant expected speed and achieve equality.

We can leverage the fact that the variance of  $\mathbf{x}$  decomposes as

$$\text{var}(\mathbf{x}) = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])^2] = \mathbb{E}[\mathbf{x}^2] - \mathbb{E}[\mathbf{x}]^2 \quad (16.55)$$

to see that

$$\int_0^1 \mathbb{E}[\|\dot{c}_t\|^2] dt = \int_0^1 \mathbb{E}[\|\dot{c}_t\|^2] dt - \int_0^1 \text{var}(\|\dot{c}_t\|) dt \quad (16.56)$$

$$= \bar{\mathcal{E}}[c] - \int_0^1 \text{var}(\|\dot{c}_t\|) dt. \quad (16.57)$$



If we assume that  $c$  has been parametrized to have constant expected speed, we then get

$$\bar{\mathcal{E}}[c] = \bar{\mathcal{L}}^2[c] + \int_0^1 \text{var}(\|\dot{c}_t\|) dt. \quad (16.58)$$

Minimizing expected curve energy, thus, does not always minimize the expected curve length. Rather, this balances the minimization of expected curve length and the minimization of curve variance.

## Exercises

### Exercise 16.1: Expectations of products.

When tangent vectors are random, we must concern ourselves with their products to determine lengths and angles. Here, covariances play a crucial role. Recall that the covariance between two random vectors is given by

$$\text{cov}(\mathbf{v}, \mathbf{u}) = \mathbb{E}[(\mathbf{v} - \mathbb{E}[\mathbf{v}])(\mathbf{u} - \mathbb{E}[\mathbf{u}])^\top]. \quad (16.59)$$

1. Show that

$$\mathbb{E}[\mathbf{v}\mathbf{u}^\top] = \mathbb{E}[\mathbf{v}]\mathbb{E}[\mathbf{u}]^\top + \text{cov}(\mathbf{v}, \mathbf{u}), \quad (16.60)$$

$$\mathbb{E}[\mathbf{v}^\top \mathbf{u}] = \mathbb{E}[\mathbf{v}]^\top \mathbb{E}[\mathbf{u}] + \text{cov}(\mathbf{v}^\top, \mathbf{u}^\top), \quad (16.61)$$

$$\mathbb{E}[\|\mathbf{v}\|^2] = \mathbb{E}[\mathbf{v}]^\top \mathbb{E}[\mathbf{v}] + \text{var}(\mathbf{v}). \quad (16.62)$$

### Exercise 16.2: Tangential norms.

Let the stochastic Jacobian  $\mathbf{J} \in \mathbb{R}^{D \times d}$  be given by

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_1^\top \\ \vdots \\ \mathbf{J}_D^\top \end{pmatrix}, \quad (16.70)$$

where the rows  $\mathbf{J}_i \sim \mathcal{N}(\mu_i, \Sigma_i)$  are independent. Further, consider two displacements  $\Delta \mathbf{x}_1, \Delta \mathbf{x}_2 \in \mathbb{R}^d$ .

1. What is the distribution of  $\mathbf{J}_i^\top \Delta \mathbf{x}_1$ ?

2. Show that

$$\mathbb{E} [\|\mathbf{J}\Delta_1\|^2] = \sum_{i=1}^D (\mu_i^\top \Delta \mathbf{x}_1)^2 + \Delta \mathbf{x}_1^\top \Sigma_i \Delta \mathbf{x}_1. \quad (16.71)$$

3. Show that

$$\mathbb{E} [(\mathbf{J}\Delta_1)^\top (\mathbf{J}\Delta_2)] = \sum_{i=1}^D (\mu_i^\top \Delta \mathbf{x}_1)(\mu_i^\top \Delta \mathbf{x}_2) + \Delta \mathbf{x}_1^\top \Sigma_i \Delta \mathbf{x}_2. \quad (16.72)$$

*Exercise 16.3: Gaussian process latent variable models.*

Consider the MNIST dataset.

1. Implement a Gaussian process latent variable model [30] on a subset of the 1-digit observations.
2. Implement a computer program to compute geodesics under the expected metric (16.47) associated with the model.
3. Implement a computer program that samples a single decoder function from the Gaussian process. Thereafter, compute geodesics with respect to the one sample. How does this compare to the expected metric?

## Chapter 17

### *Spacetime manifolds*



*In which time goes by.*

We have primarily focused on low-dimensional manifolds immersed into a high-dimensional observation space ( $d \ll D$ ). A broad class of generative models, however, do not follow this structure. Instead, they choose  $d = D$  such that the latent space and the observation space have the same dimensionality. There is, thus, no notion of an immersed submanifold in  $\mathbb{R}^D$ , and geodesics become straight lines if we follow the previous constructions.

In these models, the 'decoder' mapping from latent to observation is expressed as a *stochastic differential equation* (SDE). This SDE introduces a time parameter that, informally, interpolates between the latent and the observation space. This additional time dimension allows us to construct a notion of *spacetime geometries*. But first, let's dig into the SDEs...

#### *17.1. Denoising SDEs*

Consider a generative model that starts from pure  $D$ -dimensional Gaussian noise  $\mathcal{N}(\mathbf{0}, \mathbf{I})$  and transforms this into a synthetic observation by some stochastic process. We can engage with such an unknown process by looking at the much simpler process that transforms any distribution into pure Gaussian noise.

Deviating slightly from previous notation, we let  $\mathbf{y}_0 \in \mathbb{R}^D$  denote an observation, and describe the transition into pure Gaussian noise as

$$p_t(\mathbf{y}_t | \mathbf{y}_0) = \mathcal{N}(\mathbf{y}_t | \alpha_t \mathbf{y}_0, \sigma_t^2 \mathbf{I}), \quad \text{for } t \in (0, T]. \quad (17.1)$$

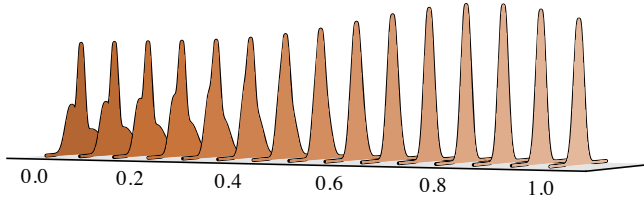


FIG. 17.1. A noising SDE gradually turns any distribution into pure Gaussian noise. A generative model can be constructed by estimating the reverse process.

Here  $\alpha_t : [0, T] \rightarrow [0, 1]$  is a monotonic function starting at one and ending at zero, while  $\sigma_t^2 = 1 - \alpha_t^2$ .<sup>1</sup> Informally, we note that at time  $t = 0$ , we have

$$p_0(\mathbf{y}_0|\mathbf{y}_0) = \mathcal{N}(\mathbf{y}_0|\mathbf{y}_0, 0 \cdot \mathbf{I}) = \delta_{\mathbf{y}_0}(\mathbf{y}_0), \quad (17.2)$$

i.e., just the empirical data distribution, and at time  $t = T$ , we get pure noise,

$$p_T(\mathbf{y}_T|\mathbf{y}_0) = \mathcal{N}(\mathbf{y}_T|0, \mathbf{I}). \quad (17.3)$$

This is illustrated in Fig. 17.1, where the time runs from  $t = 0$  to  $t = 1$ .

This construction is interesting because it is *reversible* [4]. It turns out that the reverse process is given by the so-called *denoising SDE*

$$d\mathbf{y}_t = (f_t \mathbf{y}_t - 1/2 g_t^2 \nabla \log p_t(\mathbf{y}_t)) + g_t d\bar{W}_t. \quad (17.4)$$

Here,  $f_t = d \log(\alpha_t)/dt$  and  $g_t^2 = -2\alpha_t d\alpha_t/dt$ , while  $\bar{W}_t$  is a reverse Wiener process. Empirically, the *score function*  $\nabla \log p_t(\mathbf{y}_t)$  is approximated, e.g., by a neural network, and the mapping from pure-noise  $\mathbf{y}_T$  to observation  $\mathbf{y}_0$  is realized by numerically solving the denoising SDE. This construction is known as a *diffusion model* [29].

The details of these constructions are beyond the scope of this brief book, and we refer the interested reader to existing texts, such as the book by Lai et al. [29]. From our perspective, the most important point is that the mapping from pure noise to observations is now *time indexed*, suggesting that a geometric investigation should incorporate time.

<sup>1</sup>These assumptions are overly restrictive and can be loosened [29]. We opt to be restrictive to focus on the geometric aspects.

## 17.2. The information geometry of spacetime

With the prerequisites in place, we may now investigate the natural geometry of the spacetime, i.e.,  $[0, T] \times \mathbb{R}^D$ . Any point  $\theta = (t, \mathbf{y}_t)$  in the spacetime has an associated distribution  $p_t(\mathbf{y}_t)$ . Infinitesimally, a reasonable choice of metric is then the KL-divergence,

$$\text{dist}^2(\theta, \theta + \delta) = \text{KL}(p_t(\mathbf{y}_t) \| p_{t+\delta_t}(\mathbf{y}_t + \delta_{\mathbf{y}})), \quad (17.5)$$

where we decomposed  $\delta$  as  $(\delta_t, \delta_{\mathbf{y}})$ . Using the KL-divergence infinitesimally is equivalent to endowing the spacetime with the Fisher–Rao metric (Chapter 14).

Evaluating the Fisher–Rao metric is, however, nontrivial as we only have access to  $p_t(\mathbf{y}_t)$  through the denoising SDE. Somewhat surprisingly, Karczewski et al. [24] has shown that  $p_t(\mathbf{y}_t)$  belongs to the exponential family.

**Proposition 1** (Karczewski et al. [24]). *Let  $\mathbf{y}_t$  be a noisy observation corresponding to diffusion time  $t$ , as introduced in Eq. 17.1. Then the denoising distribution can be written as*

$$p(\mathbf{y}_0 | \mathbf{y}_t) = h(\mathbf{y}_0) \exp(\eta(\mathbf{y}_t, t)^\top T(\mathbf{y}_0) - \psi(\mathbf{y}_t, t)), \quad (17.6)$$

where  $h$  is the data distribution density,  $\psi$  the log-partition function, and

$$\eta(\mathbf{y}_t, t) = \left( \frac{\alpha_t}{\sigma_t^2} \mathbf{y}_t, -\frac{\alpha_t^2}{2\sigma_t^2} \right) \quad (17.7)$$

$$T(\mathbf{y}_0) = (\mathbf{y}_0, \|\mathbf{y}_0\|^2) \quad (17.8)$$

$$\mu(\mathbf{y}_t, t) = \left( \underbrace{\mathbb{E}[\mathbf{y}_0 | \mathbf{y}_t]}_{\text{space}}, \underbrace{\sigma_t^2 / \alpha_t \text{div}_{\mathbf{y}_t} \mathbb{E}[\mathbf{y}_0 | \mathbf{y}_t] + \|\mathbb{E}[\mathbf{y}_0 | \mathbf{y}_t]\|^2}_{\text{time: } \mathbb{E}[\|\mathbf{y}_0\|^2 | \mathbf{y}_t]} \right). \quad (17.9)$$

This provides access to a way of computing the Fisher–Rao metric, curve energies and more as detailed in Eq. 14.26.

Figure 17.2 provides a one-dimensional example. In this case, the spacetime is  $[0, T] \times \mathbb{R}$ . The figure shows a geodesic interpolation between two distributions. Empirically, the geodesic gradually changes two factors of the distribution: (a) it moves the mean of one distribution

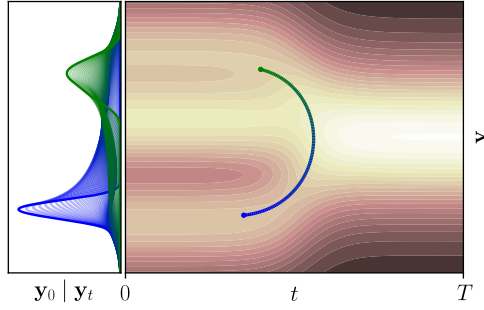


FIG. 17.2. A spacetime geodesic traverses  $[0, T] \times \mathbb{R}^D$ . Intuitively, we can view the geodesic as gradually adding noise to one endpoint until the result can be denoised into the other endpoint.

to match the other, and (b) it initially increases the variance of the initial distribution and later decreases the variance to match the target distribution. Informally, the geodesic determines how much noise must be added to one distribution before it can be denoised into the other.

## *Exercises*

### *Exercise 17.1: Proposition 1.*

The spacetime geometry crucially relies on the denoising distribution being within the exponential family.

1. *Prove Proposition 1.*

### *Exercise 17.2: Numerical implementation.*

We now consider an implementation of the spacetime geometry.

- *First generate a dataset by sampling from a one-dimensional mixture of Gaussians with two components.*
- *Train a diffusion model to fit the created data.*
- *Implement a computer program to compute spacetime geodesics.*

### *Exercise 17.3: Larger models.*

We now extend the previous exercise to deal with larger models.

- *Train a diffusion model to fit the MNIST data.*
- *Implement a computer program to compute spacetime geodesics. What is the empirical behavior of these geodesics?*

## Chapter 18

### *Parameter manifolds*



*In which everything is latent  
and the distant is near.*

So far, we have considered latent variable models that compress an observation  $\mathbf{y}$  into a latent representation  $\mathbf{x}$ . However, most of our analysis applies to a broader collection of statistical models.

Consider a predictive model given by the function  $f_\theta$  that maps an observation into the parameters of the distribution  $p(\mathbf{y}|\mathbf{x})$ . For example, a *classifier* maps the observation into a vector of class probabilities. Abstractly, we write  $f_\theta : \mathbb{R}^d \mapsto \mathcal{H}$ , where  $\mathbf{x} \in \mathbb{R}^d$  and  $\mathcal{H}$  is the space of probability distributions.<sup>1</sup>

The function parameters  $\theta \in \mathbb{R}^P$  generally have to be estimated from data, e.g., by maximizing the data log-likelihood (assuming i.i.d. data)

$$\theta_{\text{opt}} = \underset{\theta}{\operatorname{argmax}} \sum_{n=1}^N \log p(\mathbf{y}_n | \mathbf{x}_n), \quad (18.1)$$

where we explicate the role of  $f_\theta$  by changing the notation to

$$\theta_{\text{opt}} = \underset{\theta}{\operatorname{argmax}} \sum_{n=1}^N \log p(\mathbf{y}_n | f_\theta(\mathbf{x}_n)). \quad (18.2)$$

The elementary goal of this chapter is to determine the geometric structure of the parameter space  $\mathbb{R}^P$ . Such structural information can then guide methods for Bayesian approximations, optimization algorithms, and more.

---

<sup>1</sup>See Chapter 14 for an introduction to this space; in the classifier-case, this space is merely the set of positive vectors that sum to one.



## 18.1. Overparametrization

To see how the parameter space can easily take a nontrivial geometric structure, let's start with a simple example. We define the function  $f_\theta : \mathbb{R} \rightarrow \mathbb{R}$ ,

$$f_\theta(\mathbf{x}) = \theta_2 \operatorname{ReLU}(\theta_1 \mathbf{x}), \quad (18.3)$$

where  $\operatorname{ReLU}(\cdot) = \max(\cdot, 0)$  is the *rectified linear unit* activation function commonly used with neural networks (Fig. 18.1). The resulting function is piecewise linear, consisting of two pieces.

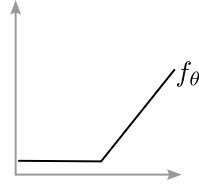


FIG. 18.1. ReLU.

We note that for any  $\alpha > 0$ , we can reparametrize the function as

$$f_\theta(\mathbf{x}) = \theta_2 \operatorname{ReLU}(\theta_1 \mathbf{x}) = \theta_2/\alpha \operatorname{ReLU}(\alpha \theta_1 \mathbf{x}). \quad (18.4)$$

The parameter vectors  $\theta = (\theta_1, \theta_2)$  and  $\theta' = (\alpha \theta_1, \theta_2/\alpha)$ , thus, generate the same function, i.e.

$$f_\theta = f_{\theta'} \quad \text{even if } \theta \neq \theta', \quad (18.5)$$

such that we clearly have an identifiability problem.

Since we only have two parameters, we can easily plot the associated parameter space. Figure 18.2 shows example paths in the parameter space,

$$\alpha \mapsto \begin{pmatrix} \alpha \theta_1 \\ \theta_2/\alpha \end{pmatrix}, \quad (18.6)$$

along which the corresponding function is unchanged. Ideally, we would like to give the parameter space a geometry such that

$$\operatorname{dist}(\theta, \theta') = 0 \quad \text{for } \theta = (\theta_1, \theta_2) \text{ and } \theta' = (\alpha \theta_1, \theta_2/\alpha). \quad (18.7)$$

Clearly, Euclidean geometry does not have this property.

## 18.2. Function-space metrics

We have previously seen that pulling back a metric from the output space gives the input space a natural geometric structure. In the

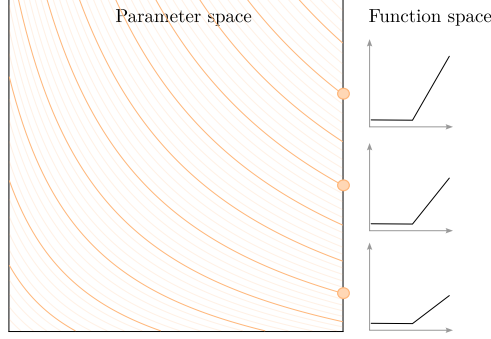


FIG. 18.2. The function  $f_{\theta}(\mathbf{x}) = \theta_2 \text{ReLU}(\theta_1 \mathbf{x})$  is overparametrized such that different parameters can lead to the same function. The orange curves in the left panel indicate paths along which  $f$  does not change even if  $\theta$  changes. For this simple function, we can explicitly characterize these paths, but in general, we cannot.

current regime, we would need a metric over the space of generated functions to give the parameter space a natural metric.

One option is to consider the *expected inner product* between function outputs with respect to a density  $p(\mathbf{x})$ ,

$$\langle f_{\theta}, f_{\theta'} \rangle = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [f_{\theta}(\mathbf{x})^{\top} f_{\theta'}(\mathbf{x})]. \quad (18.8)$$

This inner product is constant when either function is reparametrized. Pulling back this metric gives the usual infinitesimal inner product (these derivations are completely analogous to those in equations 5.11 to 5.13)

$$\langle \delta_1, \delta_2 \rangle_{\theta} = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [(f_{\theta+\delta_1}(\mathbf{x}) - f_{\theta}(\mathbf{x}))^{\top} (f_{\theta+\delta_2}(\mathbf{x}) - f_{\theta}(\mathbf{x}))] \quad (18.9)$$

$$= \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [(\mathbf{J}_{\theta}(\mathbf{x}) \delta_1)^{\top} (\mathbf{J}_{\theta}(\mathbf{x}) \delta_2)] \quad (18.10)$$

$$= \delta_1^{\top} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\mathbf{J}_{\theta}(\mathbf{x})^{\top} \mathbf{J}_{\theta}(\mathbf{x})] \delta_2. \quad (18.11)$$

Here, the Jacobian  $\mathbf{J}_{\theta}(\mathbf{x}) = \partial f_{\theta}(\mathbf{x}) / \partial \theta \in \mathbb{R}^{D \times P}$  is with respect to the parameters  $\theta$  and, thus, depends on  $\mathbf{x}$ . As usual,  $D$  indicates the output dimension of  $f_{\theta}$ .

Pulling back the expected inner product, thus, gives a local metric

$$\mathbf{G}_{\theta} = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\mathbf{J}_{\theta}(\mathbf{x})^{\top} \mathbf{J}_{\theta}(\mathbf{x})] \in \mathbb{R}^{P \times P}. \quad (18.12)$$

The distance measure under this local metric, by construction, satisfies Eq. 18.7, such that reparametrizations of a function are distance zero apart.

The alert reader may question if this local metric is *Riemannian*. It is not. For a metric to be Riemannian, we require that

$$\theta \neq \theta' \Rightarrow \text{dist}(\theta, \theta') > 0, \quad (18.13)$$

which, by design, is not the case with the function-space metric in Eq. 18.12. We shall, however, soon see that we can still recover a suitable Riemannian metric such that our previous arsenal of methods still applies. But first, we consider an alternative choice of metric.

### 18.3. Information metrics

Equation 18.8 assumes an Euclidean metric over the output space. As discussed on page 110, this inner product depends on how we choose to parametrize the density  $f_\theta(\mathbf{x})$ , which introduces an element of arbitrariness in our analysis. Since we only engage with the output-space inner product *locally*, we can choose any Riemannian metric  $\mathbf{F}_{f_\theta(\mathbf{x})}$  for this space. Our infinitesimal inner product (18.11) then becomes

$$\langle \delta_1, \delta_2 \rangle_\theta^{\mathbf{F}} = \delta_1^\top \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\mathbf{J}_\theta(\mathbf{x})^\top \mathbf{F}_{f_\theta(\mathbf{x})} \mathbf{J}_\theta(\mathbf{x})] \delta_2. \quad (18.14)$$

A natural choice of metric  $\mathbf{F}_{f_\theta(\mathbf{x})}$  is the Fisher–Rao metric, which measures local changes in terms of the KL divergence between distributions. In this case, the parameter-space metric becomes

$$\mathbf{G}_\theta = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\mathbf{J}_\theta(\mathbf{x})^\top \mathbf{F}_{f_\theta(\mathbf{x})} \mathbf{J}_\theta(\mathbf{x})]. \quad (18.15)$$

### 18.4. Pseudo metrics

We can understand the geometry of the parameter space by looking more carefully at the metric. The analysis is the same if we consider Eq. 18.12 or Eq. 18.15, so we let  $\mathbf{G}_\theta$  denote either choice.

Directions exist in parameter space in which the associated function does not change, implying that distances are zero in those directions.

This implies that the metric  $\mathbf{G}_\theta \in \mathbb{R}^{P \times P}$  has a null space. We let  $R$  denote the rank of  $\mathbf{G}_\theta$  such that the null space has dimensionality  $P - R$ . In practice, we won't know the rank  $R$  analytically and will rely on numerical techniques to get an estimate.

To give ourselves a notation to work with, we introduce an eigendecomposition of the metric

$$\mathbf{G}_\theta = \mathbf{V}_\theta \boldsymbol{\lambda}_\theta \mathbf{V}_\theta^\top, \quad (18.16)$$

where  $\boldsymbol{\lambda}_\theta$  is a diagonal matrix containing the eigenvalues of  $\mathbf{G}_\theta$ . Due to  $\mathbf{G}_\theta$  having a null space, some of these eigenvalues will be zero, which we explicate as (dropping the  $\theta$  subscript to reduce notational clutter)

$$\mathbf{G}_\theta = (\mathbf{V}_{\text{im}} \quad \mathbf{V}_{\text{ker}}) \begin{pmatrix} \boldsymbol{\lambda}_{\text{im}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} (\mathbf{V}_{\text{im}} \quad \mathbf{V}_{\text{ker}})^\top. \quad (18.17)$$

Here  $\mathbf{V}_{\text{im}} \in \mathbb{R}^{P \times R}$  is an orthonormal basis of the *image* of  $\mathbf{G}_\theta$ , i.e.,  $\mathbf{V}_{\text{im}}$  contains the eigenvectors of  $\mathbf{G}_\theta$  that have nonzero eigenvalues. Likewise,  $\mathbf{V}_{\text{ker}} \in \mathbb{R}^{P \times P-R}$  is an orthonormal basis of the *kernel*<sup>2</sup> of  $\mathbf{G}_\theta$ .

The metric eigendecomposition (18.17) suggests that, locally, the parameter space decomposes into a *kernel part*, where functions don't change, and an *image part*, where functions change. These parts turn out to be manifolds that we can engage with.

## 18.5. Kernel manifolds

The metric kernel is the set of possible directions in parameter space in which functions do not change locally. Assuming  $f_\theta$  is a smooth function, the Jacobian  $\mathbf{J}_\theta(\mathbf{x})$  changes smoothly. This, in turn, implies that the eigenvectors and values of the metric also change smoothly. It is then possible to trace out trajectories in parameter space with zero length. Formally, we say that  $c : [0, 1] \rightarrow \mathbb{R}^P$  is an *improper curve* if its speed  $\|\dot{c}_t\|_{\mathbf{G}_{c_t}}$  is zero for all  $t$ . From a given starting point  $\theta \in \mathbb{R}^P$ , we can consider the set of points along such improper curves that connect

---

<sup>2</sup>The *kernel* of a matrix is equivalent to its *null space*. We use the term *kernel* as it will allow us to define *kernel manifolds*, which, bluntly put, sounds better than *null space manifolds*.

to the starting point,

$$\mathcal{M}_{\text{ker}}^\theta = \{\theta' \in \mathbb{R}^P \mid \exists c \text{ an improper curve such that } c_0 = \theta \text{ and } c_1 = \theta'\}. \quad (18.18)$$

For any  $\theta' \in \mathcal{M}_{\text{ker}}^\theta$ , we then have that  $f_\theta = f_{\theta'}$ . We call  $\mathcal{M}_{\text{ker}}^\theta$  the *kernel manifold* and note that it characterizes the orange reparametrization curves in the exemplary Fig. 18.2.

Neatly,  $\mathcal{M}_{\text{ker}}^\theta$  is a (path-connected)  $(P-R)$ -dimensional manifold. The natural metric on this manifold suggests that all pairwise distances within  $\mathcal{M}_{\text{ker}}^\theta$  are zero. Sometimes it is convenient to give the kernel manifold a (proper) Riemannian metric. The, by far, simplest choice is to use the Euclidean parameter space metric on  $\mathcal{M}_{\text{ker}}^\theta$ , which can be written as  $\mathbf{G}_\theta^{\text{ker}} = \mathbf{V}_{\text{ker}} \mathbf{V}_{\text{ker}}^\top$ . As an example, Roy et al. [40] uses such a metric to define distributions over reparametrizations of neural networks. However, I want to emphasize that such a metric is *not* invariant to reparametrizations.

## 18.6. Image manifolds

Similarly, we can define the *image manifold* as the set of points connected by paths with strictly positive speeds. To formalize, say that  $c : [0, 1] \rightarrow \mathbb{R}^P$  is a *proper curve* if its speed  $\|\dot{c}_t\|_{\mathbf{G}_{c_t}}$  is strictly positive for all  $t$ . We then define the *image manifold* as

$$\mathcal{M}_{\text{im}}^\theta = \{\theta' \in \mathbb{R}^P \mid \exists c \text{ a proper curve such that } c_0 = \theta \text{ and } c_1 = \theta'\}. \quad (18.19)$$

Figure 18.3 shows image and kernel manifolds for the overparametrized function  $f_\theta(\mathbf{x}) = \theta_2 \text{ReLU}(\theta_1 \mathbf{x})$ . Note that the image manifold changes with function parametrization.

As before,  $\mathcal{M}_{\text{im}}^\theta$  is an  $R$ -dimensional manifold, and we can give it a metric to practically engage with the manifold. Unlike for the kernel manifold, we have a natural choice of Riemannian metric over  $\mathcal{M}_{\text{im}}^\theta$ , which is simply

$$\mathbf{G}_\theta^{\text{im}} = \mathbf{V}_{\text{im}} \boldsymbol{\lambda}_{\text{im}} \mathbf{V}_{\text{im}}^\top = \mathbf{G}_\theta. \quad (18.20)$$

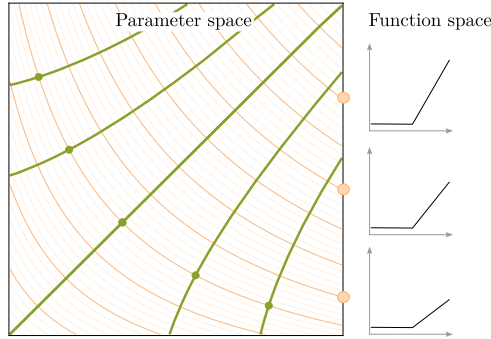


FIG. 18.3. Reparametrizations of  $f_{\theta}(\mathbf{x}) = \theta_2 \text{ReLU}(\theta_1 \mathbf{x})$ . Akin to Fig. 18.2, orange curves in the left panel show kernel manifolds. Green curves show image manifolds relative to the parameters indicated by green circles.

Thus, if we restrict ourselves to parameters connected by *proper* curves, the pull-back metric gives us access to a Riemannian manifold. Points on this manifold correspond to *distinct* functions, while the kernel manifold captures reparametrizations of a given function.

### 18.7. Finite-sample metric estimators

So far, we have considered metrics taken as expectations over a data distribution,

$$\mathbf{G}_{\theta} = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\mathbf{J}_{\theta}(\mathbf{x})^{\top} \mathbf{F}_{f_{\theta}(\mathbf{x})} \mathbf{J}_{\theta}(\mathbf{x})]. \quad (18.21)$$

In practice, we rarely have access to  $p(\mathbf{x})$ , and if we had, we would rarely be able to carry out the expectation in closed form. In practice, we generally have to resort to a Monte Carlo estimate of the metric,

$$\bar{\mathbf{G}}_{\theta} = \frac{1}{S} \sum_{s=1}^S \mathbf{J}_{\theta}(\mathbf{x}_s)^{\top} \mathbf{F}_{f_{\theta}(\mathbf{x}_s)} \mathbf{J}_{\theta}(\mathbf{x}_s), \quad (18.22)$$

where the samples  $\mathbf{x}_s$  are assumed to be drawn from  $p(\mathbf{x})$ . In practice, the samples would constitute our training data or a subset thereof.

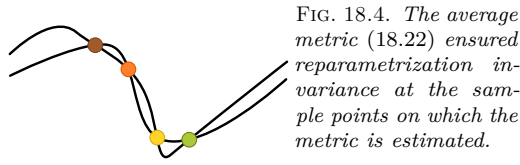


FIG. 18.4. The average metric (18.22) ensured reparametrization invariance at the sample points on which the metric is estimated.

The *average metric* (18.22) is an instance of the so-called *generalized Gauss-Newton* matrix, which is commonly used in optimization to approximate the Hessian of a function [13].

The average metric (18.22) has similar, but not identical, invariance properties as the expected metric (18.21). Under the average metric, we see that for an infinitesimal vector  $\delta$ ,

$$\text{dist}_{\bar{\mathbf{G}}_\theta}^2(\theta, \theta + \delta) = \delta^\top \bar{\mathbf{G}}_\theta \delta \quad (18.23)$$

$$= \frac{1}{S} \sum_{s=1}^S \delta^\top \mathbf{J}_\theta(\mathbf{x}_s)^\top \mathbf{F}_{f_\theta(\mathbf{x}_s)} \mathbf{J}_\theta(\mathbf{x}_s) \delta \quad (18.24)$$

$$= \frac{1}{S} \sum_{s=1}^S \|f_{\theta+\delta}(\mathbf{x}_s) - f_\theta(\mathbf{x}_s)\|_{\mathbf{F}_{f_\theta(\mathbf{x}_s)}}^2. \quad (18.25)$$

As the individual terms in the sum are non-negative, the distance is only zero if the two functions  $f_{\theta+\delta}$  and  $f_\theta$  take the same value at the sample points  $\{\mathbf{x}_s\}_{s=1}^S$ . Figure 18.4 illustrates the situation.

The kernel manifold associated with the average metric (18.22) is, thus, less restricted than that of the expected metric (18.21). The average metric only retains invariance properties *on the sampled data*. This implies that caution should be taken when small amounts of data are available, as the invariance is potentially quite limited.

Consider an infinite sequence of independent samples  $\{\mathbf{x}_1, \mathbf{x}_2, \dots\}$  from  $p(\mathbf{x})$  and let  $\mathcal{M}_{\text{ker}}^{\theta, S}$  be the kernel manifold associated with the average metric estimated from the first  $S$  samples. Then we see that

$$\mathcal{M}_{\text{ker}}^\theta \subseteq \dots \subseteq \mathcal{M}_{\text{ker}}^{\theta, S} \subseteq \mathcal{M}_{\text{ker}}^{\theta, S-1} \subseteq \dots \subseteq \mathcal{M}_{\text{ker}}^{\theta, 1}. \quad (18.26)$$

Said differently, when we increase the number of training data, we get closer to achieving full reparametrization invariance.

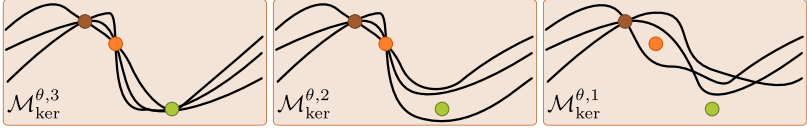


FIG. 18.5. *Finite-sample estimators of the expected pull-back metric (18.21) are only partially invariant to reparametrizations. More observations imply a greater extent of invariance.*

## Exercises

### Exercise 18.1: Bias-free ReLU networks.

Consider a neural network consisting of layers

$$f_l(\mathbf{x}) = \text{ReLU}(\theta_l \mathbf{x}), \quad (18.27)$$

such that the joint network is

$$f(\mathbf{x}) = (f_L \circ \dots \circ f_1)(\mathbf{x}) = f_L(\dots f_2(f_1(\mathbf{x}))). \quad (18.28)$$

1. Show that two parameter vectors  $\theta$  and  $\theta'$  produce the same function  $f$  if

$$\sum_{l=1}^L \log \theta_l = \sum_{l=1}^L \log \theta'_l. \quad (18.29)$$

### Exercise 18.2: ReLU geodesics.

Consider the neural network

$$f(\mathbf{x}) = \theta_2 \text{ReLU}(\theta_1 \mathbf{x}) \quad (18.32)$$

and observational data

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{y}_1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbf{x}_2 \\ \mathbf{y}_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \quad (18.33)$$

This data can perfectly fit with the parameters  $\theta_1 = \theta_2 = 1$ .

1. Show that  $\theta_1 = 7$  and  $\theta_2 = 1/7$  provides an equally good fit to the data.



2. Derive the finite-sample estimate of the metric  $\mathbf{G}_\theta = \mathbb{E}[\mathbf{J}_\theta(\mathbf{x})^\top \mathbf{J}_\theta(\mathbf{x})]$  given by

$$\bar{\mathbf{G}}_\theta = \frac{1}{2} \left( \mathbf{J}_\theta(\mathbf{x}_1)^\top \mathbf{J}_\theta(\mathbf{x}_1) + \mathbf{J}_\theta(\mathbf{x}_2)^\top \mathbf{J}_\theta(\mathbf{x}_2) \right). \quad (18.34)$$

3. Write a computer program to compute the geodesic between parameters

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} 21 \\ 3/7 \end{pmatrix}. \quad (18.35)$$

*Exercise 18.3: Hyperbolic tangent networks.*

Consider the neural network

$$f(\mathbf{x}) = \theta_2 \tanh(\theta_1 \mathbf{x}) \quad (18.39)$$

and observational data

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{y}_1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbf{x}_2 \\ \mathbf{y}_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \quad (18.40)$$

1. Derive the finite-sample estimate of the metric  $\mathbf{G}_\theta = \mathbb{E}[\mathbf{J}_\theta(\mathbf{x})^\top \mathbf{J}_\theta(\mathbf{x})]$  given by

$$\bar{\mathbf{G}}_\theta = \frac{1}{2} \left( \mathbf{J}_\theta(\mathbf{x}_1)^\top \mathbf{J}_\theta(\mathbf{x}_1) + \mathbf{J}_\theta(\mathbf{x}_2)^\top \mathbf{J}_\theta(\mathbf{x}_2) \right). \quad (18.41)$$

It may be helpful to recall that the derivative of  $\tanh(\mathbf{x})$  is  $1 - \tanh^2(\mathbf{x})$ .

2. Write a computer program to compute the geodesic between parameters

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} 21 \\ 3/7 \end{pmatrix}. \quad (18.42)$$

*Exercise 18.4: Uncertainty quantification, take 1.*

Consider the dataset

$$\mathbf{x}_n \sim \mathcal{N}(0, 1), \quad n = 1, \dots, 25, \quad (18.48)$$

$$\mathbf{y}_n = \sin(\pi \mathbf{x}_n) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.05). \quad (18.49)$$

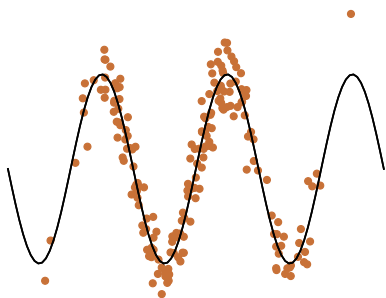


FIG. 18.6. The synthetic dataset used in exercise 18.4 (brown dots). The black curve indicates the true underlying function  $y = \sin(\pi x)$ .

1. Train a neural network regressor on the generated data.
2. Write a computer program to evaluate the average metric (18.22) associated with the trained model.
3. Perform an eigen-decomposition of the average metric to estimate the image and kernel directions.
4. Sample from a standard Gaussian distribution in the parameter space, and project the samples onto the kernel directions. What is the empirical behavior of these projected samples when treated as new neural network models?

*Exercise 18.5: Uncertainty quantification, take 2.*

Consider a randomly selected subset of the MNIST data consisting of 1000 observations.

1. Train a neural network classifier on the selected data.
2. Write a computer program to evaluate the average metric (18.22) associated with the trained model.
3. Perform an eigen-decomposition of the average metric to estimate the image and kernel directions.
4. Sample from a standard Gaussian distribution in the parameter space, and project the samples onto the kernel directions. What is the empirical behavior of these projected samples when treated as new neural network models?

## Chapter 19

### *Parting remarks & further reading*

*In which our story ends,  
while many new begin.*

No text is complete before acknowledging its incompleteness. This book is incomplete in a multitude of ways.

First and foremost, this is a book on differential geometry, yet it only scratches the surface of this topic. If you continue to study geometry, I hope this book has given you the intuitions and practical understanding that the traditional mathematical literature is more easily approachable. One potential reference is the excellent book by Lee [32].

#### *19.1. Disregarded material*

For a text on differential geometry, I have held out several topics that are usually considered essential. Hopefully, this has not been too upsetting to the reader. The most glaring holdout is *curvature*.

Riemannian geometry is concerned with making quantitative statements about curved objects (manifolds). Topology is concerned with the object itself, e.g., if there are holes in our manifold. Arguably, the treatment of topology in this book is quite superficial, as we only provide sufficient details to define the concept of a manifold. Much more could, and perhaps should, be said.

In the context of generative models, we rarely have much topological information a priori,<sup>1</sup> making it difficult to incorporate topological information into our models. A more common scenario is the desire to estimate the topology of a data manifold. *Topological data analysis*

---

<sup>1</sup>e.g., when do you know the Betti numbers associated with a dataset?

[10] seeks to answer this question, though the associated tools and algorithms are still in their infancy.

The other held-out topic, *curvature*, in contrast, plays a quite central role in machine learning. In essence, the many different measures of curvature attempt to quantify how much our manifolds bend and stretch. Practically all *regularization* techniques are designed to minimize such bending. Yet regularization techniques are rarely developed using geometric notions of curvature and instead, consider much simpler alternatives.<sup>2</sup> This could suggest that geometric notions of curvature are overly complex to be practical or that there is a missed opportunity for regularization research.

## 19.2. Historical developments

As far as I know, statistical identifiability was first discussed in 1919 in the seminal paper by Thomson [45]. Interestingly, many of the same problems were studied and solved, in a different context, by Gauss [16] and Riemann [38] in the mid-eighteenth century. Arguably, the identifiability problem is broader than those studied here, but the order of events is interesting nonetheless. Syrota et al. [44] provides a measure-theoretic proof that pull-back metrics provide the promised identifiability proof.

There is a long and rich history of differential geometry in data science. One of the key works in this direction is Ulf Grenander's *pattern theory* [18], which argues that statistical data analysis should focus on how we *transform* one observation into another as the grounding principle. For example, when studying Euclidean data, this transformation is merely a *translation*, but if we study data on the unit sphere, the natural transformation is rather a *rotation*.

A similar line of thinking is ever-present in the analysis of *shape data*. This started from the ideas of D'Arcy Thompson [14] who studied the evolution of *fish* through changes in their shape. This reduces the statistical analysis to focus on the *transformations* between different fish, rather than on the individual fish (see figure).

The idea of statistically analyzing transformations naturally leads

---

<sup>2</sup>e.g.,  $L_2$  weight regularization, *dropout*, etc.

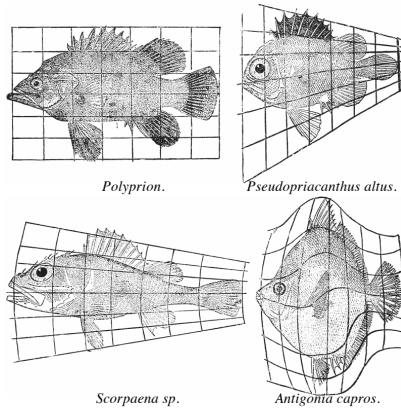


FIG. 19.1. D'Arcy Thompson [14] pioneered the idea of studying evolution through shape. Since it's difficult to parametrize the actual shape of an animal (here fish), he opted to parametrize the transformations between shapes (background grid). One can then think of these transformations as the observed data and study them statistically. The figure is taken from the original book of D'Arcy Thompson.

to differential geometry. In particular, the space of *diffeomorphisms*<sup>3</sup> is naturally modeled as a Riemannian manifold. The question of how we best parametrize such transformations is heavily studied in *shape statistics*. Much fundamental work on statistics on manifolds is developed within this context, see e.g., the wonderful introduction by Pennec [34].

This book focuses on the geometry induced by generative models. The development of this was greatly inspired by the above works. To the best of my knowledge, the first paper in this direction was the work of Tosi et al. [48] who studied expectations of Riemannian metrics in models where the map from latent space to observation space is a Gaussian process [36]. Arvanitidis et al. [6] revitalized this work within neural network models. In this process, the mathematical elegance of Gaussian processes was replaced by the empirical effectiveness of neural networks. This sparked several following works, including applications in *clustering* [52], *biology* [15], *optimization* [1] and *robotics* [7, 42, 11].

The original Gaussian process model [48] has undergone deeper

<sup>3</sup>i.e., invertible smooth transformations with smooth inverses

theoretical study. Adler and Taylor [2] provides a thorough investigation of similar models for the case of a *prior* model.<sup>4</sup> Hauberg [21] discuss links between uncertainty and topology through studies of Gaussian process extrapolations. Finally, Pouplin et al. [35] shows that the original idea of looking at *expected Riemannian metrics* is not entirely unproblematic. They show that random Riemannian manifolds do not naturally result in Riemannian manifolds in expectation, but rather in so-called Finsler manifolds. Fortunately, these turn out to be remarkably similar.

Our objective is to use geometry to characterize latent variable models. The wider goal of characterizing distributions is studied in information geometry [3, 33]. We briefly touched on this topic in Chapter 15, but the bulk of this field was ignored to focus the text. Information geometry plays a key role in modern optimization and sampling algorithms and many other areas [25, 17]. The classic book by Amari [3] and the more recent primer by Nielsen [33] are both highly recommended readings.

---

<sup>4</sup>i.e., the model before observing any data

## Appendix A

### Ordinary differential equations

*In which we take one step at a time.*

We use ordinary differential equations (ODEs) all the time in machine learning, but we often do not talk about them. The text below is by no means a complete exposition, but it should suffice for an operational understanding.

#### A.1. First-order ODEs and systems thereof

A *first-order ODE* is an equation of the form

$$\dot{c}_t = f(t, c_t), \tag{A.1}$$

where  $c : [0, 1] \rightarrow \mathbb{R}$  is a curve, and  $\dot{c} : [0, 1] \rightarrow \mathbb{R}$  is the curve derivative, i.e.,  $\dot{c}_t = dc/dt$ . The function  $f : [0, 1] \times \mathbb{R} \rightarrow \mathbb{R}$ , thus, evaluates the derivative of the curve  $c$  assuming we know the actual value of the curve.<sup>1</sup>

Usually, we work with ODEs when we are interested in a curve  $c$ , but we only have access to the derivative of the curve. Our prime use case is when computing geodesics, we can derive an ODE that geodesics must satisfy and use this to arrive at a numerical algorithm.

Often, the unknown curve lives in  $\mathbb{R}^D$  rather than being in a one-dimensional space. In this case, the ODE now take the form  $f : [0, 1] \times \mathbb{R}^D \rightarrow \mathbb{R}^D$ , i.e., we have

$$f(t, c_t) = \begin{pmatrix} f_1(t, [c_t]_1) \\ \vdots \\ f_D(t, [c_t]_D) \end{pmatrix}. \tag{A.2}$$

---

<sup>1</sup>The choice of the interval  $[0, 1]$  is arbitrary, and you should feel free to change to another interval if that's easier.

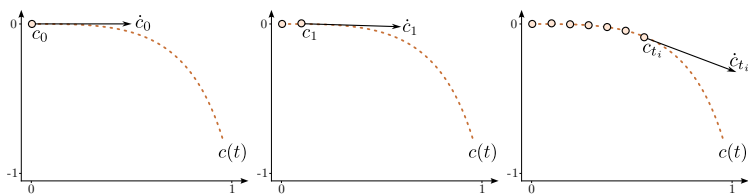


FIG. A.1. *Euler's method for solving ODEs merely evaluates the given ODE and uses the given derivative to approximate the underlying function. Step by step, this traces out a function estimate.*

This is often known as a *system of ODEs*.

## A.2. Euler's method

The easiest way to understand ODEs is to consider how you might solve one. That is, given an ODE and an *initial value*

$$c(0) = c_0, \tag{A.3}$$

i.e., you should assume that we know the true value of the curve  $c$  at time  $t = 0$ . From this knowledge, we can evaluate the ODE to provide us with the derivative of the unknown curve  $c$  at  $t = 0$ ,

$$\left. \frac{dc}{dt} \right|_{t=0} = \dot{c}_0 = f(0, c_0). \tag{A.4}$$

If we make a Taylor expansion of the unknown curve around  $c_0$ , we get

$$c(t) = c_0 + \dot{c}_0 t + \mathcal{O}(t^2). \tag{A.5}$$

This suggests that for small values of  $t$  we can evaluate the curve as  $c(t) \approx c_0 + \dot{c}_0 t = c_o + f(0, c_o)t$ . This is what *Euler's method* does to approximate the solution to an ODE. Given a sequence of time-steps  $\{t_1 = 0, t_2, \dots, t_N = 1\}$ , we use the ODE to form a local Taylor expansion from which we extrapolate the solution curve and proceed to the next time-step:



*Alg.A.2: Euler's method***Input:**  $c_0, f$ .For  $i \in 0, 1, \dots, N$ :

$$\dot{c}_{t_i} = f(\tau, c_{t_i})$$

$$c_{t_{i+1}} = c_{t_i} + \dot{c}_{t_i}(t_{i+1} - t_i)$$

Euler's method is *not* a very good numerical algorithm, but it is a great way to gain intuitions about the behavior of an ODE. More elaborate numerical algorithms for solving ODEs include the *Runge-Kutta* family of solvers [49], which are often easily available in most programming environments.

Euler's method is practically identical to *gradient descent*, which is ever-present in machine learning. Here, we often define some loss function, but during optimization, we mostly only consider its gradient. When applying gradient descent, you can view the function that returns the gradient of the loss as an ODE, which we solve with Euler's method.

### A.3. The Picard-Lindelöf theorem

One of the most fundamental results in the theory of ODEs is the *Picard-Lindelöf theorem*. This formalizes the highly intuitive notion that the solution to an ODE is unique for a given initial value. Given that Euler's method is a fully deterministic algorithm, it is clear that the solution recovered by this method is unique. If you imagine taking infinitesimally small time steps with Euler's method, you basically arrive at the Picard-Lindelöf theorem.

Formally, the theorem states that if the ODE is continuous in  $t$  and Lipschitz continuous<sup>2</sup> in  $c_t$ , then there exists an  $\epsilon > 0$  such that the ODE has a unique solution for  $t \in [t_0 - \epsilon, t_0 + \epsilon]$ .

---

<sup>2</sup>i.e., continuous and with bounded derivatives

### A.4. Differentiating through ODEs

We often need to differentiate through ODE solutions. For example, we may consider some loss function that depends on the end-position of the ODE solution,

$$L = \phi(c_1). \quad (\text{A.6})$$

Assuming that the differential equation itself depends on some parameters  $\theta$ , we are interested in evaluating  $\partial L / \partial \theta$ .

A naive evaluation of  $\partial L / \partial \theta$  is to differentiate through the numerical solution, which is straightforward using automatic differentiation. Unfortunately, this can have a significant memory overhead as we need to store all intermediate steps taken by the numerical algorithm. For large problems, this quickly becomes infeasible.

The *adjoint method* provides an alternative, where we express the derivative  $\partial L / \partial \theta$  through the solution to another ODE. We can solve this using any numerical algorithm, which alleviates the high memory use of the naive approach. It can be shown that

$$\frac{dL}{d\theta} = \int_0^1 a_t^\top \frac{\partial f}{\partial \theta}(t, c_t) dt, \quad (\text{A.7})$$

where the *adjoint*  $a_t$  is the solution to the ODE

$$\frac{da}{dt} = -a_t^\top \frac{\partial f}{\partial c}(t, c_t), \quad (\text{A.8})$$

with the terminal condition

$$a_1 = \frac{\partial \phi}{\partial c}(c_1). \quad (\text{A.9})$$

Contemporary implementations of numerical ODE often implement the adjoint method to support automatic differentiation through the solver. In practice, it is, unfortunately, common that solving the adjoint ODE is numerically less stable than solving the original ODE. This implies that automatic differentiation through ODE may require some monitoring to ensure that results are reliable.

## Appendix B

### *Functional derivatives*

*In which the derivatives are rather  
high-dimensional.*

We commonly minimize some loss function  $L$  with respect to some vector  $\mathbf{x}$  by differentiating the loss with respect to the vector,  $\partial L / \partial \mathbf{x}$ . When, e.g., computing geodesics, we do not seek an optimal *vector*, but rather an optimal *curve*. In this case, our usual differentiation does not work. The *Euler-Lagrange* equation provides a remedy.

#### *B.1. The Euler-Lagrange equation*

We start our quest for a suitable derivative with a loss function  $L$ . We are interested in finding a curve  $c : [0, 1] \leftarrow \mathbb{R}^D$  that minimizes the *functional* (a functional is merely a function that takes another function as input)

$$S = \int_0^1 L(t, c_t, \dot{c}_t) dt. \quad (\text{B.1})$$

To find such an optimal curve, we first discretize the integral (B.1) after which we will take the corresponding limit to get a continuous result.

To discretize, we split the interval  $[0, 1]$  into  $N + 1$  points spread a part by a factor  $\epsilon = 1/N$ . We write these points as  $t_i = i\epsilon$  for  $i = 0, 1, \dots, N$ , resulting in the finite approximation

$$S_N = \sum_{i=0}^{N-1} \epsilon L(t_i, c_{t_i}, (c_{t_{i+1}} - c_{t_i})/\epsilon). \quad (\text{B.2})$$

Minimizing  $S_N$  with respect to  $c_{t_i}$  is a standard optimization problem, and we can differentiate as usual. Taking the derivative with respect to  $c_{t_i}$  gives

$$\frac{\partial S_N}{\partial c_{t_i}} = \epsilon \left( \frac{\partial L_i}{\partial c_{t_i}} + \frac{\partial L_i}{\partial \dot{c}_{t_i}} \frac{\partial \dot{c}_{t_i}}{\partial c_{t_i}} + \frac{\partial L_{i-1}}{\partial \dot{c}_{t_{i-1}}} \frac{\partial \dot{c}_{t_{i-1}}}{\partial c_{t_i}} \right), \quad (\text{B.3})$$

where we have introduced the short-hand  $L_i = L(t_i, c_{t_i}, (c_{t_{i+1}} - c_{t_i})/\epsilon)$ .

We still need the derivatives of  $\dot{c}_{t_i}$  and  $\dot{c}_{t_{i-1}}$ , which trivially evaluate to

$$\frac{\partial \dot{c}_{t_i}}{\partial c_{t_i}} = \frac{\partial}{\partial c_{t_i}} ((c_{t_{i+1}} - c_{t_i})/\epsilon) = -\frac{1}{\epsilon}, \quad (\text{B.4})$$

$$\frac{\partial \dot{c}_{t_{i-1}}}{\partial c_{t_i}} = \frac{\partial \dot{c}_{t_{i-1}}}{\partial} ((c_{t_i} - c_{t_{i-1}})/\epsilon) = \frac{1}{\epsilon}. \quad (\text{B.5})$$

Putting these together gives

$$\frac{\partial S_{\text{discrete}}}{\partial c_{t_i}} = \epsilon \left( \frac{\partial L_i}{\partial c_{t_i}} - \frac{\partial L_i}{\partial \dot{c}_{t_i}} \frac{1}{\epsilon} + \frac{\partial L_{i-1}}{\partial \dot{c}_{t_{i-1}}} \frac{1}{\epsilon} \right) \quad (\text{B.6})$$

$$= \epsilon \frac{\partial L_i}{\partial c_{t_i}} - \frac{\partial L_i}{\partial \dot{c}_{t_i}} + \frac{\partial L_{i-1}}{\partial \dot{c}_{t_{i-1}}}. \quad (\text{B.7})$$

Since we are looking for an optimum, we set this derivative to zero and reshuffle

$$\frac{\partial L_i}{\partial c_{t_i}} - \frac{1}{\epsilon} \left( \frac{\partial L_i}{\partial \dot{c}_{t_i}} - \frac{\partial L_{i-1}}{\partial \dot{c}_{t_{i-1}}} \right) = 0. \quad (\text{B.8})$$

This is the discrete version of the Euler-Lagrange equations. When we take the limit  $\epsilon \rightarrow 0$ , we get the continuous version,

$$\frac{\partial L}{\partial c_t} - \frac{d}{dt} \frac{\partial L}{\partial \dot{c}_t} = 0. \quad (\text{B.9})$$

This equation must be satisfied by the curve  $c$  that minimizes Eq. B.1.

In practice, using the Euler-Lagrange equations is purely mechanical: we evaluate the individual derivatives, put them together and shuffle things around until we get a useful result, often in the form of a differential equation.

## Bibliography

- [1] Gabriele Abbati, Alessandra Tosi, Michael Osborne, and Seth Flaxman. Adageo: Adaptive geometric learning for optimization and sampling. In *International conference on artificial intelligence and statistics*, pages 226–234. PMLR, 2018.
- [2] Robert J Adler and Jonathan E Taylor. *Random fields and geometry*. Springer Science & Business Media, 2009.
- [3] Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.
- [4] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.
- [5] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. A locally adaptive normal distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, jun 2016.
- [6] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations (ICLR)*, 2018.
- [7] Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Gerhard Neumann, and Leonel Roza. Learning riemannian manifolds for geodesic motion skills. In *Robotics: Science and Systems (RSS)*, 2021.
- [8] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.
- [9] Christopher M Bishop, Markus Svensén, and Christopher KI Williams. Gtm: The generative topographic mapping. *Neural computation*, 10(1):215–234, 1998.

- [10] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.
- [11] Nutan Chen, Alexej Klushyn, Alexandros Paraschos, Djalel Benbouzid, and Patrick Van der Smagt. Active learning based on data uncertainty and model sensitivity. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1547–1554. IEEE, 2018.
- [12] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.
- [13] F Dan Foresee and Martin T Hagan. Gauss-newton approximation to bayesian learning. In *International Conference on Neural Networks*, volume 3, pages 1930–1935, 1997.
- [14] D’Arcy Thompson. *On Growth and Form*. Cambridge University Press, 1917.
- [15] Nicki Skaftø Detlefsen, Søren Hauberg, and Wouter Boomsma. Learning meaningful representations of protein sequences. *Nature communications*, 13(1):1914, 2022.
- [16] Carl Friedrich Gauss. Disquisitiones generales circa superficies curvas. *Commentationes Societatis Regiae Scientiarum Gottingensis Recentiores. Comm. Class.*, 1827.
- [17] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73(2):123–214, 2011.
- [18] Ulf Grenander and Michael I Miller. *Pattern theory: from representation to inference*. OUP Oxford, 2006.
- [19] Peter E Hart, David G Stork, and Richard O Duda. *Pattern classification*. Wiley, 2000.
- [20] Søren Hauberg. Directional statistics with the spherical normal distribution. In *Proceedings of FUSION 2018*, 2018.

- [21] Søren Hauberg. Only bayes should learn a manifold, 2018.
- [22] Søren Hauberg, Oren Freifeld, and Michael J. Black. A geometric take on metric learning. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NeurIPS) 25*, pages 2033–2041. MIT Press, 2012.
- [23] Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Unraveling the single tangent space fallacy: An analysis and clarification for applying riemannian geometry in robot learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 242–249. IEEE, 2024.
- [24] Rafal Karczewski, Markus Heinonen, Alison Pouplin, Søren Hauberg, and Vikas K Garg. The spacetime of diffusion models: An information geometry perspective. In *The Fourteenth International Conference on Learning Representations*, 2026.
- [25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Representation Learning (ICLR)*, 2015.
- [26] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.
- [27] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.
- [28] Tjalling C Koopmans and Olav Reiersøl. The identification of structural characteristics. *The Annals of Mathematical Statistics*, 21(2):165–181, 1950.
- [29] Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. *The Principles of Diffusion Models*. 2025.
- [30] Neil Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of machine learning research*, 6(Nov):1783–1816, 2005.

- [31] Guy Lebanon. Learning riemannian metrics. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 362–369, 2002.
- [32] John M. Lee. *Introduction to Smooth Manifolds*. Springer, 2012.
- [33] Frank Nielsen. An elementary introduction to information geometry. *Entropy*, 22(10):1100, 2020.
- [34] Xavier Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25:127–154, 2006.
- [35] Alison Pouplin, David Eklund, Carl Henrik Ek, and Søren Hauberg. Identifying latent distances with finlerian geometry. *Transactions on Machine Learning Research (TMLR)*, 2023.
- [36] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.
- [37] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.
- [38] Bernhard Riemann. On the hypotheses which lie at the bases of geometry. *Nature*, 1873.
- [39] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.
- [40] Hrittik Roy, Marco Miani, Carl Henrik Ek, Philipp Hennig, Marvin Pförtner, Lukas Tatzel, and Søren Hauberg. Reparameterization invariance in approximate bayesian inference. In *Neural Information Processing Systems (NeurIPS)*, 2024.
- [41] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323 (6088):533–536, 1986.



- [42] Aidan Scannell, Carl Henrik Ek, and Arthur Richards. Trajectory optimisation in learned multimodal dynamical systems via latent-ode collocation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12745–12751. IEEE, 2021.
- [43] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.
- [44] Stas Syrota, Yevgen Zainchkovskyy, Johnny Xi, Benjamin Bloem-Reddy, and Søren Hauberg. Identifying metric structures of deep latent variable models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research (PMLR), 2025.
- [45] Godfrey H Thomson. The proof or disproof of the existence of general ability. *British Journal of Psychology*, 9(3):321, 1919.
- [46] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.
- [47] Jakub M. Tomczak. *Deep Generative Modeling*. Springer, 2022.
- [48] Alessandra Tosi, Søren Hauberg, Alfredo Vellido, and Neil D. Lawrence. Metrics for probabilistic geometries. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014.
- [49] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations*. Springer, 2009.
- [50] Robert Edwin Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.
- [51] Quanhan Xi and Benjamin Bloem-Reddy. Indeterminacy in generative models: Characterization and strong identifiability. In *Artificial Intelligence and Statistics (AISTATS)*, Proceedings of Machine Learning Research (PMLR), 2023.
- [52] Tao Yang, Georgios Arvanitidis, Dongmei Fu, Xiaogang Li, and Søren Hauberg. Geodesic clustering in deep generative models. *arXiv preprint arXiv:1809.04747*, 2018.

## *Index*

- Abstract manifold, 45
- Addition
  - Euclidean, 78
  - Riemannian, 80
- Adjoint method, 166
- Ambient space, 21
- Angle, 40
- Antipodal, 54
- Atlas, 18, 97
- Autoencoder, 15, 49, 93, 118
- Automatic differentiation, 66, 72
- Average metric, 155
  
- Bayesian neural networks, 127
- Boundary value problem, 63
  
- Categorical distribution (information geometry), 106
- Chart, 96
- Christoffel symbols, 60, 62
- Classifier, 148
- Covariance, 129
- Covariance function, 130
  - exponentiated quadratic, 132
  - Gaussian, 132
  - squared exponential, 132
- Curve energy, 57, 58, 108, 137
  - expected, 137, 138
- Curve length, 32, 108
  - Euclidean, 29
  - Reparametrization, 30, 34
  - Riemannian, 32, 38, 45
- Curve velocity, 38
  
- Decoder, 49
- Deep generative model, 49
- Denoising SDE, 144
- Diffeomorphism, 161
- Diffusion model, 96, 144
- Direct energy minimization, 68
  
- Embedded manifold, 26
- Encoder, 49
- Energy minimization, 68
- Euler's method, 164
- Euler-Lagrange equation, 167
- Expected energy, 137, 138
- Expected inner product, 150
- Expected length, 140
- Expected metric, 139
- Exponential family, 112
- Exponential map, 80
  
- Fisher-Rao metric, 109, 114, 125
- Flat manifold, 103
- Fréchet mean, 87
- Function-space metric, 149
  
- Gaussian conditioning, 131
- Gaussian process, 130, 161
  - derivative, 132
  - manifold, 134
- Gaussian processes, 127
- Generalized Gauss-Newton matrix, 155
- Generative model, 10
- Generator transform, 103
- Geodesic, 33, 53

- computation, 68
  - connecting, 68
  - shooting, 72
- Geodesic ODE, 59, 72, 80
- Geodesic distance, 33
- Gradient descent, 111, 165
- Homeomorphism, 96
- Identifiability, 24, 25, 100, 101, 149
- Identifiability problem, 22, 35
- Image (manifold), 152
- Immersed manifold, 27
- Improper curve, 152
- Indeterminacy transformation, 101
- Information geometry, 105, 162
- Information metric, 151
- Initial value problem, 63, 164
- Injective function, 26
- Integration, 82
- Interpolant, 53
- Isometry, 17
- Isotropic metric, 65
- Jensen's inequality, 108
- Joint distribution, 129
- Kernel (manifold), 152
- Kernel density, 49
- Kernel manifold, 153
- Kernel ridge regression, 27
- KL divergence, *see* Kullback–Leibler divergence
- Kullback–Leibler divergence, 107, 109
- Latent variable model, 10
- Left inverse, 26
- Local invertibility, 26
- Logarithm map, 79, 80
- Loss function, 21
- Manifold, 25, 96
  - expected, 127
  - expected Gaussian, 127
  - flat, 103
  - Gaussian process, 134
  - hypothesis, 34
  - image, 153
  - kernel, 152
  - learning, 21, 118
  - noisy, 118, 121
  - parameter, 148
  - projected, 123
  - Riemannian, 44, 98
  - smooth, 97
  - statistical, 105
- Mean, 87
  - Fréchet, 87
- Mean function, 128
- Mean manifold, 136, 138
- Measure, 86
- Metric, *see* Riemannian metric
  - abstract, 48
  - average, 155
- Natural gradient, 110, 112, 116
- Natural parametrization, 114
- Noisy manifold, 121
- Normal distribution, 86
- Normalizing flow, 96
- Numerical curve length, 67
- ODE, *see* Ordinary differential equation
  - geodesic, 59, 72, 80

- Ordinary differential equation, 59
  - first-order, 163
  - geodesic, 59, 72, 80
  - system of, 164
- Overparametrization, 149
- Pattern theory, 160
- PCA, *see* Principal component analysis
- Picard-Lindelöf theorem, 63, 165
- Principal component analysis, 10, 12, 15, 25, 104
- Projected manifold, 123
- Projection
  - random, 121
- Projection (manifold), 49
- Projection (maps)
  - Mercator, 20
  - Peter, 20
- Proper curve, 153
- Pseudo-Riemannian metric, 51
- Random function, 127
- Random matrix, 121
- Random projection, 121
- Reconstruction error, 15
- Regression, 130
- Regularization, 119, 160
- Reparametrization, 25, 30
- Reparametrization issue, 22
- Riemann integral, 82, 83
- Riemann sum, 82
- Riemannian geometry, 29
- Riemannian manifold, 44, 98
- Riemannian metric, 38, 42, 44
  - ambient, 92
- Riemannian submanifold, 90
- Runge-Kutta solver, 165
- Score function, 144
- Semi-Riemannian metric, *see* Pseudo-Riemannian metric
- Shape statistics, 161
- Shooting geodesic, 63
- Smooth manifold, 97
- Spacetime geometry, 143
- Statistical divergence, 107
- Statistical identifiability, *see* Identifiability
- Statistical manifold, 105, 123
- Steepest descent, 111
- Stochastic differential equation, 143
  - denoising, 144
- Submanifold, 92
- Subtraction
  - Euclidean, 77
  - Riemannian, 79
- Tangent space, 26, 39
- Theorema Egregium, 17, 103
- Topological data analysis, 159
- Topology, 94
  - connectivity, 95
  - Euclidean, 95
  - point, 95
- VAE, *see* Variational autoencoder
- Variance, 87
- Variance function, 128
- Variational autoencoder, 23, 104, 120
- velocity, 38
- Wrapped distribution, 87



