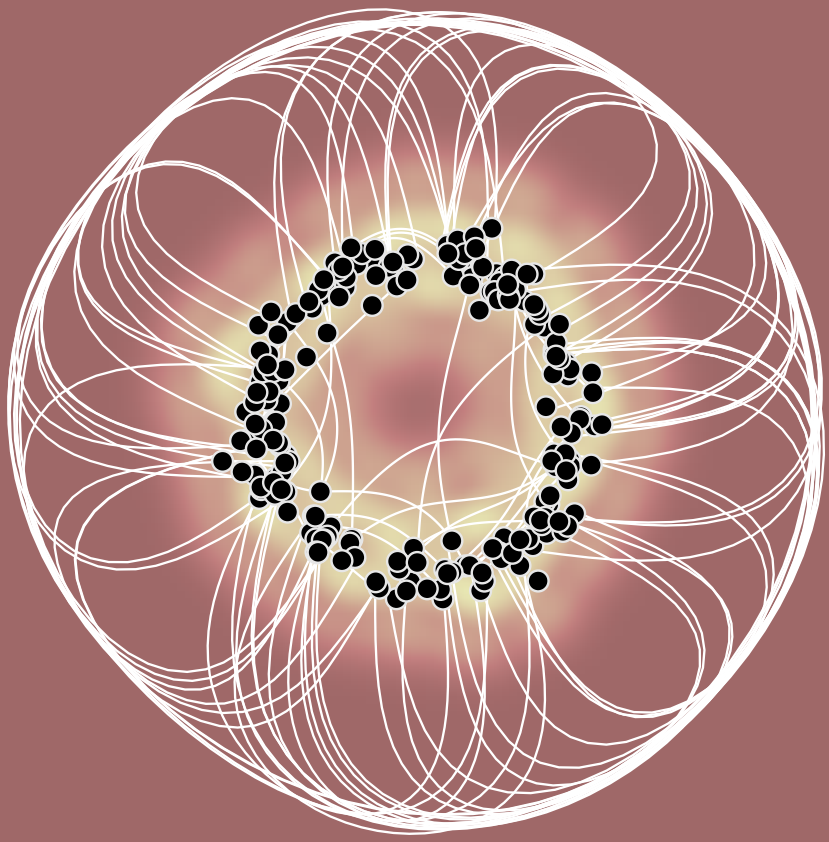
Differential geometry for generative modeling



Søren Hauberg

This version is printed on February 21, 2024.
For the latest version, please see http://hauberg.org/weekendwithbernie/

Author's note

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'. Differential geometry is all too often an inaccessible topic even if the core ideas are very intuitive. I have thus tried to make this text as easy as possible to digest...

That being said, I will assume that 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.

This document is a work-in-progress, but you are invited to improve it. Comments can be directed at `sohau@dtu.dk`

Mathematical notation

I have tried to limit 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 $\mathbb{R}$ that 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 $\mathbf{x} \in \Omega$.
$\mathsf{Sym}_{+}^{d \times d}$	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.
δ_n, δ_m	Infinitesimals, i.e. very small vectors.
$[\mathbf{x}]_i$	The i^{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 $\mathbf{x} \in \Omega$.
$\mathbf{G}_{\mathbf{x}} \in \mathsf{Sym}_+^{d \times d}$	The Riemannian metric $\mathbf{J}_{\mathbf{x}}^\intercal \mathbf{J}_{\mathbf{x}}$.
c	A curve of the form $c : [0, 1] \to \Omega$. 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. $\dot{c} = \mathsf{d}c/\mathsf{d}t$.
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 $\mathbf{x}$.

Contents

Chapter 1
Introduction Page 11

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

Chapter 2
Making maps Page 19

In which we pause to look at the world.

Chapter 3
Embedded and immersed manifolds Page 23

In which we begin to observe manifolds.

Chapter 4
Distances on manifolds Page 31

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

Chapter 5
Riemannian metrics Page 41

In which we measure from the inside out.

Chapter 6
Examples of learned metrics Page 51

In which we take an exemplary intermission.

Chapter 7
Geodesics Page 55

In which shortcuts are repeatedly taken.

Chapter 8
Computing geodesics numerically Page 67

In which we actually do something practical.

Chapter 9
Arithmetic on Riemannian manifolds Page 75

In which we joyfully rediscover plus and minus.

Chapter 10
Integration on manifolds Page 81

In which the metric speaks volumes about space.

Chapter 11
Submanifolds Page 87

In which recursion takes over (again).

Chapter 12
Noisy manifolds Page 91

In which data ruins the party only to randomly save it again.

Chapter 13
Parting remarks & further reading Page 105

In which our story ends, while many new begin.

Appendix A
Ordinary differential equations Page 109

Index Page 113

Bibliography Page 115

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 together will investigate in order 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.

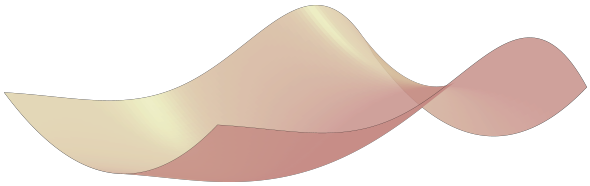


Figure 1.1: A simple two-dimensional manifold embedded in $\mathbb{R}^3$.

Geometry tells us how to measure; statistics rely on our choice of measure. This document touch on some geometric foundations that are useful when constructing statistical models. In particular, we will work towards understanding *Riemannian manifolds* as they commonly appear in machine learning.

The existing literature is focused on the mathematics of geometry, which can make the topic unnecessarily tricky to approach 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 in an effort on preserving

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...

∼ *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

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

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

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

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.

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* [27] model if we assume that there is an *affine* relationship between $\mathbf{x}$ and $\mathbf{y}$, i.e.

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

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

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

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

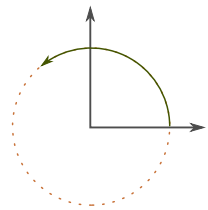
$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{b}, \mathbf{A}\mathbf{A}^\intercal + \sigma^2\mathbf{I}). \tag{1.5}$$

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 $\mathbf{x}$ are low-dimensional projections of the observations $\mathbf{y}$.

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 σ^2 are the maximum likelihood estimators of the model parameters, then we can create a new set of parameters that give identical likelihood. Let $\mathbf{R} \in \mathbb{R}^{d\times d}$ be a rotation matrix.[1] Now replace the (unobserved) latent variables with

[1] a rotation matrix is defined as satisfying $\mathbf{R}^\intercal\mathbf{R} = \mathbf{I}$



$$\hat{\mathbf{x}} = \mathbf{R}\mathbf{x} \tag{1.6}$$

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 $\hat{\mathbf{A}} = \mathbf{A}\mathbf{R}^\intercal$ and let

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

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

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

which is identical the original model (1.5) as $\mathbf{R}^\intercal\mathbf{R} = \mathbf{I}$. 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.

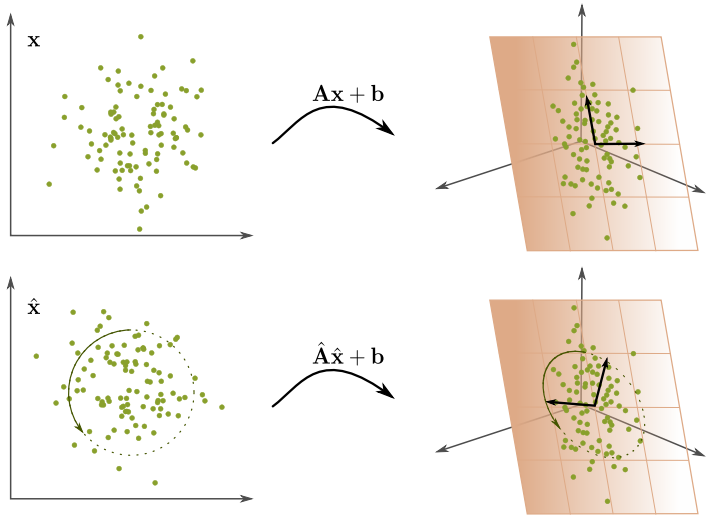


Figure 1.2: In probabilistic principal component analysis, we can get two equally good models by rotating the latent variable of one model and adapting the affine mapping correspondingly.

Deep generative models

Instead of an affine model, as in probabilistic PCA, we can also consider general nonlinear functions, such as neural networks. This results, for example, in probabilistic models of the form

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

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

That is, the latent variable is mapped through a function $f : \mathbb{R}^d \to \mathbb{R}^D$, 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 $h : \mathbb{R}^d \to \mathbb{R}^d$ with the property

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

One example of such a function is

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

where $\mathbf{R}_\theta$ is a linear transformation that rotates points by $\theta(\mathbf{x}) = \sin(\pi\|\mathbf{x}\|)$. This is illustrated in Fig. 1.3. Note that we could have made other choices of h.

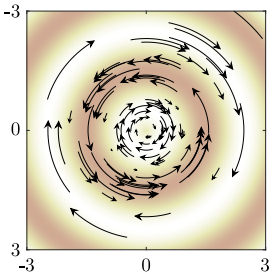


Figure 1.3: A transformation of space that preserves distances to the origin.

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 $\hat{\mathbf{x}} = h(\mathbf{x})$,

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

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

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.

∼ *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 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 *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.

∼ *Exercises* ∽

Exercise 1.1: PPCA distances and angles.
Probabilistic PCA can be equivalently expressed as

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

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

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

(with latent variables $\hat{\mathbf{x}} = \mathbf{R}\mathbf{x}$ and $\hat{\mathbf{A}} = \mathbf{A}\mathbf{R}^\intercal$, 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 $g_\theta : \mathbb{R}^D \to \mathbb{R}^d$ be a neural network with weight parameters θ, and let $f_\psi : \mathbb{R}^d \to \mathbb{R}^D$ be another network with parameters ψ. 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*

$$\theta^*, \psi^* = \operatorname*{argmin}_{\theta,\psi} \sum_{n=1}^{N} \|f_\psi(g_\theta(\mathbf{y}_n)) - \mathbf{y}_n\|^2 . \tag{1.17}$$

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.

Figure 2.1: Planet Earth. *Image courtesy of NASA.*

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.*[1]

[1] Translates to *remarkable theorem*, which indicates the level of excitement from Gauss upon his discovery.

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

Figure 2.2: *The second Borgian map* by Diego Ribero, Sevilla 1529. While beautiful, the map is fundamentally misleading as any *flat* depiction of a *curved* object (here: Earth) will not reflect distances accurately.

takes 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 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, then we cannot make this flat without getting a distorted picture. This is the fundamental challenge, that geometry solves.

∼ *Exercises* ∽

Exercise 2.1: Maps of the world.

Humans have long aimed to map their surroundings in order to reliably navigate. The resulting maps have often been used to form mental pictures of 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 does 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 $\mathbf{y}_1, \ldots, \mathbf{y}_N$ 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 is inherent of lower dimension than the space in which we made our observations.

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

$$f : \mathbb{R}^d \to \mathbb{R}^D, \qquad d \leq D \tag{3.1}$$

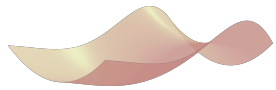


Figure 3.1: A manifold spanned by a function $f : \mathbb{R}^2 \to \mathbb{R}^3$.

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 $\mathbb{R}^D$. Our interest will predominantly be cases where f is nonlinear, though, as this will allow our models to capture more variability in the data.

∼ *Manifold learning* ∽

In order 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,

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

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

Here two important points must be raised.

First, we have parametrized the mapping f by a vector θ. For instance, if we have chosen f to be polynomial, then θ will contain the corresponding coefficients, or if f is a neural network, θ 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 whichever 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 $h : \mathbb{R}^d \to \mathbb{R}^d$, and assume that $\{f, \mathbf{x}_1, \dots, \mathbf{x}_N\}$ span an optimal manifold (according to our loss function), then

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

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

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

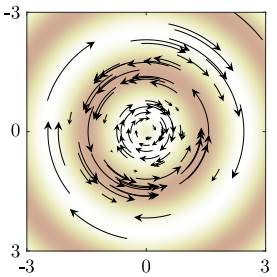


Figure 3.2: An example of a *reparametrization* given by a function $h : \mathbb{R}^2 \to \mathbb{R}^2$. This figure is a repetition of Fig. 1.3.

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

∼ *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)* [16, 22] 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 [23, 11]. 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.

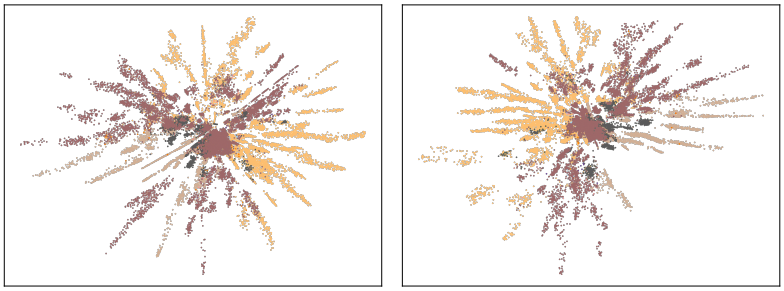


Figure 3.3: The low-dimensional representations $\mathbf{x}_n$ of two variational autoencoders fitted to the same data. Here each point corresponds to one protein from the beta-lactamase family. The two different representations may result in different biological interpretations.

∼ *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.

∼ *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

$$f : \Omega \to \mathbb{R}^D, \tag{3.6}$$

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 Ω under f, i.e. $\mathcal{M} = f(\Omega)$. That's it for now. We will revisit this definition later when it turns 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.

∼ *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. 3.4, which is described by a function $f : [a, b] \to \mathbb{R}^2$. 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

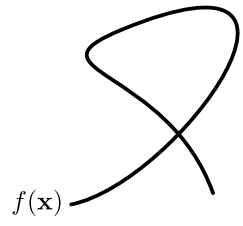


Figure 3.4: The curve $f(\mathbf{x}) \in \mathbb{R}^2$ self-intersect and is *not* an embedded manifold.

is *not* invertible at this point. Consequently, this manifold is *not* embedded.

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[1] 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.

[1] Technically, we refer to this as the *left inverse* to emphasize that f is only invertible on its image.

∼ *Immersed manifolds* ∽

Practically, it is too restrictive to only study embedded manifolds, and we, therefore, open up for allowing functions f that is only *locally invertible*. That is, if the Jacobian $\mathbf{J} = {}^{\partial f}/_{\partial \mathbf{x}}$ has full rank for all $\mathbf{x} \in \Omega$, 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. 3.5 below). 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.

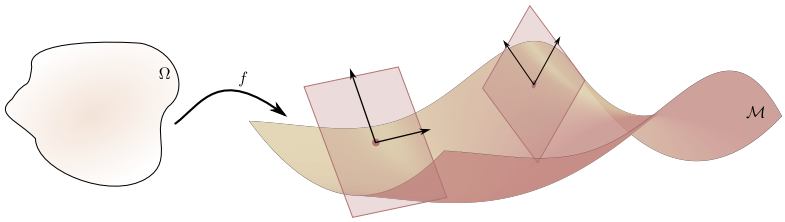


Figure 3.5: The Jacobian of f can be interpreted as giving a basis for a *tangential plane* to the manifold. When the Jacobian has full rank then this tangential plane has the same dimensionality as the manifold (i.e. the manifold does not collapse anywhere)

∼ Exercises ∽

Exercise 3.1: You really should, you know. . . .
If you have not 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* [26] will, for most kernels, extrapolate to 0 (see Fig. 3.6 for an example). Assume a generative model $\mathbf{y} = f(\mathbf{x})$, where

$$f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_D(\mathbf{x}) \end{pmatrix} \tag{3.7}$$

consist of D functions $f_i : \mathbb{R}^d \to \mathbb{R}$.

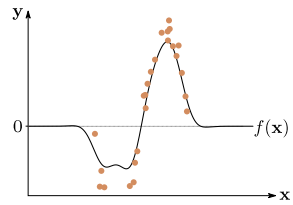


Figure 3.6: The regression curve $f(\mathbf{x})$ of a kernel ridge regressor extrapolates to zero.

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

$$f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} 2x_1^2 + x_2^2 \\ x_1 \\ x_2 \end{pmatrix}. \tag{3.8}$$

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

$$f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \cos(x_1) \\ \sin(x_1) \\ \cos(x_2) \end{pmatrix}. \tag{3.9}$$

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.*

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.

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

$$\{c(0), c(t_1)\}, \{c(t_1), c(t_2)\}, \ldots, \{c(t_N), c(1)\}, \tag{4.1}$$

where $t_n \in]0, 1[, n = 1, \ldots, N$. 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.

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

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

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

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

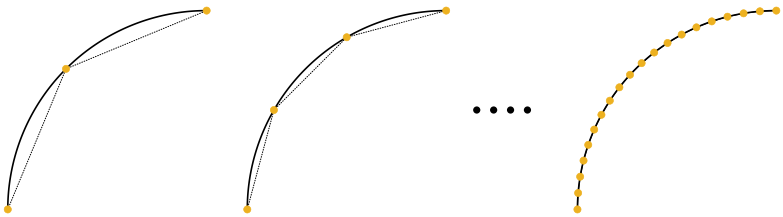


Figure 4.1: The length of a curve (4.5) can be understood as "cutting up" the curve in small linear segments and adding up their lengths.

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

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

$$= \int_0^1 \left\| \frac{\mathsf{d}c(t)}{\mathsf{d}t} \right\| \mathsf{d}t, \tag{4.5}$$

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.

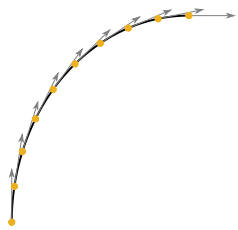


Figure 4.2: The length of a curve can also be expressed by integrating the curve derivative, which avoids the use of a limit.

∼ *Reparametrization issues* ∽

A reoccurring 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 $c : [0, 1] \to \mathbb{R}^D$, then its length is given by Eq. 4.5 above. Now we introduce a smooth monotonic function $h : [0, 1] \to [0, 1]$ with the properties

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

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.

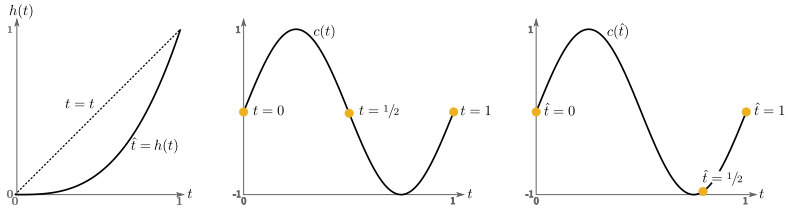


Figure 4.3: A sine curve under two parametrizations. While the points generated along the curve are identical in the two parametrizations, we get different positions along the curve for a fixed input.

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

$$\begin{aligned}
\text{Length}(c) &= \int_0^1 \left\| \frac{\mathrm{d}c}{\mathrm{d}t} \right\| \mathrm{d}t && \text{(Definition of curve length)} \\
&= \int_0^1 \left\| \left(\frac{\mathrm{d}c}{\mathrm{d}t} \right)_{t=h(s)} \right\| \frac{\mathrm{d}h}{\mathrm{d}s} \mathrm{d}s && t \to h(s) \\
&= \int_0^1 \left\| \left(\frac{\mathrm{d}c}{\mathrm{d}t} \right)_{t=h(s)} \frac{\mathrm{d}h}{\mathrm{d}s} \right\| \mathrm{d}s && {}^{\mathrm{d}h}/_{\mathrm{d}s} > 0 \\
&= \int_0^1 \left\| \frac{\mathrm{d}c}{\mathrm{d}s} \right\| \mathrm{d}s && \text{(Chain rule)} \\
&= \text{Length}(\hat{c}). && (4.7)
\end{aligned}$$

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,[1] 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.

[1] e.g. we may seek a curve that is as short as possible

∼ *Measuring on immersed manifolds* ∽

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

Let $c : [0, 1] \to \Omega$ be a smooth curve in Ω 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.*

$$\text{Length}_{\mathcal{M}}(c) \stackrel{\text{def}}{=} \text{Length}(f(c)) = \int_0^1 \left\| \frac{\mathsf{d}}{\mathsf{d}t} f(c_t) \right\| \mathsf{d}t, \tag{4.8}$$

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

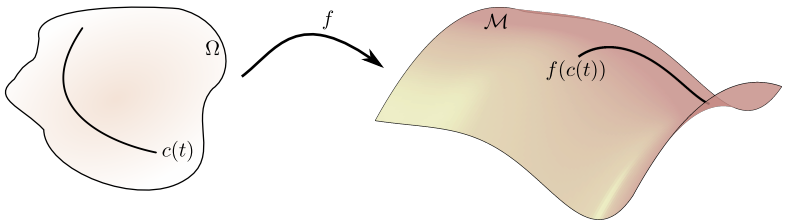


Figure 4.4: The length of a curve on a manifold is simply the Euclidean length of the immersed curve.

Given two points $c_0, c_1 \in \Omega$, we can define the shortest connecting path as[2]

$$\begin{aligned} c_{01} = &\operatorname*{argmin}_{c} \text{Length}_{\mathcal{M}}(c) \\ &\text{subject to } c(0) = c_0 \text{ and } c(1) = c_1. \end{aligned} \tag{4.9}$$

[2] 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.

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,

$$\mathsf{dist}(c_0, c_1) = \text{Length}_{\mathcal{M}}(c_{01}). \tag{4.10}$$

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

$$\mathsf{dist}(x, y) = 0 \quad \Leftrightarrow \quad x = y \tag{4.11}$$

$$\mathsf{dist}(x, y) = \mathsf{dist}(y, x) \tag{4.12}$$

$$\mathsf{dist}(x, y) \leq \mathsf{dist}(x, z) + \mathsf{dist}(z, y) \tag{4.13}$$

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 then cannot be the globally shortest path as a shorter path exists. The third requirement[3] 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.

[3] Also known as the triangle inequality.

∼ *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, the distance definition does not care about curve parametrization. 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

$$\Omega \subset \mathbb{R}^d \tag{4.14}$$

$$f : \Omega \to \mathbb{R}^D, \quad d < D \tag{4.15}$$

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

$$\hat{\Omega} = h(\Omega) \subset \mathbb{R}^d \tag{4.16}$$

$$\hat{f}(\mathbf{x}) = f\left(h^{-1}(\mathbf{x})\right). \tag{4.17}$$

Then we have

$$\hat{f}(\hat{\Omega}) = f(h^{-1}(g(\Omega))) \tag{4.18}$$
$$= f(\Omega) \tag{4.19}$$
$$= \mathcal{M}. \tag{4.20}$$

In words, $\hat{\Omega}$ and $\hat{f}$ define the exact same manifold as Ω 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

$$\mathbf{y} = f(\mathbf{x}) + \epsilon, \tag{4.21}$$

where ϵ 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 $\mathbf{x}_{1:N} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ 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 [17]. 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[4] change under different parametrizations, i.e. in general

$$\|\mathbf{x}_n - \mathbf{x}_m\| \neq \|h(\mathbf{x}_n) - h(\mathbf{x}_m)\|. \quad \text{(true in general, but exceptions exist)}$$

[4] The exception is when h performs a rotation, which does preserve distances.

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

$$c(0) = \mathbf{x}_n \quad \text{and} \quad c(1) = \mathbf{x}_m, \tag{4.22}$$

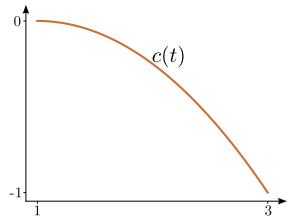
then the corresponding curve along the manifold is $t \mapsto f(c(t))$. If we reparametrize manifold by h, then c become $t \mapsto h(c(t))$,which will be immersed as $t \mapsto f(h^{-1}(h(c(t)))) = f(c(t))$. This implies that the length of a curve (4.5) does not depend on the parametrization of the 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 $c : [0, 1] \to \mathbb{R}^2$ defined as (see plot in the margin)

$$c(t) = \begin{pmatrix} 2t + 1 \\ -t^2 \end{pmatrix}. \tag{4.23}$$



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

$$\int \sqrt{1 + t^2} \mathrm{d}t = \frac{1}{2}\left(\sqrt{1 + t^2}\ t \sinh^{-1}(t)\right) + C, \tag{4.24}$$

for some constant C.

Exercise 4.2: Numerical computation of Euclidean curve lengths.

Consider the curve $c : [0, 1] \to \mathbb{R}^2$ defined in Eq. 4.23.

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 35, a proof sketch is provided for why the geodesic distance (4.10) is a valid distance measure. This is somewhat 'handwavy'.

1. *Prove that Eq. 4.11 holds for the geodesic distance.*
2. *Prove that Eq. 4.12 holds for the geodesic distance.*
3. *Prove that Eq. 4.13 holds for the geodesic distance.*

Exercise 4.4: Manifold curve lengths.

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

$$f(x_1, x_2) = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{pmatrix}. \tag{4.25}$$

Let $c(t) = \mathbf{v}t$ 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*

$$\text{Length}_{\mathcal{M}}(c) = \|\mathbf{v}\| \int_0^1 \sqrt{1 + t^2}\mathrm{d}t. \tag{4.26}$$

Exercise 4.5: Reparametrizations.
Consider the manifold spanned by the function $f : \mathbb{R}^2 \to \mathbb{R}^3$ defined by Eq. 4.25 (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 $h : \mathbb{R}^2 \to \mathbb{R}^2$ defined as

$$h(x_1, x_2) = \begin{pmatrix} x_1^3 \\ x_2^3 \end{pmatrix}. \tag{4.27}$$

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. 4.8

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

$$= \int_0^1 \|\mathbf{J}_{c_t} \dot{c}_t\| \, \mathsf{d}t \tag{5.2}$$

$$= \int_0^1 \sqrt{\dot{c}_t^\mathsf{T} \mathbf{J}_{c_t}^\mathsf{T} \mathbf{J}_{c_t} \dot{c}_t} \mathsf{d}t, \tag{5.3}$$

where $\dot{c}_t = {}^{\mathsf{d}c}/_{\mathsf{d}t}|_{t=t}$ is the curve derivative (*velocity*) 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.

∼ *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

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

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

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

$$\mathbf{x}, \epsilon \in \mathbb{R}^d \tag{5.6}$$

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

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 $\mathbf{x}$.

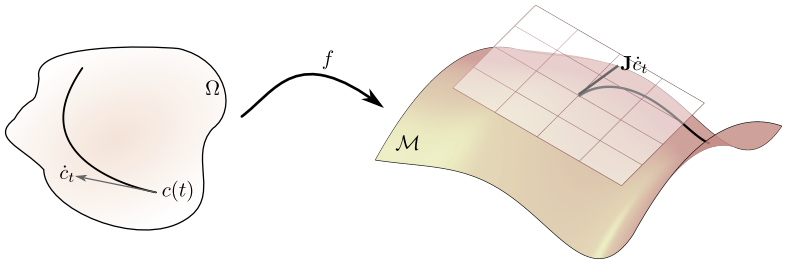


Figure 5.1: A manifold is locally Euclidean, which is reflected by the tangent space of our manifold. For immersed manifolds, the Jacobian of f provides us with a basis for this tangent space.

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.

∼ *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,

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

we can heuristically write this as

$$= \int_0^1 \|\mathbf{J}_{c_t} \left(\dot{c}_t \, \mathsf{d}t\right)\| \, . \qquad \text{(Heuristic notation)}$$

That is, we may think of $\dot{c}_t \, \mathrm{d}t$ 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 δ_1 and δ_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,

$$\langle \delta_1, \delta_2 \rangle_{\mathbf{x}} = (\mathbf{J}_{\mathbf{x}} \delta_1)^{\mathsf{T}} (\mathbf{J}_{\mathbf{x}} \delta_2) = \delta_1^{\mathsf{T}} \mathbf{J}_{\mathbf{x}}^{\mathsf{T}} \mathbf{J}_{\mathbf{x}} \delta_2. \tag{5.9}$$

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

$$\begin{aligned} &\langle f(\mathbf{x} + \delta_1) - f(\mathbf{x}), f(\mathbf{x} + \delta_2) - f(\mathbf{x}) \rangle && (5.10) \\ &\quad = (f(\mathbf{x}) + \mathbf{J}_{\mathbf{x}} \delta_1 - f(\mathbf{x}))^{\mathsf{T}} (f(\mathbf{x}) + \mathbf{J}_{\mathbf{x}} \delta_2 - f(\mathbf{x})) && (5.11) \\ &\quad = \delta_1^{\mathsf{T}} \mathbf{J}_{\mathbf{x}}^{\mathsf{T}} \mathbf{J}_{\mathbf{x}} \delta_2. && (5.12) \end{aligned}$$

∼ *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,

$$\begin{aligned} \|\mathbf{v}\|_{\mathbf{x}} &= \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\mathbf{x}}} && (5.13) \\ &= \|\mathbf{J}\mathbf{v}\|. && (5.14) \end{aligned}$$

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

$$\cos\theta = \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{\|\mathbf{v}_1\| \cdot \|\mathbf{v}_2\|}, \tag{5.15}$$

where θ 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, respectfully, 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.

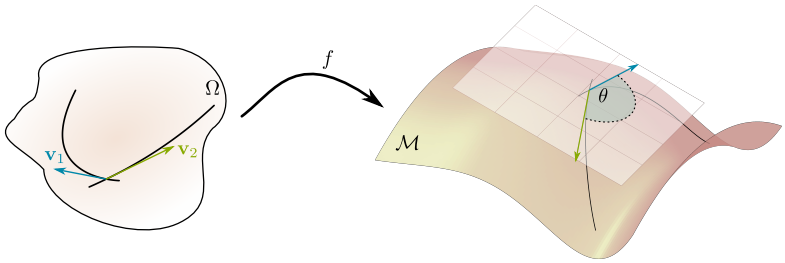


Figure 5.2: The angle between two curves can be computed in the tangent space of the point of intersection as the angle between velocity vectors.

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

$$c_1(t_1) = \mathbf{x} = c_2(t_2). \tag{5.16}$$

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

$$\mathbf{v}_1 = \dot{c}_1(t_1) \tag{5.17}$$

$$\mathbf{v}_2 = \dot{c}_2(t_2), \tag{5.18}$$

where $\dot{c} = {}^{\mathrm{d}c}/_{\mathrm{d}t}$ is short-hand for curve derivatives. The angle between the curves then becomes

$$\theta = \cos^{-1}\left(\frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{\|\mathbf{v}_1\|_{\mathbf{x}} \cdot \|\mathbf{v}_2\|_{\mathbf{x}}}\right) \tag{5.19}$$

$$= \cos^{-1}\left(\frac{\mathbf{v}_1^{\intercal}\mathbf{J}_{\mathbf{x}}^{\intercal}\mathbf{J}_{\mathbf{x}}\mathbf{v}_2}{\|\mathbf{J}_{\mathbf{x}}\mathbf{v}_1\| \cdot \|\mathbf{J}_{\mathbf{x}}\mathbf{v}_2\|}\right). \tag{5.20}$$

∼ *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 $\mathbf{J}^\intercal\mathbf{J}$, e.g. as with inner products

$$\langle \delta_1, \delta_2 \rangle_{\mathbf{x}} = \delta_1^\intercal \mathbf{J}_{\mathbf{x}}^\intercal \mathbf{J}_{\mathbf{x}} \delta_2. \qquad \text{as defined in Eq. 5.9}$$

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

$$\mathbf{G}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^\intercal \mathbf{J}_{\mathbf{x}}, \tag{5.21}$$

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

$$\langle \delta_1, \delta_2 \rangle_{\mathbf{x}} = \delta_1^\intercal \mathbf{G}_{\mathbf{x}} \delta_2. \tag{5.22}$$

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.

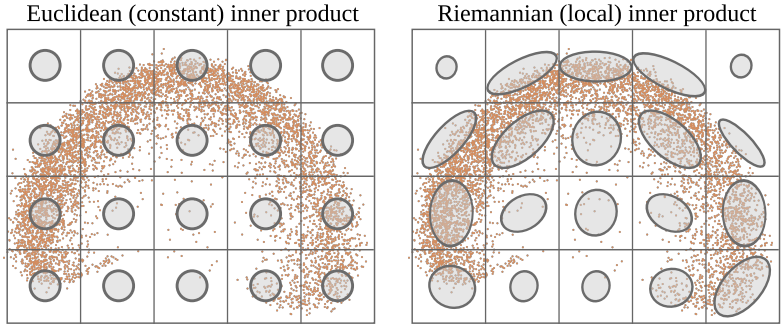


Figure 5.3: In a Euclidean view, the inner product structure is the same everywhere in space. In the Riemannian view, this changes locally. The inner product is such that local unit circles have an ellipsoidal shape.

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 $\mathbf{G}$ become positive definite[1] by construction. We can also show this directly by forming a singular value decomposition (SVD) of $\mathbf{J}$ as

[1] i.e. all eigenvalues are strictly positive

$$\begin{aligned} \mathbf{J} =& \mathbf{U}\mathbf{S}\mathbf{V}^\intercal, \\ \text{where } \mathbf{U} \in \mathbb{R}^{D\times D}, &\mathbf{S} \in \mathbb{R}^{D\times d} \text{ and } \mathbf{V} \in \mathbb{R}^{d\times d}. \end{aligned} \tag{5.23}$$

Here $\mathbf{U}$ and $\mathbf{V}$ are rotations, i.e. $\mathbf{U}^\intercal\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^\intercal\mathbf{V} = \mathbf{I}$, 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

$$\begin{aligned} \mathbf{G} &= \mathbf{J}^\intercal\mathbf{J} && (5.24)\\ &= (\mathbf{U}\mathbf{S}\mathbf{V}^\intercal)^\intercal (\mathbf{U}\mathbf{S}\mathbf{V}^\intercal) && (5.25)\\ &= \mathbf{V}\mathbf{S}^\intercal\mathbf{U}^\intercal\mathbf{U}\mathbf{S}\mathbf{V}^\intercal && (5.26)\\ &= \mathbf{V}\Lambda\mathbf{V}^\intercal, && (5.27) \end{aligned}$$

where $\Lambda = \mathbf{S}^\intercal\mathbf{S} \in \mathbb{R}^{d\times d}$ 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.

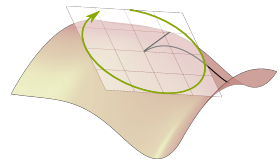


Figure 5.4: The particular choice of basis of our tangent space is rarely all that important. What matters is that we get norms and angles correct.

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?

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.* $\mathbf{R}^\intercal\mathbf{R} = \mathbf{I}$) such that $\tilde{\mathbf{J}} = \mathbf{RJ}$ spans the same tangent space. The corresponding metric

$$\tilde{\mathbf{G}} = \tilde{\mathbf{J}}^\intercal\tilde{\mathbf{J}} = (\mathbf{RJ})^\intercal\mathbf{RJ} = \mathbf{J}^\intercal\mathbf{R}^\intercal\mathbf{RJ} = \mathbf{G} \tag{5.28}$$

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 the metric allows for elegant interpretations of computations on manifolds.

∼ *Riemannian metrics and manifolds* ∽

We can view the metric as a function

$$\mathbf{G}_\mathbf{x} : \Omega \to \mathsf{Sym}_+^{d\times d}, \tag{5.29}$$

where $\mathsf{Sym}_+^{d\times d}$ 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.

∼ *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

$$\langle \delta_1, \delta_2 \rangle_{\mathbf{x}} = \delta_1^\intercal \mathbf{G}_{\mathbf{x}} \delta_2 \qquad \text{local inner product (5.9)}$$

$$\|\mathbf{v}\|_{\mathbf{x}} = \sqrt{\mathbf{v}^\intercal \mathbf{G}_{\mathbf{x}} \mathbf{v}} \qquad \text{local norm (5.14)}$$

$$\theta = \cos^{-1}\left(\frac{\mathbf{v}_1^\intercal \mathbf{G}_{\mathbf{x}} \mathbf{v}_2}{\sqrt{\mathbf{v}_1^\intercal \mathbf{G}_{\mathbf{x}} \mathbf{v}_1} \cdot \sqrt{\mathbf{v}_2^\intercal \mathbf{G}_{\mathbf{x}} \mathbf{v}_2}} \right) \qquad \text{angle between vectors (5.20)}$$

$$\text{Length}_{\mathcal{M}}(c) = \int_0^1 \sqrt{\dot{c}_t^\intercal \mathbf{G}_{c_t} \dot{c}_t} \mathrm{d}t. \qquad \text{curve length (4.5)}$$

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

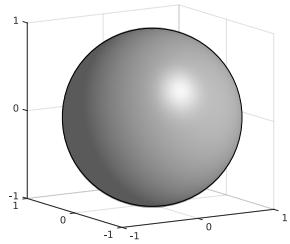
Since we can compute all elementary quantities of interest from the metric alone, it is worth asking if we can forget all about f and $\mathbf{J}$ and only care about the metric. This leads to the notion of *abstract manifolds*, which are simply manifolds that come with a metric, but where an explicit 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 $f : (0, \pi) \times [0, 2\pi) \to \mathbb{R}^3$



$$f(x_1, x_2) = \begin{pmatrix} \sin(x_1)\cos(x_2) \\ \sin(x_1)\sin(x_2) \\ \cos(x_1) \end{pmatrix}. \tag{5.30}$$

This will span the unit sphere.

1. *Derive the Jacobian matrix of f.*
2. *Derive the metric associated with f according to Eq. 5.21.*
3. *Show that the metric is positive definite.*

Exercise 5.2: Quadratic metric.

Consider a two-dimensional abstract manifold with the metric

$$\mathbf{G}_\mathbf{x} = \left(1 + \|\mathbf{x}\|^2\right)\mathbf{I}, \qquad \mathbf{x} \in \mathbb{R}^2. \tag{5.31}$$

Consider the points

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \mathbf{x}_2 = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \mathbf{x}_3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}. \tag{5.32}$$

1. *Compute the local norms of the tangent vector $\mathbf{v}_1 = \begin{pmatrix} 1 & 0 \end{pmatrix}^\intercal$ 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 $\mathbf{v}_2 = \begin{pmatrix} 0 & 1 \end{pmatrix}^\intercal$ in the same three points of tangency.*

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.

∼ *Abstract density metrics* ∽

Our general focus, thus far, has been on models where the manifold is defined through a mapping $f : \mathbb{R}^d \to \mathbb{R}^D$, but one can also learn the metric directly. Consider a data set $\mathbf{y}_{1:N} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, 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 $g : \mathbb{R}^d \to \mathbb{R}_+$ 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

[18], i.e.

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

Here $p(\mathbf{y})$ is any smooth estimate of the density, e.g. a *kernel density* estimate [13] or even a *deep generative model* [28]. 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.

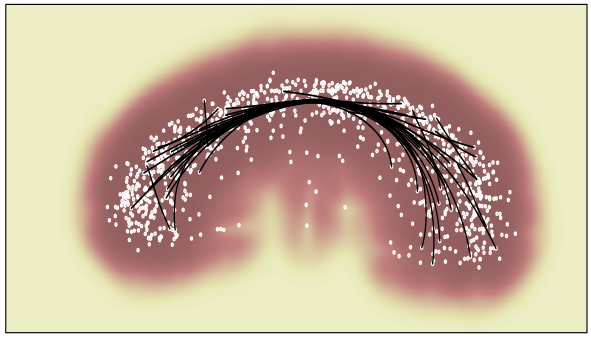


Figure 6.1: Direct learning of a Riemannian metric. The background color reflects the learned function g as in Eq. 6.2. As g takes small values near the data, geodesics are naturally drawn towards regions of high density.

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 [18, 15, 3].

∼ *Autoencoder metrics* ∽

The *autoencoder* [24] is a prototypical example of the manifolds discussed thus far. Briefly put, this model class defines a manifold through a *decoder* function $f : \mathbb{R}^d \to \mathbb{R}^D$ and approximates the *projection* onto the manifold with an *encoder* function $g : \mathbb{R}^D \to \mathbb{R}^d$. This pair of functions is then jointly trained by minimizing the squared distance between observations and their approximate projections onto the manifold,

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

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 12 when discussing noisy manifolds.

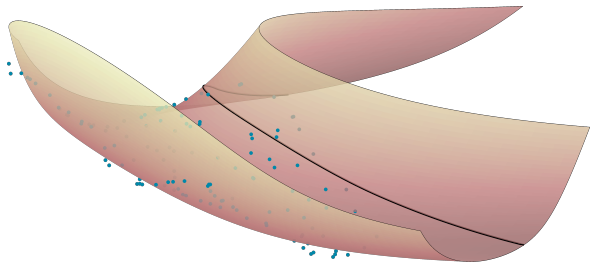


Figure 6.2: A manifold learned from data (blue) and an example geodesic (black). Note how the geodesic avoids 'climbing the steep hill' of the manifold.

∼ *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.

$$f : \mathbb{R}^D \to C, \tag{6.4}$$

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 (5.21)

$$\mathbf{G}_\mathbf{x} = \mathbf{J}_\mathbf{x}^\intercal \mathbf{J}_\mathbf{x}, \tag{6.5}$$

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 $\min(D, K)$, we see that the number of classes cannot be smaller 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.

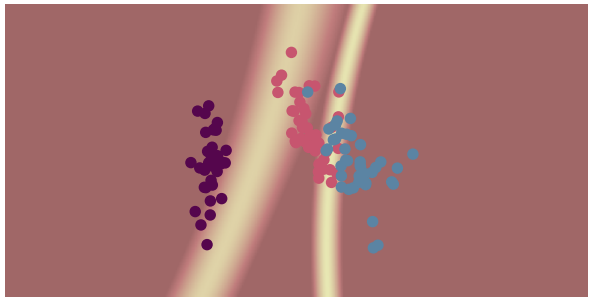


Figure 6.3: The (square root determinant of the) pull-back metric associated with a neural network performing classification on a two-dimensional projection of the Iris dataset. The metric is then over the space of observed data.

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?*). One way to mentally proceed is to think of regions of input space where all points have a distance of zero to each other as just being a single point. Here we could easily start an exciting intellectual journey, but instead, we simply stop and call such beyond the scope of this short text.

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).

∼ *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.

∼ *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 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 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.

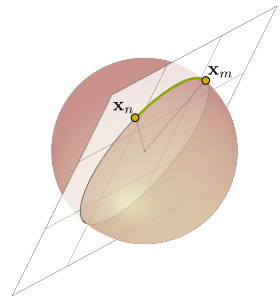


Figure 7.1: Geodesics on a sphere may be found as the intersection of a plane and the sphere.

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. 7.2). 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 (Fig. 7.3). This implies that infinitely many geodesics exist that connect antipodal points on the sphere. From this example, we conclude that in general, we cannot expect geodesics to be unique.

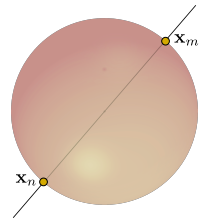


Figure 7.2: Antipodal points do not span a plane, and the geodesic is not unique...

Since geodesics are defined as the shortest connecting curves,

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

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.

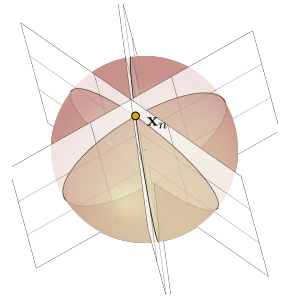


Figure 7.3: ...instead we may pick *any* plane containing the antipodal points, such that we get infinitely many geodesics.

∼ *Constant speed parametrizations* ∽

So, geodesics are not unique. To further add to the lack of non-uniqueness we have previously seen (page 32) 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 7.4 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

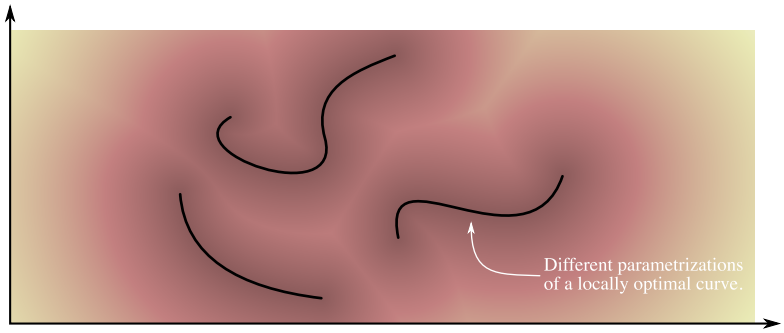


Figure 7.4: The optimization landscape from curve length minimization. Here coordinates are different curve parameters. Due to reparametrization issues each local optimum forms a continuum (black curves) along which corresponding curves have the same shape and length.

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 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 $c : [0, 1] \to \mathbb{R}^D$ with length

$$\text{Length}_{\mathcal{M}}[c] = \int_0^1 \|\dot{c}_t\| \mathsf{d}t, \tag{7.2}$$

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

$$\text{Length}_{\mathcal{M}}[c] = \int_0^1 v \mathsf{d}t \tag{7.3}$$

$$= v \int_0^1 \mathsf{d}t = v. \tag{7.4}$$

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.

∼ *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

$$|\langle u, v\rangle| \leq \|u\| \ \|v\|, \tag{7.5}$$

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

$$\langle u, v\rangle = \int_0^1 u_t v_t \mathsf{d}t \tag{7.6}$$

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

$$u_t = \|\dot{c}_t\|. \tag{7.7}$$

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

$$v_t = 1. \tag{7.8}$$

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

$$\text{Length}_{\mathcal{M}}[c] = \int_0^1 \|\dot{c}_t\| \mathsf{d}t = \langle u, v \rangle. \tag{7.9}$$

Then by the Cauchy-Schwarz inequality, we have

$$\begin{aligned} \text{Length}_{\mathcal{M}}[c] &\leq \|u\| \ \|v\| && (7.10) \\ &= \sqrt{\int_0^1 \|\dot{c}_t\|^2 \mathsf{d}t} \ \sqrt{\int_0^1 1^2 \mathsf{d}t} && (7.11) \\ &= \sqrt{\int_0^1 \|\dot{c}_t\|^2 \mathsf{d}t}. && (7.12) \end{aligned}$$

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

$$\mathcal{E}[c] = \int_0^1 \|\dot{c}_t\|^2 \mathsf{d}t. \tag{7.13}$$

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

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

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,

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

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. 7.5 vs. Fig. 7.4).

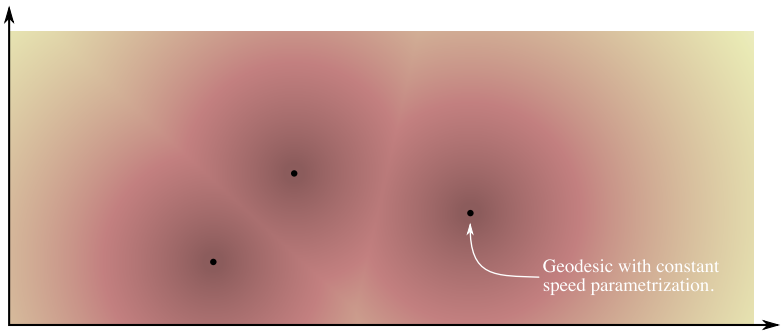


Figure 7.5: The optimization landscape from curve energy minimization. This figure should be contrasted with that of length minimization (Fig. 7.4). Now optima are well isolated and correspond to constant speed curves.

∼ *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 (*ODE*s)* 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

$$\int_0^1 E(t, c_t, \dot{c}_t)\mathsf{d}t, \tag{7.16}$$

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

$$\frac{\partial E_t}{\partial c_t} = \frac{\mathsf{d}}{\mathsf{d}t}\frac{\partial E_t}{\partial \dot{c}_t}. \tag{7.17}$$

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

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

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

$$[\ddot{c}_t]_j = -\sum_{i=1}^{d}\sum_{k=1}^{d} \Gamma_j^{ik}(c_t)\,[\dot{c}_t]_k\,[\dot{c}_t]_j \tag{7.19}$$

$$\Gamma_j^{ik}(c_t) = \frac{1}{2}\left[\mathbf{G}_{c_t}^{-1}\right]_{ij}\left(2\frac{\partial\,[G_{c_t}]_{ij}}{\partial\,[c_t]_k} - \frac{\partial\,[G_{c_t}]_{jk}}{\partial\,[c_t]_i}\right), \tag{7.20}$$

where $[\cdot]_i$ denotes an indexing into the i^{th} dimension of a vector, and similarly with matrix indexing $[\cdot]_{ij}$. Clearly, this is a notional mess, but such is the situation. Here the $\Gamma_j^{ik}(c_t)$ coefficients (informally) reflect how much the metric changes relative to the metric itself. This is indicative of how much the manifold curves at a given point.

∼ *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,

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

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

$$\frac{\mathsf{d}E_t}{\mathsf{d}[c_t]_i} = \sum_{j=1}^{d}\sum_{k=1}^{d} \frac{\mathsf{d}[\mathbf{G}_{c_t}]_{jk}}{\mathsf{d}[c_t]_i}[\dot{c}_t]_j[\dot{c}_t]_k. \tag{7.22}$$

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

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

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

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

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

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

$\frac{\mathsf{d}}{\mathsf{d}t}ab = \left(\frac{\mathsf{d}}{\mathsf{d}t}a\right)b + a\left(\frac{\mathsf{d}}{\mathsf{d}t}b\right)$

$\frac{\mathsf{d}}{\mathsf{d}t}f\left(g_t^{(1)},\ldots,g_t^{(d)}\right) = \sum_{k=1}^{d}\frac{\mathsf{d}g^{(k)}}{\mathsf{d}t}\frac{\mathsf{d}f\left(g_t^{(1)},\ldots,g_t^{(d)}\right)}{\mathsf{d}g_t^{(k)}}$

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

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

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

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

$[\mathbf{A}^{-1}\mathbf{b}]_i = \sum_{j=1}^{d}[\mathbf{A}^{-1}]_{ij}[\mathbf{b}]_j$

If we introduce the short-hand notation

$$\Gamma_j^{ik}(c_t) = \frac{1}{2}\left[\mathbf{G}_{c_t}^{-1}\right]_{ij}\left(2\frac{\partial\left[G_{c_t}\right]_{ij}}{\partial\left[c_t\right]_k} - \frac{\partial\left[G_{c_t}\right]_{jk}}{\partial\left[c_t\right]_i}\right), \tag{7.29}$$

we get the desired geodesic ODE

$$\left[\ddot{c}_t\right]_j = -\sum_{i=1}^{d}\sum_{k=1}^{d}\Gamma_j^{ik}(c_t)\left[\dot{c}_t\right]_k\left[\dot{c}_t\right]_j. \tag{7.30}$$

∼ *Alternative expression of the geodesic ODE* ∽

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 [4]:

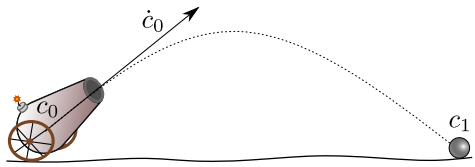
$$\ddot{c}_t = -\frac{1}{2}\mathbf{G}_{c_t}^{-1}\Big\{2(\mathbf{I}\otimes\dot{c}_t^\intercal)\partial_{c_t}\mathrm{vec}(\mathbf{G}_{c_t})\,\dot{c}_t - \partial_{c_t}\mathrm{vec}(\mathbf{G}_{c_t})^\intercal\,(\dot{c}_t\otimes\dot{c}_t)\Big\}.$$

Here $\mathrm{vec}(\cdot)$ 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, but it can be practical in numerical environments.

∼ *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 be 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*[1] 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

[1] Readers unfamiliar with this theorem may consult Appendix A for a brief recap.

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.

∼ Exercises ∽

Exercise 7.1: Euclidean metric.
Consider the Euclidean metric of $\mathbb{R}^d$, i.e. $\mathbf{G} = \mathbf{I}$.

1. *Derive the coefficients of the geodesics* ODE *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 $f : (0, \pi) \times [0, 2\pi) \to \mathbb{R}^3$

$$f(x_1, x_2) = \begin{pmatrix} \sin(x_1)\cos(x_2) \\ \sin(x_1)\sin(x_2) \\ \cos(x_1) \end{pmatrix}. \tag{7.31}$$

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

$$\mathbf{G}_{\mathbf{x}} = \begin{pmatrix} 1 & 0 \\ 0 & \sin^2(x_1) \end{pmatrix} \tag{7.32}$$

1. *Derive the coefficients of the geodesics* ODE *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

$$\mathbf{G}_{\mathbf{x}} = \left(1 + \|\mathbf{x}\|^2\right)\mathbf{I}, \qquad \mathbf{x} \in \mathbb{R}^2. \tag{7.33}$$

1. *Derive the coefficients of the geodesics* ODE *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* $\ddot{c}_0$*?*

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.

∼ *Do yourself a favor...* ∽

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 $\mathbf{G} = \mathbf{J}^\intercal\mathbf{J}$ requires a Jacobian matrix. And so forth. Computing such derivatives is often feasible, but it is tedious 'grunt work'.

If you do any form of numerical differential geometry, I strongly recommend that you use software to compute the required derivatives. Fortunately, tools for *automatic differentiation* [31] are plentiful and constantly evolving.

∼ *Connecting geodesics* ∽

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

summarized as

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.

Step 1, take 1: piecewise linear curves

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

$$t_i = \frac{i}{N}, \qquad \text{for } i = 0, \ldots, N. \tag{8.1}$$

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. 8.1).

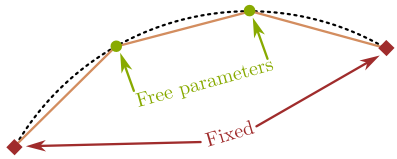


Figure 8.1: A continuous curve (dashed, black) is approximated by a piecewise linear curve (brown). The endpoints (red diamonds) are considered fixed, while the green circles are considered free parameters that can move during optimization.

Step 1, take 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,

$$c(t) = (1-t)c_0 + tc_1 + \tilde{c}(t), \tag{8.2}$$

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. 8.2. This decomposition neatly allows you to change the curve endpoints with little effort.

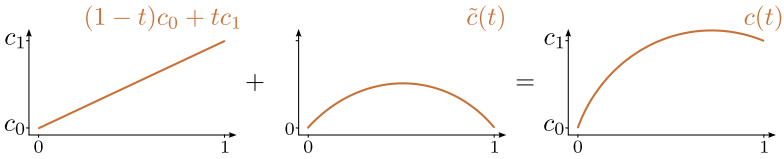


Figure 8.2: A curve $c(t)$ is split into a linear part, $(1-t)c_0+tc_1$, and a nonlinear remainder $\tilde{c}(t)$. This has the benefit that the same parametrization of $\tilde{c}(t)$ can be used independently of the geodesic endpoints.

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

$$\tilde{c}_i(t) = \sum_{k=1}^{K} w_{ik} t^k + w_{i0}, \tag{8.3}$$

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 $i = 1, \ldots, d$ and $k = 1, \ldots, K-1$.

Step 2, take 1: energy quadrature

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

$$\mathcal{E}[c] = \int_0^1 \|\dot{c}_t\|^2_{\mathbf{G}_{c_t}} \, \mathrm{d}t \approx \tilde{\mathcal{E}}[c]. \tag{8.4}$$

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

$$\tilde{\mathcal{E}}[c] = \sum_{s=0}^{S} \|\dot{c}_s\|^2_{\mathbf{G}_{c_s}} \Delta_s, \tag{8.5}$$

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 $\|\dot{c}_s\|^2_{\mathbf{G}_{c_s}} = \dot{c}_s^\intercal \mathbf{G}_{c_s} \dot{c}_s$. When working with pullback metrics $\mathbf{G} = \mathbf{J}^\intercal\mathbf{J}$ it is, instead, often more practical to recall that the curve energy can be written as

$$\mathcal{E} = \int_0^1 \left\| \frac{\mathsf{d}}{\mathsf{d}t} f(c_t) \right\|^2 \mathsf{d}t. \tag{8.6}$$

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

$$\tilde{\mathcal{E}}[c] = \sum_{s=1}^{S} \|f(c_s) - f(c_{s-1})\|^2. \tag{8.7}$$

Compared to working directly with the metric $\mathbf{G} = \mathbf{J}^\intercal\mathbf{J}$, 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 (8.7) 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 (8.6) as an expectation,

$$\mathcal{E} = \int_0^1 \left\| \frac{\mathsf{d}}{\mathsf{d}t} f(c_t) \right\|^2 \mathsf{d}t = \mathbb{E}_{p(t)} \left[\left\| \frac{\mathsf{d}}{\mathsf{d}t} f(c_t) \right\|^2 \right], \tag{8.8}$$

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

$$\tilde{\mathcal{E}}[c] = \frac{1}{S} \sum_{s=1}^{S} \left\| \frac{\mathsf{d}}{\mathsf{d}t} f(c_{t_s}) \right\|^2, \qquad t_s \sim \mathcal{U}(0,1). \tag{8.9}$$

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

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

$$w_{\text{opt}} = \operatorname*{argmin}_{\mathbf{w}} \tilde{\mathcal{E}}[c_{\mathbf{w}}]. \tag{8.10}$$

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.

∼ *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[1] and instead assume that they are available through some library implementation.

[1] Interested readers can consult standard textbooks [30].

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

$$[\ddot{c}_t]_j = h(t, c_t, \dot{c}_t) = -\sum_{i=1}^{d}\sum_{k=1}^{d} \Gamma_j^{ik}(c_t)\,[\dot{c}_t]_k\,[\dot{c}_t]_j\,, \tag{8.11}$$

where Γ_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 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

$$\dot{k}_t = \begin{pmatrix} \dot{c}_t \\ \ddot{c}_t \end{pmatrix} = \begin{pmatrix} \dot{c}_t \\ h(t, c_t, \dot{c}_t) \end{pmatrix} \quad \begin{matrix} \leftarrow \text{ part of } k_t \\ \leftarrow \text{ the geodesic ODE.} \end{matrix} \tag{8.12}$$

This gives rise to the following high-level algorithm.

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 $k_t = \texttt{odeint}(\dot{k},\, k_0,\, t)$.
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 $t_1, \ldots, t_n$ where the solution curve should be evaluated. Usually, these are equidistant points from 0 to 1.[2]

[2] `linspace(0, 1, 25)` in several programming environments.

∼ *Connecting geodesics, again* ∽

Modern ODE solvers support automatic differentiation, such that we can differentiate the solution with respect to the initial condition [9]. That is, we can evaluate $\partial c_t / \partial \dot{c}_0$.

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

$$\dot{c}_0^* = \operatorname*{argmin}_{\dot{c}_0} \|c_1 - c(1; \dot{c}_0)\|^2. \tag{8.13}$$

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

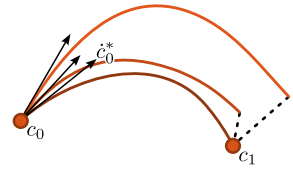


Figure 8.3: The initial velocity of the geodesic ODE is optimized such that the resulting geodesic hits the right end-point. The dashed lines indicate the optimization criteria.

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 [9]. 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

$$\mathbf{G}_{\mathbf{x}} = \left(1 + \|\mathbf{x}\|^2\right) \mathbf{I}, \qquad \mathbf{x} \in \mathbb{R}^2. \tag{8.14}$$

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: 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.

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

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

Here we have parametrized the shortest path over $t \in [0, 1]$, which is an arbitrary choice. We see that Δ 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.*

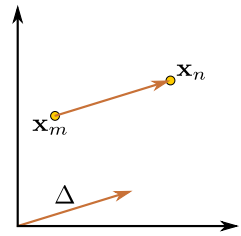


Figure 9.1: Subtraction in a Euclidean space is an operator that returns a vector representing the shortest path (straight line) connecting two points.

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 Δ 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* Δ. 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

$$- : \quad \mathcal{M} \times \mathcal{M} \to T \tag{9.2}$$

$$+ : \quad \mathcal{M} \times T \ \to \mathcal{M}. \tag{9.3}$$

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 $\text{dist}(\mathbf{x}_n, \mathbf{x}_m) = \|\mathbf{x}_n - \mathbf{x}_m\| = \|\Delta\|$.

∼ *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*[1] 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 Δ 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 Δ as the Riemannian counterpart to the difference vector attained by Euclidean subtraction.

[1] See Appendix A if you are not familiar with this result.

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

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

and note that Δ by definition belongs to the tangent space $T_{\mathbf{x}_m}$. If the geodesic is parametrized over the time interval $[0, 1]$ with

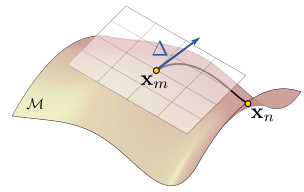


Figure 9.2: The logarithm map (subtraction) returns the initial velocity of the connecting geodesic.

constant speed, then we see that its length is

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

$$= \|\partial_t c_{mn}(0)\| \int_0^1 \mathsf{d}t \tag{9.6}$$

$$= \|\Delta\|. \tag{9.7}$$

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 Δ is a member of the tangent space $T_{\mathbf{x}_m}$, then the norm of Δ 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

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

$$\partial_t c(0) = \Delta. \tag{9.9}$$

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

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

∼ *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

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

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

$$\mathsf{dist}(\mathbf{y}_1, \mathbf{y}_2) = \arccos(\mathbf{y}_1^\intercal \mathbf{y}_2). \tag{9.12}$$

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

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 $f : \mathbb{R}^d \to \mathbb{R}^D$ and $\Omega \subset \mathbb{R}^d$. If Ω is compact then the manifold has a finite volume defined as

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

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 Ω rather than the high-dimensional $\mathcal{M}$. This is what we will set out to do.

∼ *Ordinary integration* ∽

Before extending our view of integration, let us briefly recap the basics of the *Riemann integral*.[1] Given a real function $h : \mathbb{R} \to \mathbb{R}$, 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 the first-axis into segments of size Δ_x and then

[1] 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.

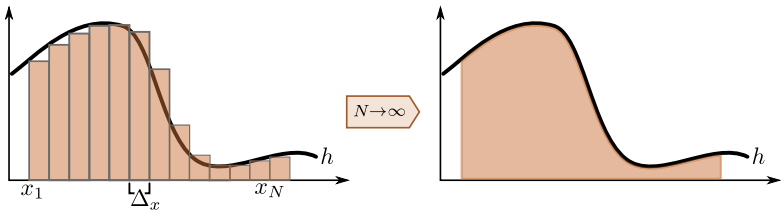


Figure 10.1: The Riemann sum approximates the area under the curve by summing the volume of small segments that fit the curve. Note how the segment width Δ_x depends on the first axis being the real line $\mathbb{R}$.

summing the areas of the corresponding boxes that fit under the curve,

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

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 $N \to \infty$,

$$\int_a^b h(x)\mathsf{d}x = \lim_{N\to\infty} \sum_{n=1}^{N} h(x_n)\Delta_x. \tag{10.3}$$

The notation $\mathsf{d}x$ should be thought of as the limiting value of Δ_x, *i. e.* it is the width of an infinitesimal segment.

∼ *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 $h : \mathcal{M} \to \mathbb{R}$, 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 $h : \mathcal{M} \to \mathbb{R}$ is the *integrand*, i.e. the function

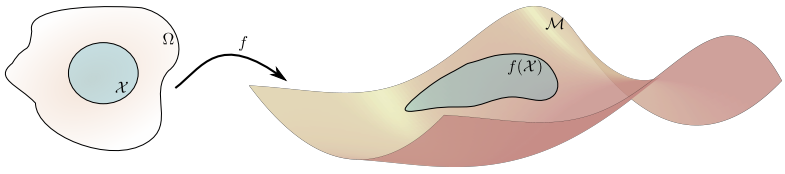


Figure 10.2: When integrating a function over a subset of the manifold, $f(\mathcal{X})$ it is generally more practical to express it as in integral over $\mathcal{X}$ instead.

which we seek to integrate. Then we can define the integral of interest as

$$\int_{f(\mathcal{X})} h\left(\mathbf{y}\right) \mathsf{d}\mathbf{y}. \tag{10.4}$$

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

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

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 Ω 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*

$$\mathbf{J}_\mathbf{x} = \mathbf{U}_\mathbf{x}\mathbf{S}_\mathbf{x}\mathbf{V}_\mathbf{x}^\intercal, \tag{10.6}$$

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.[2] Since orthogonal matrices (rotation matrices) do not influence the volume of the parallelogram spanned by the Jacobian, we observe that this must be the product of the diagonal of $\mathbf{S}_x$,

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

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

$$\mathbf{G}_\mathbf{x} = \mathbf{J}_\mathbf{x}^\intercal\mathbf{J}_\mathbf{x} = \mathbf{V}_\mathbf{x}\mathbf{S}_\mathbf{x}^\intercal\mathbf{S}_\mathbf{x}\mathbf{V}_\mathbf{x}^\intercal. \tag{10.8}$$

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

$$V^2(\mathbf{x}) = \det\left(\mathbf{G}_\mathbf{x}\right) \quad \Leftrightarrow \quad V(\mathbf{x}) = \sqrt{\det\left(\mathbf{G}_\mathbf{x}\right)}. \tag{10.9}$$

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

$$\int_{\mathcal{X}} h(f(\mathbf{x}))\sqrt{\det\left(\mathbf{G}_\mathbf{x}\right)}\mathsf{d}\mathbf{x}. \tag{10.10}$$

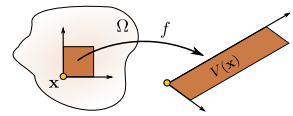


Figure 10.3: The volume of an infinitesimal patch in Ω changes when measured in ambient space.

[2] As the Jacobian is assumed to have full rank.

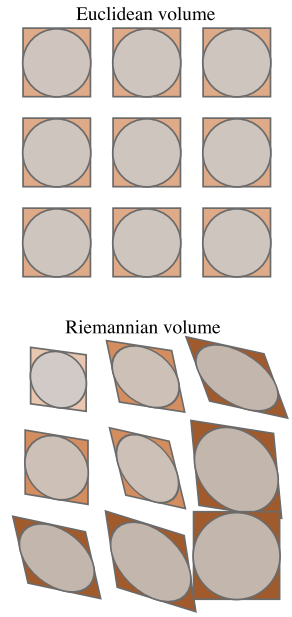


Figure 10.4: Integration is defined by partitioning space into small segments and the volume of each segment weighs its contribution to the total integral. In Euclidean space, this volume is constant, but for Riemannian models, we must take the 'size' of the local metric into account.

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

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

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.

∼ *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,

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

The probability of an event is then defined as usual

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

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

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

For most distributions on most manifolds, we cannot evaluate the normalization constant of the density in closed-form. This implies that we most often need to call upon numerical approximations such as *Monte Carlo* techniques when building probabilistic models over manifolds.

∼ *Expectations* ∽

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

$$\mathbb{E}\left[h(\mathbf{x})\right] = \int_{\Omega} h(\mathbf{x})p(\mathbf{x})\mathsf{d}\mathcal{M}(\mathbf{x}). \tag{10.15}$$

For finite data, this is then approximated as

$$\mathbb{E}\left[h(\mathbf{x})\right] \approx \frac{1}{N}\sum_{n=1}^{N} h(\mathbf{x}_n)\mathsf{d}\mathcal{M}(\mathbf{x}_n). \tag{10.16}$$

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 μ as the average squared distance to data,

$$\mathrm{Var}[\mathbf{x}] = \mathbb{E}\left[\mathsf{dist}^2(\mathbf{x}, \mu)\right]. \tag{10.17}$$

In Euclidean statistics, it is well-known that the mean is the unique minimizer of $\mathrm{Var}[\mathbf{x}]$. 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.

∼ 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 $f : [0, \pi] \times [0, 2\pi) \to \mathbb{R}^3$

$$f(x_1, x_2) = \begin{pmatrix} \sin(x_1)\cos(x_2) \\ \sin(x_1)\sin(x_2) \\ \cos(x_1) \end{pmatrix}. \tag{10.18}$$

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

$$\mathbf{G}_\mathbf{x} = \begin{pmatrix} 1 & 0 \\ 0 & \sin^2(x_1) \end{pmatrix}. \tag{10.19}$$

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 $\int_0^\pi \int_0^{2\pi} h(x_1, x_2) d\mathcal{M}(x_1, x_2)$.*

Exercise 10.2: Quadratic metric.

Consider a two-dimensional abstract manifold with the metric

$$\mathbf{G}_\mathbf{x} = \left(1 + \|\mathbf{x}\|^2\right) \mathbf{I}, \qquad \mathbf{x} \in \mathbb{R}^2. \tag{10.20}$$

1. *Derive the volume measure of the metric.*
2. *Integrate the function $h(x_1, x_2) = x_1 x_2$ with respect to the volume measure over the set $[-1, 1]^2$, i.e. compute $\int_{-1}^1 \int_{-1}^1 h(x_1, x_2) d\mathcal{M}(x_1, x_2)$.*

Exercise 10.3: Normalizing flows.

Consider a smooth embedding[3] $f : \mathbb{R}^d \to \mathbb{R}^D$, 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. $p(\mathbf{y}) > 0 \; \forall \mathbf{y}$.

[3] If you forgot about embeddings, see page 26.

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 $\mathbf{G}_\mathbf{x} = \mathbf{J}_\mathbf{x}^\intercal \mathbf{J}_\mathbf{x}$. Use eq. 10.10 to evaluate the probability $\Pr(\mathbf{x} \in \hat{\mathcal{X}}) = \int_{f(\hat{\mathcal{X}})} p(\mathbf{y}) \mathrm{d}\mathbf{y}$.*
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.*

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?*

∼ *Composite functions* ∽

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

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

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

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

where $f_1 : \mathbb{R}^d \to \mathbb{R}^{d'}$, $f_2 : \mathbb{R}^{d'} \to \mathbb{R}^D$, and $d \leq d' \leq D$.[1] The analysis we have carried out thus far tells us that then $\Omega \subseteq \mathbb{R}^d$ is naturally equipped with a Riemannian metric $\mathbf{G}_\mathbf{x} = \mathbf{J}_\mathbf{x}^\intercal \mathbf{J}_\mathbf{x}$. The same analysis, however, also tells us that $\Omega' = f_1(\Omega) \subseteq \mathbb{R}^{d'}$ also has

[1] In a machine learning context, the obvious example of such a function is a multilayer neural network.

a Riemannian metric,

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

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

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

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

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

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

where $\mathbf{J}_{\mathbf{x}}^{(1)} = {}^{\partial f_1}/_{\partial \mathbf{x}}$ is the Jacobian of f_1. Inserting the definition of the norm (5.14) then gives

$$\text{Length}(c) = \int_0^1 \sqrt{\left(\mathbf{J}_{c_t}^{(1)} \dot{c}_t\right)^\intercal \mathbf{G}_{c_t}^{(2)} \left(\mathbf{J}_{c_t}^{(1)} \dot{c}_t\right)} \mathsf{d}t \tag{11.7}$$

$$= \int_0^1 \sqrt{\dot{c}_t^\intercal \left(\mathbf{J}_{c_t}^{(1)\intercal} \mathbf{G}_{c_t}^{(2)} \mathbf{J}_{c_t}^{(1)}\right) \dot{c}_t} \mathsf{d}t. \tag{11.8}$$

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

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 Ω is

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

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

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 $\mathbf{G}_{\mathbf{x}}$.

∼ *Ambient Riemannian 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 Ω.

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

$$f : \Omega \to \mathcal{M}^{(D)}, \qquad \Omega \subseteq \mathbb{R}^d \tag{11.11}$$

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 Ω as

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

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

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

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* $\mathbf{G}_1 = \mathbf{J}_{f_1}^\intercal \mathbf{J}_{f_1}$, $\mathbf{G}_2 = \mathbf{J}_{f_2}^\intercal \mathbf{J}_{f_2}$, *and* $\mathbf{G}_3 = \mathbf{J}_{f_3}^\intercal \mathbf{J}_{f_3}$, *and write* $\mathbf{G}_{f_3}$ *in terms of* $\mathbf{G}_{f_2}$.

Chapter 12

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 said data is practically always finite in amount and subject to observation noise, we should start off with the acknowledgment that the estimated manifold is just that: *an estimate*. But how does this imperfection reveal itself in the geometry of the manifold?

∼ *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* [24], is to find a function f that minimizes the squared distance between observations and projections thereof onto the manifold:

$$f^* = \operatorname*{argmin}_f \sum_{n=1}^{N} \|f(\mathbf{x}_n) - \mathbf{y}_n\|^2 , \tag{12.1}$$

where $\mathbf{x}_n$ is chosen such that $f(\mathbf{x}_n)$ is the projection[1] of $\mathbf{y}_n$ onto the manifold spanned by f:

$$\mathbf{x}_n = \operatorname{proj}_{\mathcal{M}}(\mathbf{y}_n) = \operatorname*{argmin}_{\mathbf{x}} \|f(\mathbf{x}) - \mathbf{y}_n\|^2 . \tag{12.2}$$

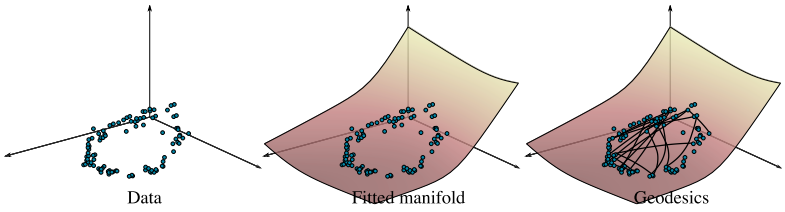
[1] Technically, an autoencoder does not optimize the representations $\mathbf{x}_{1:N}$, but rather optimizes a neural network that predicts the representation $\mathbf{x}_n$ from an observation $\mathbf{y}_n$. This is irrelevant from our perspective as it is merely an approximation.

Combining Eqs. 12.1 and 12.2 into one optimization problem gives

$$f^*, \mathbf{x}_1^*, \ldots, \mathbf{x}_n^* = \underset{f, \mathbf{x}_{1:N}}{\operatorname{argmin}} \sum_{n=1}^{N} \|f(\mathbf{x}_n) - \mathbf{y}_n\|^2 . \tag{12.3}$$

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 12.1 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.[2] 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 fails to capture the structure hidden in the data.



[2] 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.

Figure 12.1: 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.

This example highlights an unfortunate situation: we want our models to be smooth, as experience suggests that such models are more stable, 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 result...

∼ *Gaussian noisy manifolds* ∽

Perhaps the result of Fig. 12.1 is unsurprising. After all, we are trying to estimate a function $f : \mathbb{R}^2 \to \mathbb{R}^3$ 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.[3] We will consider the additive Gaussian noise model where

$$\mathbf{y} = f(\mathbf{x}) + \epsilon\, \sigma(\mathbf{x}), \tag{12.4}$$

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

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathbf{y}|f(\mathbf{x}), \operatorname{diag}\left(\sigma^2(\mathbf{x})\right)\right), \tag{12.5}$$

where $\operatorname{diag} : \mathbb{R}^D \to \mathbb{R}^{D\times D}$ create a diagonal matrix from a vector, and $\mathcal{N}$ denotes the density of the Gaussian distribution, *i. e.*

$$\mathcal{N}\left(\mathbf{y}|\mu, \Sigma\right) = \frac{1}{\sqrt{(2\pi)^D \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mu)^\intercal \Sigma^{-1}(\mathbf{y}-\mu)\right).$$

Examples of such models include the *generative topographic map* [6] and the *variational autoencoder* [16, 22].

[3] For now we focus on the case of *noisy data*. The case of truly uncertain models will handled in a chapter that remains to be written.

∼ *Randomly projected manifolds* ∽

Intuitively, we see that the model in Eq. 12.4 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 Fig. 12.1, 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. 12.4 in matrix form,

$$\mathbf{y} = f(\mathbf{x}) + \epsilon\, \sigma(\mathbf{x}) \tag{12.6}$$

$$= \begin{pmatrix}\mathbf{I} & \epsilon\end{pmatrix}\begin{pmatrix} f(\mathbf{x}) \\ \sigma(\mathbf{x})\end{pmatrix} = \mathbf{P}h(\mathbf{x}). \tag{12.7}$$

Here $\mathbf{P} = (\mathbf{I}, \epsilon) \in \mathbb{R}^{D\times 2D}$ is a *random matrix*[4] which we can choose

[4] That's just a fancy way of saying a matrix with random entries.

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. 12.4.

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.

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 'bend circle' in a high-dimensional space. To these points, we add Gaussian noise and fit a randomly projected manifold of the type in Eq. 12.4.

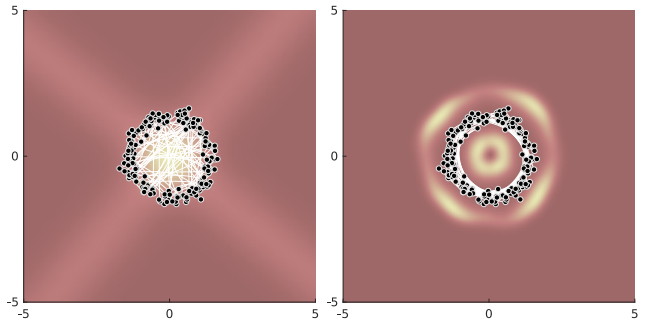


Figure 12.2: Data distributed near a circle in a high-dimensional space is represented by a low-dimensional manifold. *Left:* geodesics under the metric associated with the mean of the estimated manifold. *Right:* geodesics under a metric that takes the uncertainty of the manifold into account. In both plots, the background color represents the volume measure associated with the metric.

This procedure gives a low-dimensional representation, which we can give a Riemannian metric. The panels of Fig. 12.2 show this representation equipped with different metrics. The left panel disregards the noise and considers the metric associated with f alone, *i.e.* $\mathbf{G}_\mathbf{x} = \mathbf{J}_f(\mathbf{x})^\intercal \mathbf{J}_f(\mathbf{x})$. The white curves correspond to geodesics between randomly chosen points; it is quite evident that here 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. 12.1. The right panel of Fig. 12.2 on the other hand considers the Riemannian metric associated with $h(\mathbf{x}) = (f(\mathbf{x}); \sigma(\mathbf{x}))$, which take the form

$$\mathbf{G}_\mathbf{x} = \mathbf{J}_f(\mathbf{x})^\intercal \mathbf{J}_f(\mathbf{x}) + \mathbf{J}_\sigma(\mathbf{x})^\intercal \mathbf{J}_\sigma(\mathbf{x}). \tag{12.8}$$

Derived in exercise 12.1.

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. 12.2 is the choice of metric, and the difference between these is the term $\mathbf{J}_\sigma(\mathbf{x})^\intercal \mathbf{J}_\sigma(\mathbf{x})$ in Eq. 12.8. Clearly, this makes an important improvement to the metric. In this particular example, σ 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 σ takes large values when σ changes from small to large values. Curves that pass through such regions then become rather long, and the shortest curves (geodesics) naturally try to avoid these regions. In the right panel of Fig. 12.2 the expensive regions appear as the yellow 'walls' that surround the data. On randomly projected manifolds, uncertainty, thus, pushes geodesics toward the observed data.

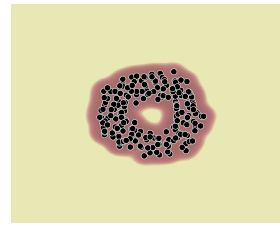


Figure 12.3: The summed per-dimension variance plotted in the latent representation space, i.e. $\sum_{d=1}^{D} \sigma^2(\mathbf{x})$, as background. Dots correspond to latent data representations.

∼ *Statistical submanifolds* ∽

The idea of *projected manifolds* rely crucially on the the decomposition in Eq. 12.6, which we here repeat

$$\mathbf{y} = f(\mathbf{x}) + \epsilon\, \sigma(\mathbf{x}). \tag{12.9}$$

(valid for Gaussian distributions)

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.[5] This prevents us from applying the idea of randomly projected manifolds.

[5] e.g. the *categorical* or *multinomial* distribution cannot be as simply decomposed.

So far, our perspective has been that we have a map f from the latent representations into the space of *observations*. Our studied

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 think of a map between latent representations $\mathbf{x}$ and conditional distributions $p(\mathbf{y}|\mathbf{x})$. For example, the Gaussian model (12.5) can be viewed as

$$f : \mathbf{x} \mapsto \mathcal{N}\left(\mathbf{y}|f(\mathbf{x}), \text{diag}\left(\sigma^2(\mathbf{x})\right)\right). \tag{12.10}$$

If we had access to a distance measure between distributions, then we could pull this measure into the latent representation space.

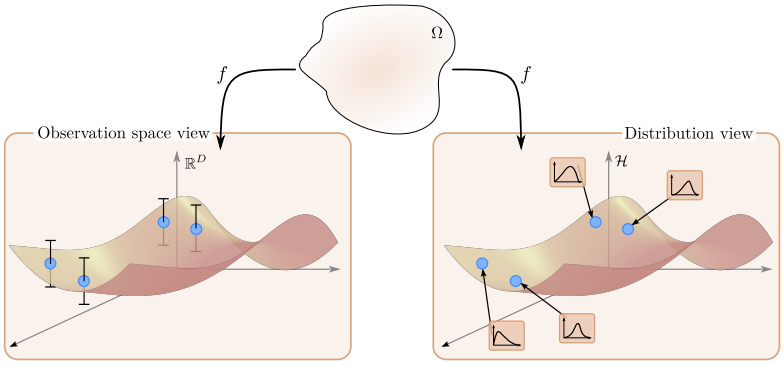


Figure 12.4: Two different perspectives leading to different latent geometries. In both views, we have latent representations in Ω that are mapped through a function f. The *observation space view* (left) maps latent representations to the space of observations. To account for noise, this mapping must be stochastic forcing us to consider projected manifolds. In the *distribution view* (right), 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.

The space of distributions

The construction we are working towards requires first a space of distributions, and second a distance measure over this space. This may sound terribly abstract, but it's actually rather straightforward.

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

For example, if $p(\mathbf{y}|\mathbf{x})$ is the univariate Gaussian $\mathcal{N}(\mu, \sigma^2)$ then the parameters are $\theta = (\mu, \sigma)$. The space of such distributions is then $\mathcal{H} = \mathbb{R} \times \mathbb{R}_+$ since μ is real-valued and σ is positive. 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) $\Sigma \in \mathsf{Sym}_+^{D\times D}$, we have that the space of such distributions is $\mathcal{H} =$

$\mathbb{R}^D \times \mathsf{Sym}_+^{D\times D}$. Throughout the text, $\mathcal{H}$ will denote the space of considered distributions.

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

Distances between distributions (a naive approach)

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 distributions. Once we have such a measure, we can pull this into our latent representation space to construct a latent distance measure.

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

$$\text{dist}(p_1, p_2) = \sqrt{(\mu_1 - \mu_2)^2 + (\sigma_1 - \sigma_2)^2}. \tag{12.11}$$

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 which did not depend on how we had chosen to parametrize the distributions. A *reparametrization invariant* distance measure, that is.

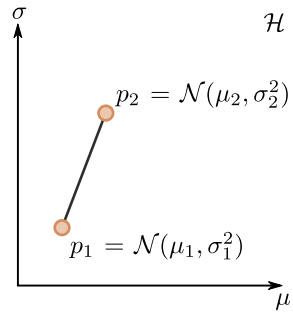


Figure 12.5: The Euclidean distance between two univariate Gaussian parametrized as $\theta = (\mu, \sigma)$. *There is much arbitrariness in this choice of distance measure.*

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

$$\text{KL}(p\|q) = \int \log\left(\frac{p(\mathbf{y})}{q(\mathbf{y})}\right) p(\mathbf{y}) \mathsf{d}\mathbf{y} = \mathbb{E}_{p(\mathbf{y})}\left[\log\left(\frac{p(\mathbf{y})}{q(\mathbf{y})}\right)\right] \tag{12.12}$$

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

We can easily observe that the KL–divergence is positive for distinct distributions using *Jensen's inequality*[6]

$$\mathrm{KL}(p\|q) = -\int \log\left(\frac{q(\mathbf{y})}{p(\mathbf{y})}\right) p(\mathbf{y})\mathsf{d}\mathbf{y} \tag{12.13}$$

$$\geq -\log \underbrace{\int \frac{q(\mathbf{y})}{p(\mathbf{y})} p(\mathbf{y})\mathsf{d}\mathbf{y}}_{=1} = 0. \tag{12.14}$$

[6] For a concave function ϕ, $\phi(\mathbb{E}[\mathbf{y}]) \geq \mathbb{E}[\phi(\mathbf{y})]$.

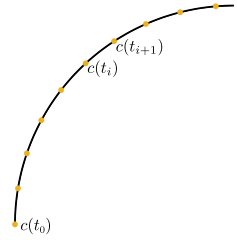
By Jensen's inequality, we also see that equality only holds when $p(\mathbf{y}) = q(\mathbf{y})$ for all $\mathbf{y}$.

The KL–divergence is, however, not a distance measure as it is generally *not* symmetric, i.e. $\mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p)$. This prevents us from directly pulling the KL–divergence into the latent representation space to form a metric.

Curve lengths and energies, revisited

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

$$\mathrm{Length}(c) = \lim_{N\to\infty} \sum_{i=0}^{N} \mathsf{dist}\left(f(c(t_i)),\ f(c(t_{i+1}))\right). \tag{12.15}$$



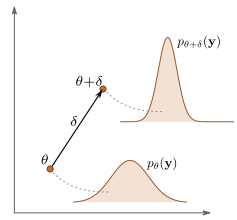
Similarly, for the *energy* of a curve

$$\mathcal{E}(c) = \lim_{N\to\infty} \sum_{i=0}^{N} \mathsf{dist}^2\left(f(c(t_i)),\ f(c(t_{i+1}))\right). \tag{12.16}$$

We, thus, see that we only need our choice of similarity measure between distributions to be a distance function *locally*.

The Fisher–Rao metric

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



$$\mathrm{KL}(p\|q) = -\frac{1}{2}\delta^\intercal \underbrace{\int p(\mathbf{y})\nabla_\theta^2 \log p(\mathbf{y})\mathbf{dy}}_{\mathbf{F}} \delta + \mathcal{O}(\|\delta\|^3), \tag{12.17}$$

We detail this computation in exercise 12.3.

where ∇_θ^2 denotes the Hessian. Infinitessimally, we, thus, see that the KL–divergence is a simple quadratic equation, $\mathrm{KL}(p\|q) = \delta^\intercal \mathbf{F}\delta$.

The matrix $\mathbf{F}$ can be rewritten in the more elegant form as

$$\mathbf{F} = -\frac{1}{2}\int p(\mathbf{y})\nabla^2 \log p(\mathbf{y})\mathbf{dy} \tag{12.18}$$

$$= \frac{1}{2}\int p(\mathbf{y})\nabla_\theta \log p(\mathbf{y})\nabla_\theta \log p(\mathbf{y})^\intercal \mathbf{dy} \tag{12.19}$$

Derived in exercise 12.3.

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.*

Pulling back Fisher–Rao

Now we are finally equipped to provide a distribution-defined metric over learned latent representations. Recall that we consider a manifold spanned by f within the space of distributions, i.e.

$$f : \Omega \to \mathcal{H}, \tag{12.20}$$

$$f : \mathbf{x} \mapsto p_\theta(\mathbf{y}|\mathbf{x}). \tag{12.21}$$

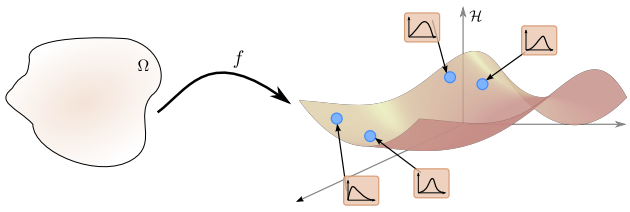


Figure 12.6: The model maps latent representations $\mathbf{x}$ to conditional distributions $p_\theta(\mathbf{y}|\mathbf{x})$. The space of such distributions $\mathcal{H}$ (the set of possible parameters θ) has a Riemannian (Fisher–Rao) metric $\mathbf{F}$. We can bring this metric into the latent representation space to have a reparametrization invariant latent metric.

We can, thus, define the energy of a curve $c : [0, 1] \to \Omega$ in the latent representation as

$$\mathcal{E}(c) = \lim_{N\to\infty} \sum_{i=0}^{N} \mathrm{KL}\big(f(c(t_i)) \parallel f(c(t_{i+1}))\big) \tag{12.22}$$

$$= \int_0^1 \left\| \frac{\mathsf{d}}{\mathsf{d}t} f(c_t) \right\|^2_{\mathbf{F}_{c_t}} \mathsf{d}t \tag{12.23}$$

$$= \int_0^1 \dot{c}_t^\intercal \mathbf{J}_{c_t}^\intercal \mathbf{F}_{c_t} \mathbf{J}_{c_t} \dot{c}_t \mathsf{d}t. \tag{12.24}$$

Here $\|\Delta\|^2_{\mathbf{F}_{c_t}} = \Delta^\intercal \mathbf{F}_{c_t} \Delta$, where $\mathbf{F}_{c_t}$ is the Fisher–Rao metric at $c_t = c(t)$.

We, thus, see that placing a Fisher-Rao metric on the space of distributions yields a latent Riemannian metric on the form

$$\mathbf{G}_{c_t} = \mathbf{J}_{c_t}^\intercal \mathbf{F}_{c_t} \mathbf{J}_{c_t}. \tag{12.25}$$

Unlike the randomly projected manifolds (12.8), this metric applies to practically all models as it merely relies on the availability of a KL–divergence.

Computational remarks

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 θ. 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,

$$\mathcal{E}(c) \approx \sum_{i=0}^{N} \mathrm{KL}\big(f(c(t_i)) \parallel f(c(t_{i+1}))\big), \tag{12.26}$$

which, for most distributions, can be done efficiently.

In cases where the metric is required, it often suffices to evaluate products of vectors with the metric, i.e. $\mathbf{G}\mathbf{v}$ for some vector $\mathbf{v}$. Noting that the Fisher–Rao metric is the Hessian of the KL–divergence, allows us to evaluate products using *automatic differentiation*. Letting $\mathtt{hvp}_g(\mathbf{v}) = \nabla^2 g \mathbf{v}$ denote the product between the Hessian of some function g and the vector $\mathbf{v}$, we can write

$$\mathbf{u}^\intercal \mathbf{G} \mathbf{v} = \mathbf{u}^\intercal \mathbf{J}^\intercal \mathbf{F} \mathbf{J} \mathbf{v} = (\mathbf{J}\mathbf{u})^\intercal \mathtt{hvp}_{\mathrm{KL}}(\mathbf{J}\mathbf{v}). \tag{12.27}$$

Similarly, letting $\texttt{jvp}_g(\mathbf{v})$ denote multiplication with the Jacobian of g, we get

$$= \texttt{jvp}_f(\mathbf{u})^\intercal \texttt{hvp}_{\mathrm{KL}} \texttt{jvp}_f(\mathbf{v}). \tag{12.28}$$

This is relevant as systems for automatic differentiation generally provide efficient algorithms for `jvp` and `hvp`.

∼ *Exercises* ∽

Exercise 12.1: Metric of randomly projected manifold.
Consider the manifold spanned by the mapping $f : \mathbb{R}^d \to \mathbb{R}^{2D}$ defined as

$$h(\mathbf{x}) = \begin{pmatrix} f(\mathbf{x}) \\ \sigma(\mathbf{x}). \end{pmatrix} \tag{12.29}$$

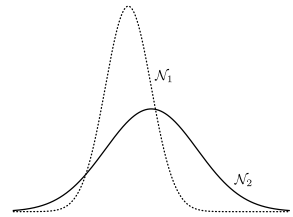
1. *Derive the pullback metric of this manifold.*

Exercise 12.2: Choice of parametrization.
Consider two normal distributions

$$\mathcal{N}_1 = \mathcal{N}(\mu_1, \sigma_1^2) \qquad \text{and} \qquad \mathcal{N}_2 = \mathcal{N}(\mu_2, \sigma_2^2), \tag{12.30}$$

where

$$\mu_1 = 0, \quad \mu_2 = 1, \quad \sigma_1 = 1, \quad \sigma_2 = 2. \tag{12.31}$$



The '*natural parametrization*' of the Gaussian.

1. *Compute the Euclidean distance between these distributions in the following parametrizations*

$$\theta = (\mu, \sigma) \tag{12.32}$$
$$\theta = (\mu, \sigma^2) \tag{12.33}$$
$$\theta = (\mu, \sigma^{-2}) \tag{12.34}$$
$$\theta = \left(\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2} \right). \tag{12.35}$$

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 pullback metric corresponds to that of the randomly projected manifold* (12.8).

Exercise 12.3: Deriving Fisher–Rao.
Consider the distribution $p_\theta(\mathbf{y})$ parametrized by θ, and let $p_{\theta+\delta}(\mathbf{y})$ be an infinitessimal displacement with respect to θ. The KL–divergence between these is

$$\begin{aligned} \mathrm{KL}(p_\theta \| p_{\theta+\delta}) = &\int \log\left(p_\theta(\mathbf{y})\right) p_\theta(\mathbf{y}) \mathsf{d}y \\ &- \int \log\left(p_{\theta+\delta}(\mathbf{y})\right) p_\theta(\mathbf{y}) \mathsf{d}y. \end{aligned} \tag{12.36}$$

1. *Make a second-order Taylor expansion of* $\log p_\theta(\mathbf{y})$ *with respect to* θ.

2. Use the Taylor expansion to approximate $p_{\theta+\delta}$ to derive Eq. 12.17.

3. Show that the Hessian with respect to θ of $\log p_\theta(\mathbf{y})$ is

$$\begin{aligned}\nabla^2_\theta \log p_\theta(\mathbf{y}) = &\frac{1}{\log p_\theta(\mathbf{y})} \nabla^2_\theta p_\theta(\mathbf{y}) \\ &- \nabla_\theta \log p_\theta(\mathbf{y}) \, \nabla_\theta \log p_\theta(\mathbf{y})^\intercal.\end{aligned} \tag{12.37}$$

4. *Finally, show that the Fisher–Rao metric is*

$$\mathbf{G} = \frac{1}{2} \int p_\theta(\mathbf{y}) \nabla_\theta \log p_\theta(\mathbf{y}) \nabla_\theta \log p_\theta(\mathbf{y})^\intercal \mathtt{d}\mathbf{y}. \tag{12.38}$$

Exercise 12.4: Deriving the Gaussian KL–divergence.
Consider two Gaussian distributions over $\mathbf{y} \in \mathbb{R}^D$

$$p(\mathbf{y}) = \mathcal{N}(\mu_p, \Sigma_p) \qquad \text{and} \qquad q(\mathbf{y}) = \mathcal{N}(\mu_q, \sigma_q). \tag{12.39}$$

1. *Assume that* $\Sigma_p = \Sigma_q = \mathbf{I}$ *and derive an expression for the KL–divergence* $\mathrm{KL}(p\|q)$.

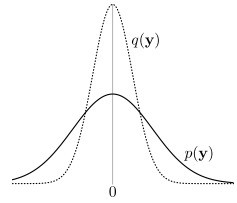
2. *Assume that* $\Sigma_p = \Sigma_q$ *and derive an expression for the KL–divergence* $\mathrm{KL}(p\|q)$.

3. *Assume that* $\Sigma_p \neq \Sigma_q$ *and derive an expression for the KL–divergence* $\mathrm{KL}(p\|q)$.

Exercise 12.5: Asymmetry of the KL–divergence.
The KL–divergence is inherently an asymmetric measure, i.e. $\mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p)$. While this property can be annoying when we seek a distance measure, it is fundamentally a sound property. Consider two Gaussian distributions over $\mathbf{y} \in \mathbb{R}$

$$p(\mathbf{y}) = \mathcal{N}(0, \sigma_p^2) \qquad \text{and} \qquad q(\mathbf{y}) = \mathcal{N}(0, \sigma_q^2), \tag{12.40}$$

where $\sigma_p \gg \sigma_q$.



1. *Pick values of* σ_p *and* σ_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 13

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.

∼ *The books is not done yet* ∽

First, I should mention that the book's incompleteness is partly due to it being unfinished. Several planned chapters have yet to be finished.

A fundamental chapter that remains unwritten is an introduction to *stochastic manifolds*. These occur when we let the function spanning our manifold be stochastic. This contrasts the *noisy manifolds* of chapter 12, where the *output* of the function is considered stochastic. A simple example of a stochastic manifold is a neural network, where we have a posterior distribution over the network weights.

A wonderful aspect of stochastic manifolds is that they provide a mechanism for sampling actual manifolds. This ensures that individual samples are smooth manifolds, unlike noisy manifolds where the sampled object is generally not continuous. Furthermore, the metric also becomes stochastic. The mathematical treatment of stochastic manifolds, therefore, is a bit more involved, and for now refer the reader to the original paper of Tosi et al. [29] who consider the expectations of the pullback metric or to the more elaborate

work of Pouplin et al. [20] who show that stochastic Riemannian manifolds are not Riemannian in expectation.

Another chapter that should be included in the book concerns general random variables. We have consistently considered *latent variables* and investigated the geometry of the latent variable space. The analysis is, however, applicable to any random variable. For example, we may want to endow the space of weights in a neural network with a Riemannian metric using the same tools as we have considered for latent variables.

Finally, the text has emphasized the construction of geometries but has placed less focus on *what we should use the geometries for*. Since we have a well-defined *measure* on the latent geometries, we can easily define probability distributions. From these, many established data analysis pipelines can be extended to the geometric setting. The densities defined with respect to the geometries, however, tend to lack closed-form expressions, which complicate analysis. But one should never shy away from a good challenge...

∼ *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 two most glaring heldouts are *topology* and *curvature*.

Riemannian geometry is concerned with making quantitative statements about curved objects (manifolds). Topology is concerned with the object itself, e.g. if are there holes in our manifold. One can reasonably argue, that we cannot give a proper definition of what a manifold is without first understanding topology (i.e. should we be making measurements on an object that we do not understand?).

In the context of generative models, we rarely have much topological information a priori,[1] 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* [7] seeks to answer this question, though the associated tools and algorithms are still in their infancy.

[1] e.g. when do you know the Betti numbers associated with a dataset?

The other heldout topic, *curvature*, in contrast, plays a quite central role in machine learning. In essence, the many different mea-

sures 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 more simple alternatives.[2] This could suggest that geometric notions of curvature are overly complex to be practical or that there is a missed opportunity for regularization research.

[2] e.g. L_2 weight regularization, *dropout*, etc.

∼ *Historical developments* ∽

The summary here is highly subjective and should mostly be taken as an indication of my interests rather than a complete exposition.

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* [12], 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 everpresent in the analysis of *shape data*. This started from the ideas of D'Arcy Thompson [10] 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)

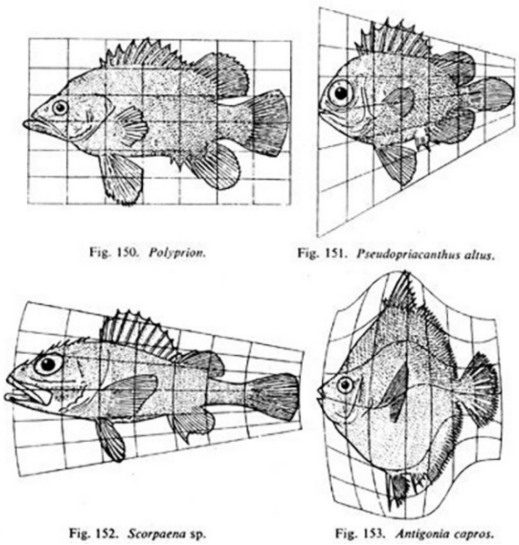


Figure 13.1: D'Arcy Thompson [10] pioneered the idea of studying evolution through *shape*. Since it's difficult to parametrize the actual shape of an animal (here *fish*), he opted to parametrize the *transformations* between shapes (background grid). One can then think of these transformations as the observed data and study these statistically. The figure is taken from the original book of D'Arcy Thompson.

The idea of statistically analyzing transformations naturally leads to differential geometry. In particular, the space of *diffeomorphisms*[3] 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 [19].

[3] i.e. invertible smooth transformations with smooth inverses

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. [29] who studied expectations of Riemannian metrics in models where the map from latent space to observation space is a *Gaussian process* [21]. Arvanitidis et al. [4] 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* [32], *biology* [11], *optimization* [1] and *robotics* [5, 25, 8].

The original Gaussian process model [29] has undergone deeper theoretical study. Adler and Taylor [2] provide a thorough investigation of similar models for the case of a *prior* model.[4] Hauberg [14] discuss links between uncertainty and topology through studies of Gaussian process extrapolations. Finally, Pouplin et al. [20] 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.

[4] i.e. the model before observing any data

Chapter A

Ordinary differential equations

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.

∼ *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 $c : [0,1] \to \mathbb{R}$ is a curve, and $\dot{c} : [0,1] \to \mathbb{R}$ is the curve derivative, i.e. $\dot{c}_t = \mathrm{d}c/\mathrm{d}t$. The function $f : [0,1] \times \mathbb{R} \to \mathbb{R}$, thus, evaluates the derivative of the curve c assuming we know the actual value of the curve.

The choice of the interval $[0,1]$ is arbitrary, and you should feel free to change to another interval if that's easier.

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 $f : [0,1] \times \mathbb{R}^D \to \mathbb{R}^D$, i.e. we have

$$f(t, c_t) = \begin{pmatrix} f_1(t, [c_t]_1) \\ \vdots \\ f_D(t, [c_t]_D) \end{pmatrix}. \tag{A.2}$$

This is often known as a *system of* ODE*s*.

∼ *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*

$$c(0) = c_0, \tag{A.3}$$

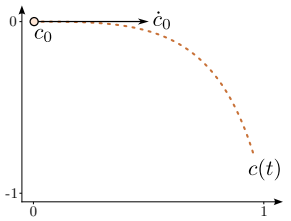
i.e. you should assume that we know the true value of the curve c at time $t = 0$. From this knowledge, we can evaluate the ODE to provide us with the derivative of the unknown curve c at $t = 0$,

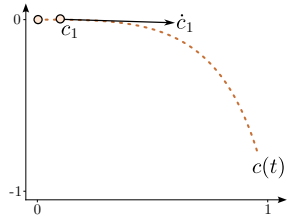
$$\left.\frac{\mathrm{d}c}{\mathrm{d}t}\right|_{t=0} = \dot{c}_0 = f(0, c_0). \tag{A.4}$$

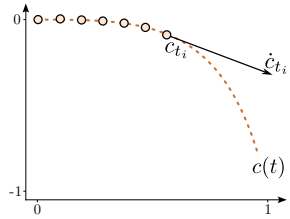
If we make a Taylor expansion of the unknown curve around c_0, we get

$$c(t) = c_0 + \dot{c}_0 t + \mathcal{O}(t^2). \tag{A.5}$$

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 $\{t_1 = 0, t_2, \ldots, t_N = 1\}$, we use the ODE to form a local Taylor expansion from which we extrapolate the solution curve and proceed to the next time-step:







Euler's method

Input: c_0, f.

For $i \in 0, 1, \ldots, N$:

$\dot{c}_{t_i} = f(\tau, c_{t_i})$

$c_{t_{i+1}} = c_{t_i} + \dot{c}_{t_i}(t_{i+1} - t_i)$

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 [30], which are often easily available in most programming environments.

Euler's method is practically identical to *gradient descent*, which is everpresent 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 then solve with Euler's method.

∼ *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[1] in c_t, then there exists an $\epsilon > 0$ such that the ODE has a unique solution for $t \in [t_0 - \epsilon, t_0 + \epsilon]$.

[1] i.e. continuous and with bounded derivatives

Index

euler's method, 110
fisher–rao metric, 99
fréchet mean, 85
gaussian process, 108
jensen's inequality, 98
kl, 97
kullback–leibler, 97
mercator's projection, 21
monte carlo, 84
peter's projection, 21
picard-lindelöf theorem, 63, 111
riemann integral, 81, 82
riemann sum, 81
riemannian geometry, 31
riemannian manifold, 47
riemannian metric, 47
runge-kutta, 110
theorema egregium, 19
topological data analysis, 106
abstract manifolds, 48
ambient space, 23
angle, 43
antipodal, 56
autoencoders, 91
autoencoder, 52
automatic differentiation, 67, 71, 100
categorical, 97
curvature, 106
curve length, 34
decoder, 52
deep generative model, 52
diffeomorphisms, 108
direct energy minimization, 67
divergence, 97
embedded, 26
encoder, 52
energy, 59, 98
exponential map, 78
first-order ODE, 109
generative topographic map, 93
geodesic distance, 34
geodesics, 55
geodesic, 34
gradient descent, 110
identifiability issue, 36
identifiability problem, 24
immersed manifold, 27
initial value problems, 63
initial value, 110
injective, 27
integrand, 82
integration by substitution, 83
integration, 81
interpolant, 55
isometry, 19
kernel density, 52
kernel ridge regression, 28
latent variable models, 12
left inverse, 27
locally invertible, 27
logarithm map, 77
loss function, 23
manifold hypothesis, 36
manifold, 26
mean, 85
measure, 84, 106
metric, 45
natural parametrization, 102
noisy manifolds, 94, 105
ordinary differential equations (ODEs), 60
pattern theory, 107
projected manifolds, 95
projection, 52
pseudo-riemannian, 54
random matrix, 93
random projection, 93
reconstruction error, 16
regularization, 92, 107
reparametrization issue, 24
reparametrized, 32
semi-riemannian, 54
shape statistics, 108
shooting geodesics, 64
singular value decomposition, 83
statistical divergences, 97
stochastic manifolds, 105
submanifold, 89
system of ODEs, 109
tangent space, 26, 42
topology, 106
two-point boundary value problem, 63
uniform density, 84
variance, 85
variational autoencoder (vae), 25
variational autoencoder, 93
velocity, 41

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] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. A locally adaptive normal distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, jun 2016.

[4] 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.

[5] Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Gerhard Neumann, and Leonel Rozo. Learning riemannian manifolds for geodesic motion skills. In *Robotics: Science and Systems (RSS)*, 2021.

[6] Christopher M Bishop, Markus Svensén, and Christopher KI Williams. Gtm: The generative topographic mapping. *Neural computation*, 10(1):215–234, 1998.

[7] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[8] 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.

[9] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[10] D'Arcy Thompson. *On Growth and Form*. Cambridge University Press, 1917.

[11] Nicki Skafte Detlefsen, Søren Hauberg, and Wouter Boomsma. Learning meaningful representations of protein sequences. *Nature communications*, 13(1):1914, 2022.

[12] Ulf Grenander and Michael I Miller. *Pattern theory: from representation to inference*. OUP Oxford, 2006.

[13] Peter E Hart, David G Stork, and Richard O Duda. *Pattern classification*. Wiley, 2000.

[14] Søren Hauberg. Only bayes should learn a manifold, 2018.

[15] 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.

[16] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[17] Tjalling C Koopmans and Olav Reiersøl. The identification of structural characteristics. *The Annals of Mathematical Statistics*, 21(2):165–181, 1950.

[18] Guy Lebanon. Learning riemannian metrics. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 362–369, 2002.

[19] Xavier Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25:127–154, 2006.

[20] Alison Pouplin, David Eklund, Carl Henrik Ek, and Søren Hauberg. Identifying latent distances with finslerian geometry. *Transactions on Machine Learning Research (TMLR)*, 2023.

[21] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[22] 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.

[23] 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.

[24] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[25] 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.

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

[27] 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.

[28] Jakub M. Tomczak. *Deep Generative Modeling*. Springer, 2022.

[29] Alessandra Tosi, Søren Hauberg, Alfredo Vellido, and Neil D. Lawrence. Metrics for probabilistic geometries. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014.

[30] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations*. Springer, 2009.

[31] Robert Edwin Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.

[32] Tao Yang, Georgios Arvanitidis, Dongmei Fu, Xiaogang Li, and Søren Hauberg. Geodesic clustering in deep generative models. *arXiv preprint arXiv:1809.04747*, 2018.