1 Introduction

Optimization is a staple of mathematical modeling. In this rich framework, we consider a set S called the *search space*—it contains all possible answers to our problem, good and bad—and a *cost function* $f\colon S \to \mathbb{R}$ which associates a cost $f(x)$ to each element x of S. The goal is to find $x \in S$ such that $f(x)$ is as small as possible, that is, a best answer. We write

$$\min_{x \in S} f(x)$$

to represent both the optimization problem and the minimal cost (if it exists). Occasionally, we wish to denote specifically the subset of S for which the minimal cost is attained; the standard notation is

$$\arg\min_{x \in S} f(x),$$

bearing in mind that this set might be empty. We will discuss a few simple applications which can be modeled in this form.

Rarely, optimization problems admit an analytical solution. Typically, we need numerical algorithms to (try to) solve them. Often, the best algorithms exploit mathematical structure in S and f.

An important special case arises when S is a linear space such as $\mathbb{R}^n$. Minimizing a function f in $\mathbb{R}^n$ is called *unconstrained optimization* because the variable x is free to move around $\mathbb{R}^n$, unrestricted.

If f is sufficiently differentiable and $\mathbb{R}^n$ is endowed with an inner product (i.e., if we make it into a Euclidean space), then we have a notion of gradient and perhaps even a notion of Hessian for f. These objects give us a firm understanding of how f behaves locally around any given point. Famous algorithms such as gradient descent and Newton's method exploit these objects to move around $\mathbb{R}^n$ efficiently in search of a solution.

Notice, however, that the Euclidean structure of $\mathbb{R}^n$ and the smoothness of f are irrelevant to the definition of the optimization problem itself: they are merely structures that we may (and as experience shows, we should) use algorithmically to our advantage.

Subsuming linearity, we focus on *smoothness* as the key structure to exploit: we assume the set S is a *smooth manifold* and the function f is smooth on S. This calls for precise definitions, constructed first in Chapter 3. For a first intuition, one can think

of smooth manifolds as surfaces in $\mathbb{R}^n$ that do not have kinks or boundaries, such as a plane, a sphere, a torus, or a hyperboloid.

We could think of optimization over such surfaces as *constrained*, in the sense that x is not allowed to move freely in $\mathbb{R}^n$: it is constrained to remain on the surface. Alternatively, and this is the viewpoint favored here, we can think of this as unconstrained optimization, in a world where the smooth surface is the only thing that exists: like an ant walking on a large ball might feel unrestricted in its movements, aware only of the sphere it lives on; or like the two-dimensional inhabitants of Flatland [Abb84] who find it hard to imagine that there exists such a thing as a third dimension, feeling thoroughly free in their own subspace.

A natural question then is: can we generalize the Euclidean algorithms from unconstrained optimization to handle the broader class of optimization over smooth manifolds? The answer is essentially yes, going back to the 1970s [Lue72, Lic79], the 1980s [Gab82] and the 1990s [Udr94, Smi94, HM96, Rap97, EAS98], and sparking a significant amount of research in the past two decades.

To generalize algorithms such as gradient descent and Newton's method, we need a proper notion of gradient and Hessian on smooth manifolds. In the linear case, this required the introduction of an inner product: a Euclidean structure. In our more general setting, we leverage the fact that smooth manifolds can be linearized locally around every point. The linearization at x is called the *tangent space* at x. By endowing each tangent space with its own inner product (varying smoothly with x, in a sense to be made precise), we construct what is called a *Riemannian structure* on the manifold: it becomes a *Riemannian manifold*.

A Riemannian structure is sufficient to define gradients and Hessians on the manifold, paving the way for optimization. There exist several Riemannian structures on each manifold: our choice may impact algorithmic performance. In that sense, identifying a useful structure is part of the algorithm design—as opposed to being part of the problem formulation, which ended with the definition of the search space (as a crude set) and the cost function.

Chapter 2 covers a few simple applications, mostly to give a sense of how manifolds come up. We then go on to define smooth manifolds in a restricted[1] setting in Chapter 3, where manifolds are *embedded* in a linear space, much like the unit sphere in three-dimensional space. In this context, we define notions of smooth functions, smooth vector fields, gradients and *retractions* (a means to move around on a manifold). These tools are sufficient to design and analyze a first optimization algorithm in Chapter 4: Riemannian gradient descent. As readers progress through these chapters, it is the intention that they also read bits of Chapter 7 from time to time: useful embedded manifolds are studied there in detail. Chapter 5 provides more advanced geometric tools for embedded manifolds, including the notions of Riemannian *connections* and

[1] Some readers may know Whitney's celebrated embedding theorems, which state that any smooth manifold can be embedded in a linear space [BC70, p. 82]. The mere existence of an embedding, however, is of little use for computation.

Hessians. These are put to good use in Chapter 6 to design and analyze Riemannian versions of Newton's method and the trust-region method.

The linear *embedding space* is useful for intuition, to simplify definitions, and to design tools. Notwithstanding, all the tools and concepts we define in the restricted setting are *intrinsic*, in the sense that they are well defined regardless of the embedding space. We make this precise much later, in Chapter 8, where all the tools from Chapters 3 and 5 are redefined in the full generality of standard treatments of differential geometry. This is also the time to discuss topological issues to some extent. Generality notably makes it possible to discuss a more abstract class of manifolds called *quotient manifolds* in Chapter 9. They offer a beautiful way to harness symmetry, so common in applications.

In closing, Chapter 10 offers a limited treatment of more advanced geometric tools such as the Riemannian distance, geodesics, the exponential map and its inverse, parallel transports and transporters, notions of Lipschitz continuity, finite differences, and covariant differentiation of tensor fields. Then, Chapter 11 covers elementary notions of convexity on Riemannian manifolds with simple implications for optimization. This topic has been around since the 1990s, and has been gaining traction in research lately.

More than 150 years ago, Riemann invented a new kind of geometry for the abstract purpose of understanding curvature in high-dimensional spaces. Today, this geometry plays a central role in the development of efficient algorithms to tackle technological applications Riemann himself—arguably—could have never envisioned. Through this book, I invite you to enjoy this singularly satisfying success of mathematics, with an eye to turning geometry into algorithms.

2 Simple examples

Before formally defining what manifolds are, and before introducing any particular algorithms, this chapter surveys simple problems that are naturally modeled as optimization on manifolds. These problems are motivated by applications in various scientific and technological domains. We introduce them chiefly to illustrate how manifolds arise and to motivate the mathematical abstractions in subsequent chapters.

The first example leads to optimization on an affine subspace: it falls within the scope of optimization on manifolds, but one can also handle it with classical tools. Subsequently, we encounter optimization on spheres, products of spheres, orthonormal matrices, the set of all linear subspaces, rotation matrices, fixed-rank matrices, positive definite matrices and certain quadratic surfaces. Through those, we get a glimpse of the wide reach of optimization on manifolds.

Below, we use a few standard concepts from linear algebra and calculus that are revisited in Section 3.1.

2.1 Sensor network localization from directions: an affine subspace

Consider n sensors located at unknown positions $t_1, \ldots, t_n$ in $\mathbb{R}^d$. We aim to locate the sensors, that is, estimate the positions t_i, based on some directional measurements. Specifically, for each pair of sensors (i, j) corresponding to an edge of a graph G, we receive a noisy measurement of the direction from t_j to t_i:

$$v_{ij} \approx \frac{t_i - t_j}{\|t_i - t_j\|},$$

where $\|x\| = \sqrt{x_1^2 + \cdots + x_d^2}$ is the Euclidean norm on $\mathbb{R}^d$ induced by the inner product $\langle u, v \rangle = u^\top v = u_1 v_1 + \cdots + u_d v_d$.

There are two fundamental ambiguities in this task. First, directional measurements reveal nothing about the global location of the sensors: translating the sensors as a whole does not affect pairwise directions. Thus, we may assume without loss of generality that the sensors are centered:

$$t_1 + \cdots + t_n = 0.$$

Second, the measurements reveal nothing about the global scale of the sensor arrangement. Specifically, scaling all positions t_i by a scalar $\alpha > 0$ as αt_i has no effect on the

directions separating the sensors so that the true scale cannot be recovered from the measurements. It is thus legitimate to fix the scale arbitrarily, to break symmetry. One fruitful way is to assume the following [HLV18]:

$$\sum_{(i,j)\in G} \left\langle t_i - t_j, v_{ij} \right\rangle = 1.$$

Indeed, if this constraint holds for some set of locations $t_1, \ldots, t_n$, then it does not hold for locations $\alpha t_1, \ldots, \alpha t_n$ unless $\alpha = 1$.

Given a tentative estimator $\hat{t}_1, \ldots, \hat{t}_n \in \mathbb{R}^d$ for the locations, we may assess its compatibility with the measurement v_{ij} by computing

$$\left\| (\hat{t}_i - \hat{t}_j) - \left\langle \hat{t}_i - \hat{t}_j, v_{ij} \right\rangle v_{ij} \right\|.$$

Indeed, if $\hat{t}_i - \hat{t}_j$ and v_{ij} are aligned in the same direction, this evaluates to zero. Otherwise, it evaluates to a positive number, growing as alignment degrades. Combined with the symmetry-breaking conditions, this suggests the following formulation for sensor network localization from direction measurements:

$$\begin{aligned} &\min_{\hat{t}_1,\ldots,\hat{t}_n \in \mathbb{R}^d} \sum_{(i,j)\in G} \left\| (\hat{t}_i - \hat{t}_j) - \left\langle \hat{t}_i - \hat{t}_j, v_{ij} \right\rangle v_{ij} \right\|^2 \\ &\text{subject to } \hat{t}_1 + \cdots + \hat{t}_n = 0 \text{ and } \sum_{(i,j)\in G} \left\langle \hat{t}_i - \hat{t}_j, v_{ij} \right\rangle = 1. \end{aligned}$$

The role of the second constraint is clear: it excludes $\hat{t}_1 = \cdots = \hat{t}_n = 0$, which would otherwise be optimal.

Grouping the variables as the columns of a matrix, we find that the search space for this problem is an affine subspace of $\mathbb{R}^{d\times n}$: this is a *linear manifold*. It is also an *embedded submanifold* of $\mathbb{R}^{d\times n}$. Hence it falls within our framework.

With the simple cost function as above, this problem is in fact a convex quadratic minimization problem on an affine subspace. As such, it admits an explicit solution which merely requires solving a linear system. Optimization algorithms can be used to solve this system implicitly. More importantly, the power of optimization algorithms lies in the flexibility that they offer: alternative cost functions may be used to improve robustness against specific noise models for example, and those require more general algorithms [HLV18].

2.2 Single extreme eigenvalue or singular value: spheres

Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix: $A = A^\top$. By the spectral theorem, A admits n real eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ and corresponding real, orthonormal eigenvectors $v_1, \ldots, v_n \in \mathbb{R}^n$, where orthonormality is assessed with respect to the standard inner product over $\mathbb{R}^n$: $\langle u, v\rangle = u^\top v$.

For now, we focus on computing one extreme eigenpair of A: (λ_1, v_1) or (λ_n, v_n) will do. Let $\mathbb{R}^n_*$ denote the set of nonzero vectors in $\mathbb{R}^n$. It is well known that the *Rayleigh quotient*,

$$r : \mathbb{R}^n_* \to \mathbb{R} : x \mapsto r(x) = \frac{\langle x, Ax \rangle}{\langle x, x \rangle},$$

attains its extreme values when x is aligned with $\pm v_1$ or $\pm v_n$, and that the corresponding value of the quotient is λ_1 or λ_n. We will rediscover such properties through the prism of optimization on manifolds as a running example in this book. One can gain some insight by checking that $r(v_i) = \lambda_i$.

Say we are interested in the smallest eigenvalue, λ_1. Then, we must solve the following optimization problem:

$$\min_{x \in \mathbb{R}^n_*} \frac{\langle x, Ax \rangle}{\langle x, x \rangle}.$$

The set $\mathbb{R}^n_*$ is open in $\mathbb{R}^n$: it is an *open submanifold* of $\mathbb{R}^n$. Optimization over an open set has its challenges (more on this later). Fortunately, we can easily circumvent these issues in this instance.

Since the Rayleigh quotient is invariant to scaling, that is, since $r(\alpha x) = r(x)$ for all nonzero real α, we may fix the scale arbitrarily. Given the denominator of r, one particularly convenient way is to restrict our attention to unit-norm vectors: $\|x\|^2 = \langle x, x \rangle = 1$. The set of such vectors is the *unit sphere* in $\mathbb{R}^n$:

$$\mathrm{S}^{n-1} = \{x \in \mathbb{R}^n : \|x\| = 1\}.$$

This is an *embedded submanifold* of $\mathbb{R}^n$. Our problem becomes

$$\min_{x \in \mathrm{S}^{n-1}} \langle x, Ax \rangle. \tag{2.1}$$

This is perhaps the simplest non-trivial instance of an optimization problem on a manifold: we use it recurringly to illustrate concepts as they occur.

Similarly to the above, we may compute the largest singular value of a matrix $M \in \mathbb{R}^{m \times n}$ together with associated left- and right-singular vectors by solving

$$\max_{x \in \mathrm{S}^{m-1}, y \in \mathrm{S}^{n-1}} \langle x, My \rangle. \tag{2.2}$$

This is the basis of *principal component analysis*: see also Section 2.4. The search space is a Cartesian product of two spheres. This too is a manifold, specifically, an embedded submanifold of $\mathbb{R}^m \times \mathbb{R}^n$. In general:

Products of manifolds are manifolds.

This is an immensely useful property.

2.3 Dictionary learning: products of spheres

JPEG and its more recent version JPEG 2000 are some of the most commonly used compression standards for photographs. At their core, these algorithms rely on basis expansions: discrete cosine transforms for JPEG, and wavelet transforms for JPEG 2000. That is, an image (or rather, each patch of the image) is written as a linear combination of a fixed collection of basis images. To fix notation, say an image is

represented as a vector $y \in \mathbb{R}^d$ (its pixels rearranged into a single column vector) and the basis images are $b_1, \ldots, b_d \in \mathbb{R}^d$ (each of unit norm). There exists a unique set of coordinates $c \in \mathbb{R}^d$ such that

$$y = c_1 b_1 + \cdots + c_d b_d.$$

Since the basis images are fixed (and known to anyone creating or reading image files in this format), it is equivalent to store y or c.

The basis is designed carefully with two goals in mind. First, the transform between y and c should be fast to compute (one good starting point to that effect is orthogonality). Second, images encountered in practice should lead to many of the coefficients c_i being zero, or close to zero. Indeed, to recover y, it is only necessary to record the nonzero coefficients. To compress further, we may also decide not to store the small coefficients: if so, y can still be reconstructed approximately. Beyond compression, another benefit of sparse expansions is that they can reveal structural information about the contents of the image.

In *dictionary learning*, we focus on the second goal. As a key departure from the above, the idea here is not to design a basis by hand, but rather to learn a good basis from data automatically. This way, we may exploit structural properties of images that come up in a particular application. For example, it may be the case that photographs of faces can be expressed more sparsely in a dedicated basis as compared to a standard wavelet basis. Pushing this idea further, we relax the requirement of identifying a basis, instead allowing ourselves to pick more than d images for our expansions. The collection of images $b_1, \ldots, b_n \in \mathbb{R}^d$ forms a *dictionary*. Its elements are called *atoms*, and they normally span $\mathbb{R}^d$ in an overcomplete way, meaning any image y can be expanded into a linear combination of atoms in more than one way. The aim is that at least one of these expansions should be sparse, or have many small coefficients. For the magnitudes of coefficients to be meaningful, we further require all atoms to have the same norm: $\|b_i\| = 1$ for all i.

Thus, given a collection of k images $y_1, \ldots, y_k \in \mathbb{R}^d$, the task in dictionary learning is to find atoms $b_1, \ldots, b_n \in \mathbb{R}^d$ such that (as much as possible) each image y_i is a sparse linear combination of the atoms. Collect the input images as the columns of a data matrix $Y \in \mathbb{R}^{d \times k}$, and the atoms into a matrix $D \in \mathbb{R}^{d \times n}$ (to be determined). Expansion coefficients for the images in this dictionary form the columns of a matrix $C \in \mathbb{R}^{n \times k}$ so that

$$Y = DC.$$

Typically, many choices of C are possible. We aim to pick D such that there exists a valid (or approximately valid) choice of C with numerous zeros. Let $\|C\|_0$ denote the number of entries of C different from zero. Then, one possible formulation of dictionary learning balances both aims with a parameter $\lambda > 0$ as (with $b_1, \ldots, b_n$ the columns of the dictionary matrix D):

$$\begin{aligned} \min_{D \in \mathbb{R}^{d \times n}, C \in \mathbb{R}^{n \times k}} \ & \|Y - DC\|^2 + \lambda \|C\|_0 \\ \text{subject to } & \|b_1\| = \cdots = \|b_n\| = 1. \end{aligned} \tag{2.3}$$

The matrix norm $\|\cdot\|$ is the Frobenius norm, induced by the standard inner product $\langle U, V\rangle = \mathrm{Tr}(U^\top V)$.

Evidently, allowing the dictionary to be overcomplete ($n > d$) helps sparsity. An extreme case is to set $n = k$, in which case an optimal solution consists in letting D be Y with normalized columns. Then, each image can be expressed with a single nonzero coefficient (C is diagonal). This is useless of course, if only because both parties of the communication must have access to the (possibly huge) dictionary, and because this choice may generalize poorly when presented with new images. Interesting scenarios involve n much smaller than k.

The search space for D is a product of several spheres, which is an embedded submanifold of $\mathbb{R}^{d\times n}$ called the *oblique manifold*:

$$\mathrm{OB}(d,n) = (\mathrm{S}^{d-1})^n = \left\{X \in \mathbb{R}^{d\times n} : \mathrm{diag}(X^\top X) = \mathbf{1}\right\},$$

where $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector and $\mathrm{diag}\colon \mathbb{R}^{n\times n} \to \mathbb{R}^n$ extracts the diagonal entries of a matrix. The search space in C is the linear manifold $\mathbb{R}^{n\times k}$. Overall, the search space of the dictionary learning optimization problem is

$$\mathrm{OB}(d,n) \times \mathbb{R}^{n\times k},$$

which is an embedded submanifold of $\mathbb{R}^{d\times n} \times \mathbb{R}^{n\times k}$.

We note in closing that the cost function in (2.3) is discontinuous because of the term $\|C\|_0$, making it hard to optimize. A standard reformulation replaces the culprit with $\|C\|_1$: the sum of absolute values of the entries of C. This is continuous but nonsmooth. A possible further step then is to smooth the cost function, for example exploiting that $|x| \approx \sqrt{x^2 + \varepsilon^2}$ or $|x| \approx \varepsilon \log(e^{x/\varepsilon} + e^{-x/\varepsilon})$ for small $\varepsilon > 0$: these are standard tricks.

Regardless of changes to the cost function, the manifold $\mathrm{OB}(d,n)$ is non-convex so that finding a global optimum for dictionary learning as stated above is challenging: see work by Sun et al. [SQW17] for some guarantees.

2.4 Principal component analysis: Stiefel and Grassmann

Let $x_1, \ldots, x_n \in \mathbb{R}^d$ represent a large collection of centered data points in a d-dimensional linear space. We may think of it as a cloud of points. It may be the case that this cloud lies on or near a low-dimensional subspace of $\mathbb{R}^d$, and it may be distributed anisotropically in that subspace, meaning it shows more variation along some directions than others. One of the pillars of data analysis is to determine the main directions of variation of the data. This goes by the name of principal component analysis (PCA), which we encountered in Section 2.2.

One way to think of a main direction of variation, called a *principal component*, is as a vector $u \in \mathrm{S}^{d-1}$ such that projecting the data points to the one-dimensional subspace spanned by u "preserves most of the variance." Specifically, let $X \in \mathbb{R}^{d\times n}$ be the matrix whose columns are the data points and let $uu^\top$ be the orthogonal projector

from $\mathbb{R}^d$ to the span of u. We wish to maximize the following for $u \in \mathrm{S}^{d-1}$:

$$\sum_{i=1}^{n} \|uu^\top x_i\|^2 = \|uu^\top X\|^2 = \langle uu^\top X, uu^\top X\rangle = \langle XX^\top u, u\rangle.$$

We recognize the Rayleigh quotient of the matrix $XX^\top$ to be maximized for u over S^{d-1} (Section 2.2). An optimal solution is given by a dominant eigenvector of $XX^\top$, or equivalently by a dominant left singular vector of X.

Let $u_1 \in \mathrm{S}^{d-1}$ be a principal component. We would like to find a second one. That is, we aim to find $u_2 \in \mathrm{S}^{d-1}$, *orthogonal to* u_1, such that projecting the data to the subspace spanned by u_1 and u_2 preserves the most variance. The orthogonal projector to that subspace is $u_1u_1^\top + u_2u_2^\top$. We maximize

$$\|(u_1u_1^\top + u_2u_2^\top)X\|^2 = \langle XX^\top u_1, u_1\rangle + \langle XX^\top u_2, u_2\rangle$$

over $u_2 \in \mathrm{S}^{d-1}$ with $u_2^\top u_1 = 0$. The search space for u_2 is an embedded submanifold of $\mathbb{R}^d$: it is a unit sphere in the subspace orthogonal to u_1.

It is often more convenient to optimize for u_1 and u_2 simultaneously rather than sequentially. Then, since the above cost function is symmetric in u_1 and u_2, as is the constraint $u_2^\top u_1 = 0$, we add weights to the two terms to ensure u_1 captures a principal component and u_2 captures a second principal component:

$$\max_{u_1,u_2 \in \mathrm{S}^{d-1}, u_2^\top u_1 = 0} \alpha_1 \langle XX^\top u_1, u_1\rangle + \alpha_2 \langle XX^\top u_2, u_2\rangle,$$

with $\alpha_1 > \alpha_2 > 0$ arbitrary.

More generally, aiming for k principal components, we look for a matrix $U \in \mathbb{R}^{d\times k}$ with k orthonormal columns $u_1, \ldots, u_k \in \mathbb{R}^d$. The set of such matrices is called the *Stiefel manifold*:

$$\mathrm{St}(d,k) = \{U \in \mathbb{R}^{d\times k} : U^\top U = I_k\},$$

where I_k is the identity matrix of size k. It is an embedded submanifold of $\mathbb{R}^{d\times k}$. The orthogonal projector to the subspace spanned by the columns of U is $UU^\top$. Hence PCA amounts to solving the problem

$$\max_{U \in \mathrm{St}(d,k)} \sum_{i=1}^{k} \alpha_i \langle XX^\top u_i, u_i\rangle = \max_{U \in \mathrm{St}(d,k)} \langle XX^\top U, UD\rangle, \tag{2.4}$$

where $D \in \mathbb{R}^{k\times k}$ is diagonal with diagonal entries $\alpha_1 > \cdots > \alpha_k > 0$.

It is well known that collecting k top eigenvectors of $XX^\top$ (or, equivalently, k top left singular vectors of X) yields a global optimum of (2.4), meaning this optimization problem can be solved efficiently using tools from numerical linear algebra. Still, the optimization perspective offers significant flexibility that standard linear algebra algorithms cannot match. Specifically, within an optimization framework, it is possible to revisit the variance criterion by changing the cost function. This allows one to promote sparsity or robustness against outliers, for example, to develop variants such as sparse PCA [dBEG08, JNRS10] and robust PCA [MT11, GZAL14, MZL19, NNSS20]. There may also be computational advantages, for example, in tracking and online

models where the dataset changes or grows with time: it may be cheaper to update a previously computed good estimator using few optimization steps than to run a complete eigenvalue or singular value decomposition anew.

If the top k principal components are of interest but their ordering is not, then we do not need the weight matrix D. In this scenario, we are seeking an orthonormal basis U for a k dimensional subspace of $\mathbb{R}^d$ such that projecting the data to that subspace preserves as much of the variance as possible. This description makes it clear that the particular basis is irrelevant: only the selected subspace matters. This is apparent in the cost function,

$$f(U) = \langle XX^\top U, U\rangle,$$

which is invariant under orthogonal transformations. Specifically, for all Q in the orthogonal group

$$\mathrm{O}(k) = \{Q \in \mathbb{R}^{k\times k} : Q^\top Q = I_k\},$$

we have $f(UQ) = f(U)$. This induces an *equivalence relation*[1] $\sim$ on the Stiefel manifold:

$$U \sim V \qquad \iff \qquad V = UQ \text{ for some } Q \in \mathrm{O}(k).$$

This equivalence relation partitions $\mathrm{St}(d,k)$ into *equivalence classes*:

$$[U] = \{V \in \mathrm{St}(d,k) : U \sim V\} = \{UQ : Q \in \mathrm{O}(k)\}.$$

The set of equivalence classes is called the *quotient set*:

$$\mathrm{St}(d,k)/\sim = \mathrm{St}(d,k)/\mathrm{O}(k) = \{[U] : U \in \mathrm{St}(d,k)\}.$$

Importantly, $U, V \in \mathrm{St}(d,k)$ are equivalent if and only if their columns span the same subspace of $\mathbb{R}^d$. In other words: the quotient set is in one-to-one correspondence with the set of subspaces of dimension k in $\mathbb{R}^d$. With the right geometry, the latter is called the *Grassmann manifold*:

$$\mathrm{Gr}(d,k) = \{ \text{ subspaces of dimension } k \text{ in } \mathbb{R}^d \} \equiv \mathrm{St}(d,k)/\mathrm{O}(k),$$

where the symbol $\equiv$ reads "is equivalent to" (context indicates in what sense). As defined here, the Grassmann manifold is a *quotient manifold*. This type of manifold is more abstract than embedded submanifolds, but we can still develop numerically efficient tools to work with them.

Within our framework, computing the dominant eigenspace of dimension k of the matrix $XX^\top$ can be written as

$$\max_{[U]\in\mathrm{Gr}(d,k)} \langle XX^\top U, U\rangle.$$

[1] Recall that an equivalence relation $\sim$ on a set $\mathcal{M}$ is a reflexive ($a \sim a$), symmetric ($a \sim b \iff b \sim a$) and transitive ($a \sim b$ and $b \sim c \implies a \sim c$) binary relation. The equivalence class $[a]$ is the set of elements of $\mathcal{M}$ that are equivalent to a. Each element of $\mathcal{M}$ belongs to exactly one equivalence class.