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:

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

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*,

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

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):

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

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:

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

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.

The cost function is well defined over $\mathrm{Gr}(d,k)$ since it depends only on the equivalence class of U, not on U itself.

Going back to (2.4), we note in passing that k top left and right singular vectors of a matrix $M \in \mathbb{R}^{m\times n}$ can be computed by solving the following problem on a product of Stiefel manifolds (this and (2.4) are sometimes called *Brockett cost functions*):

$$\max_{U\in\mathrm{St}(m,k),V\in\mathrm{St}(n,k)} \langle MV, UD\rangle,$$

where $D = \mathrm{diag}(\alpha_1,\ldots,\alpha_k)$ with arbitrary $\alpha_1 > \cdots > \alpha_k > 0$ as above.

A book by Trendafilov and Gallo provides more in-depth discussion of applications of optimization on manifolds to data analysis [TG21].

2.5 Synchronization of rotations: special orthogonal group

In structure from motion (SfM), the 3D structure of an object is to be reconstructed from several 2D images of it. For example, in the paper "Building Rome in a Day" [ASS+09], the authors automatically construct a model of the Colosseum from over 2000 photographs freely available on the Internet. Because the pictures are acquired from an unstructured source, one of the steps in the reconstruction pipeline is to estimate camera locations and *pose*. The pose of a camera is its orientation in space.

In single particle reconstruction through cryo electron microscopy, an electron microscope is used to produce 2D tomographic projections of biological objects such as proteins and viruses. Because shape is a determining factor of function, the goal is to estimate the 3D structure of the object from these projections. Contrary to X-ray crystallography (another fundamental tool of structural biology), the orientations of the objects in the individual projections are unknown. In order to estimate the structure, a useful step is to estimate the individual orientations (though high noise levels do not always allow it, in which case alternative statistical techniques must be used).

Mathematically, orientations correspond to rotations of $\mathbb{R}^3$. Rotations in $\mathbb{R}^d$ can be represented with orthogonal matrices:

$$\mathrm{SO}(d) = \{R \in \mathbb{R}^{d\times d} : R^\top R = I_d \text{ and } \det(R) = +1\}.$$

The determinant condition excludes reflections of $\mathbb{R}^d$. The set $\mathrm{SO}(d)$ is the *special orthogonal group*: it is both a group (in the mathematical sense of the term) and a manifold (an embedded submanifold of $\mathbb{R}^{d\times d}$)—it is a *Lie group*.

In both applications described above, similar images or projections can be compared to estimate relative orientations. *Synchronization of rotations* is a mathematical abstraction of the ensuing task. It consists in estimating n individual rotation matrices,

$$R_1,\ldots,R_n \in \mathrm{SO}(d),$$

from pairwise relative rotation measurements: for some pairs (i,j) corresponding to the edges of a graph G, we observe a noisy version of $R_iR_j^{-1}$. Let $H_{ij} \in \mathrm{SO}(d)$ denote such a measurement. Then, one possible formulation of synchronization of rotations is

$$\min_{\hat{R}_1,\ldots,\hat{R}_n \in \mathrm{SO}(d)} \sum_{(i,j)\in G} \|\hat{R}_i - H_{ij}\hat{R}_j\|^2.$$

This is an optimization problem over $\mathrm{SO}(d)^n$, which is a manifold.

This also comes up in *simultaneous localization and mapping* (SLAM), whereby a robot must simultaneously map its environment and locate itself in it as it moves around [RDTEL21]. An important aspect of SLAM is to keep track of the robot's orientation accurately, by integrating all previously acquired information to correct estimator drift.

2.6 Low-rank matrix completion: fixed-rank manifold

Let $M \in \mathbb{R}^{m\times n}$ be a large matrix of interest. Given some of its entries, our task is to estimate the whole matrix. A commonly cited application for this setup is that of recommender systems, where row i corresponds to a user, column j corresponds to an item (a movie, a book ...), and entry M_{ij} indicates how much user i appreciates item j: positive values indicate appreciation, zero is neutral, and negative values indicate dislike. The known entries may be collected from user interactions. Typically, most entries are unobserved. Predicting the missing values may be helpful to automate personalized recommendations.

Of course, without further knowledge about how the entries of the matrix are related, the completion task is ill-posed. Hope comes from the fact that certain users share similar traits so that what one user likes may be informative about what another, similar, user may like. In the same spirit, certain items may be similar enough that whole groups of users may feel similarly about them. One mathematically convenient way to capture this idea is to assume M has (approximately) low rank. The rationale is as follows: if M has rank r, then it can be factored as

$$M = LR^\top,$$

where $L \in \mathbb{R}^{m\times r}$ and $R \in \mathbb{R}^{n\times r}$ are full-rank factor matrices. Row i of L, ℓ_i, attributes r numbers to user i, while the jth row of R, r_j, attributes r numbers to item j. Under the low-rank model, the rating of user i for item j is $M_{ij} = \langle \ell_i, r_j\rangle$. One interpretation is that there are r latent features (these could be movie genres, for example): a user has some positive or negative appreciation for each feature, and an item has traits aligned with or in opposition to these features; the rating is obtained as the inner product of the two feature vectors.

Under this model, predicting the user ratings for all items amounts to *low-rank matrix completion*. Let Ω denote the set of pairs (i,j) such that M_{ij} is observed. Allowing for noise in the observations and inaccuracies in the model, we aim to solve

$$\min_{X\in\mathbb{R}^{m\times n}} \sum_{(i,j)\in\Omega} (X_{ij} - M_{ij})^2$$
$$\text{subject to } \operatorname{rank}(X) = r.$$

The search space for this optimization problem is the set of matrices of a given size and rank:

$$\mathbb{R}_r^{m\times n} = \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = r\}.$$

This set is an embedded submanifold of $\mathbb{R}^{m\times n}$ which is frequently useful in machine learning applications.

Another use for this manifold is solving high-dimensional matrix equations that may come up in systems and control applications: aiming for a low-rank solution may be warranted in certain settings, and exploiting this can lower the computational burden substantially. Yet another context where optimization over low-rank matrices occurs is in completing and denoising approximately separable bivariate functions based on sampled values [Van10, Van13, MV13].

The same set can also be endowed with other geometries, that is, it can be made into a manifold in other ways. For example, exploiting the factored form more directly, note that any matrix in $\mathbb{R}_r^{m\times n}$ admits a factorization as $LR^\top$ with both L and R of full rank r. This correspondence is not one-to-one, however, since the pairs (L, R) and $(LJ^{-1}, RJ^\top)$ map to the same matrix in $\mathbb{R}_r^{m\times n}$ for all invertible matrices J: they are equivalent. Similarly to the Grassmann manifold, this leads to a definition of $\mathbb{R}_r^{m\times n}$ as a quotient manifold instead of an embedded submanifold. Many variations on this theme are possible, some of them more useful than others depending on the application [Mey11, Mis14].

The set $\mathbb{R}_r^{m\times n}$ is not closed in $\mathbb{R}^{m\times n}$, which may create difficulties for optimization. The closure of the set corresponds to all matrices of rank at most r (rather than exactly equal to r). That set is not a manifold, but it can be smoothly parameterized by a manifold in several ways [LKB22]. One particularly simple way is through the map $(L, R) \mapsto LR^\top$, where L and R are allowed to be rank deficient.

2.7 Gaussian mixture models: positive definite matrices

A common model in machine learning assumes data $x_1, \ldots, x_n \in \mathbb{R}^d$ are sampled independently from a *mixture of K Gaussians*, that is, each data point is sampled from a probability distribution with density of the form

$$f(x) = \sum_{k=1}^{K} w_k \frac{1}{\sqrt{2\pi \det(\Sigma_k)}} e^{-\frac{(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)}{2}},$$

where the centers $\mu_1, \ldots, \mu_K \in \mathbb{R}^d$; covariances $\Sigma_1, \ldots, \Sigma_K \in \operatorname{Sym}(d)^+$; and mixing probabilities $(w_1, \ldots, w_K) \in \Delta_+^{K-1}$ are to be determined. We use the following notation:

$$\operatorname{Sym}(d)^+ = \{X \in \mathbb{R}^{d\times d} : X = X^\top \text{ and } X \succ 0\}$$

for symmetric, positive definite matrices of size d, and

$$\Delta_+^{K-1} = \{w \in \mathbb{R}^K : w_1, \ldots, w_K > 0 \text{ and } w_1 + \cdots + w_K = 1\}$$

for the positive part of the simplex, that is, the set of non-vanishing discrete probability distributions over K objects. In this model, with probability w_k, a point x is sampled from the kth Gaussian, with mean μ_k and covariance Σ_k. The aim is only to estimate the parameters, not to estimate which Gaussian each point x_i was sampled from.

For a given set of observations $x_1, \ldots, x_n$, a maximum likelihood estimator solves

$$\max_{\substack{\hat{\mu}_1,\ldots,\hat{\mu}_K \in \mathbb{R}^d, \\ \hat{\Sigma}_1,\ldots,\hat{\Sigma}_K \in \mathrm{Sym}(d)^+, \\ w \in \Delta_+^{K-1}}} \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} w_k \frac{1}{\sqrt{2\pi \det(\Sigma_k)}} e^{-\frac{(x_i-\mu_k)^\top \Sigma_k^{-1} (x_i-\mu_k)}{2}} \right). \tag{2.5}$$

This is an optimization problem over $\mathbb{R}^{d\times K} \times (\mathrm{Sym}(d)^+)^K \times \Delta_+^{K-1}$, which can be made into a manifold because $\mathrm{Sym}(d)^+$ and Δ_+^{K-1} can be given a manifold structure.

The direct formulation of maximum likelihood estimation for Gaussian mixture models in (2.5) is, however, not computationally favorable. See [HS15] for a beneficial reformulation, still on a manifold.

2.8 Smooth semidefinite programs

Semidefinite programs (SDPs) are optimization problems of the form

$$\min_{X \in \mathrm{Sym}(n)} \langle C, X \rangle \qquad \text{subject to} \qquad \mathcal{A}(X) = b \qquad \text{and} \qquad X \succeq 0, \tag{2.6}$$

where $\mathrm{Sym}(n)$ is the space of real, symmetric matrices of size $n\times n$, $\langle A, B\rangle = \mathrm{Tr}(A^\top B)$, $\mathcal{A} \colon \mathrm{Sym}(n) \to \mathbb{R}^m$ is a linear map defined by m symmetric matrices $A_1, \ldots, A_m$ as $\mathcal{A}(X)_i = \langle A_i, X\rangle$, and $X \succeq 0$ means X is positive semidefinite.

SDPs are convex and they can be solved to global optimality in polynomial time using interior point methods [Nes18, §5.4.4]. Still, handling the positive semidefiniteness constraint $X \succeq 0$ and the dimensionality of the problem (namely, the $\frac{n(n+1)}{2}$ variables required to define X) both pose significant computational challenges for large n.

A popular way to address both issues is the Burer–Monteiro approach [BM05], which consists in factorizing X as $X = YY^\top$ with $Y \in \mathbb{R}^{n\times p}$: the number p of columns of Y is a parameter. Notice that X is now automatically positive semidefinite. If $p \geq n$, the SDP can be rewritten equivalently as

$$\min_{Y \in \mathbb{R}^{n\times p}} \langle CY, Y \rangle \qquad \text{subject to} \qquad \mathcal{A}(YY^\top) = b. \tag{2.7}$$

If $p < n$, this corresponds to the SDP with the additional constraint $\mathrm{rank}(X) \leq p$. There is a computational advantage to taking p as small as possible. Interestingly, if the set of matrices X that are feasible for the SDP is compact, then the *Pataki–Barvinok bound* [Pat98, Bar95] provides that at least one of the global optimizers of the SDP has rank r such that $\frac{r(r+1)}{2} \leq m$. In other words: assuming compactness, the Burer–Monteiro formulation (2.7) is *equivalent* to the original SDP so long as p satisfies $\frac{p(p+1)}{2} \geq m$. This is already the case for $p = O(\sqrt{m})$, which may be significantly smaller than n.

The positive semidefiniteness constraint disappeared, and the dimensionality of the problem went from $O(n^2)$ to np—a potentially appreciable gain. Yet, we lost something important along the way: the Burer–Monteiro problem is not convex. It is not immediately clear how to solve it.

The search space of the Burer–Monteiro problem is the set of feasible Y:

$$\mathcal{M} = \{Y \in \mathbb{R}^{n\times p} : \mathcal{A}(YY^\top) = b\}. \tag{2.8}$$

Assume the map $Y \mapsto \mathcal{A}(YY^\top)$ has the property that its differential at all Y in $\mathcal{M}$ has rank m. Then, $\mathcal{M}$ is an embedded submanifold of $\mathbb{R}^{n\times p}$. In this special case, we may try to solve the Burer–Monteiro problem through optimization over that manifold. It turns out that non-convexity is mostly benign in that scenario, in a precise sense [BVB19]:

If $\mathcal{M}$ is compact and $\frac{p(p+1)}{2} > m$, then, for a generic cost matrix C, the smooth optimization problem $\min_{Y\in\mathcal{M}} \langle CY, Y\rangle$ has no spurious local minima, in the sense that any point Y which satisfies first- and second-order necessary optimality conditions is a global optimum.

(Necessary optimality conditions are detailed in Sections 4.2 and 6.1.) Additionally, these global optima map to global optima of the SDP through $X = YY^\top$. This suggests that smooth-and-compact SDPs may be solved to global optimality via optimization on manifolds. The requirement that $\mathcal{M}$ be a regularly defined smooth manifold is not innocuous, but it is satisfied in a number of interesting applications.

There has been a lot of work on this front in recent years, including the early work by Burer and Monteiro [BM03, BM05], the first manifold-inspired perspective by Journée et al. [JBAS10], qualifications of the benign non-convexity at the Pataki–Barvinok threshold [BVB16, BVB19] and below in special cases [BBV16], a proof that p cannot be set much lower than that threshold in general [WW20], smoothed analyses to assess whether points which satisfy necessary optimality conditions approximately are also approximately optimal [BBJN18, PJB18, CM19] and extensions to accommodate scenarios where $\mathcal{M}$ is not a smooth manifold but, more generally, a real algebraic variety [BBJN18, Cif21]. See all these references for applications, including Max-Cut, community detection, the trust-region subproblem, synchronization of rotations and more.