IFT 6132 Final Project
Introduction to Manifold Optimization and Geodesic Convexity

Philippe Laferriere [*1] **Samuel Laferriere** [*1]

Abstract

In this short report, we attempt to give a (very) brief overview of manifold optimization. Manifold optimization's goal is to generalize optimization from flat Euclidean spaces to the larger domain of Riemannian manifolds. Not only does it promise to extend classical algorithms such as steepest descent and Newton's method to a whole new set of problems, but also to rethink established solutions to classical problems, such as learning Gaussian mixture models, and in so doing outperform prevailing methods. Given the brevity of this report, it is impossible for us to define every object formally while also giving many examples. We thus expect readers to be already somewhat familiar with either differential geometry or optimization. Experts of the former will hopefully enjoy learning about a new concrete application of their domain of research, and experts of the latter will discover new methods to extend their toolbox.

The word manifold, to most people who haven't formally studied them, probably only brings to mind vague pictures resembling those in Figure 1. Seeing it next to the word optimization probably leaves them wondering why we would ever want to optimize functions on such exotic looking surfaces. Indeed, apart from the sphere, which arises naturally in eigenvector optimization problems (see section 3), most of these classical manifolds studied by mathematicians won't be of much interest to us. Most of the manifolds that will arise from optimization problems will be matrix manifolds. Some of them will be easy to grasp and visualize, such as the good old Euclidean space[1] $\mathbb{R}^n$, but others will be much more abstract. Nonetheless, thanks to the Whitney embedding theorem, we know that every manifold can be embedded into some large Euclidean space, and so can safely be thought of as just being a constrained set, that is, a subset of $\mathbb{R}^n$ for some large n. In this way, every open subset of $\mathbb{R}^n$ can be turned into a manifold by adding some trivial structure. Manifold optimization, however, refers to a special set of techniques which requires further restricting the subsets considered, to those that admit a "smooth" manifold structure, as well as a special object called a Riemannian metric. These special subsets, with their added structure, are called Riemannian manifolds.

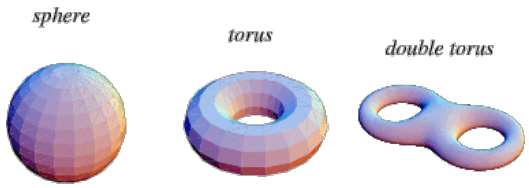

Figure 1. Classical Compact Manifolds. (Renze et al.)

The theory behind manifold optimization is now well developed (Edelman et al., 1998; Absil et al., 2008), and software packages are being created to give access to these great tools for nonspecialists. Manopt (Boumal et al., 2014) is probably the most well known, and has greatly lowered the barrier of entry for applied researchers and engineers of other disciplines. Pymanopt (Townsend et al., 2016) has further given manopt access to python's great auto-differentiation facilities, alleviating the need to implement derivatives and hessians by hand. This combination really leaves no excuse not to start using this great technology. We hope that this report will help spread the word about this field.

1. General Setting

Any optimization problem $\min_{x \in X} f(x)$ only contains two components: the domain X over which we are optimizing, and the scalar cost function $f : X \to \mathbb{R}$ that we want to minimize. Most solution methods are iterative. That is, they start from an initial guess $x_0 \in X$ and then improve on that guess iteratively. Gradient descent (GD) is a well known and simple example, which moves in the direction opposite

[*]Equal contribution [1]Department of Computer Science and Operations Research, University of Montreal, Montreal, Canada. Correspondence to: Samuel Laferriere <samlaf92@gmail.com>, Philippe Laferriere <laferriere.phil@gmail.com>.

IFT 6132 project report - Université de Montréal

[1]which can be, using its standard topology, trivially turned into a manifold.

the gradient at each point: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, with α_k being a predefined step-size.

Manifold optimization is an optimization problem where X is a manifold (see section 2). To make the distinction clear, we often use M to denote the manifold and p to denote points in it. This way, the optimization problem is written as: $\min_{p \in M} f(p)$. See for example Algorithm 1. This seemingly innocent change however, hides a lot of complexity, which we will unravel one step at a time.

Our running example throughout this report will be finding the largest eigenvalue of a symmetric matrix A. This problem can be written as (Ghojogh et al., 2019) $\min_{||x||^2=1} -x^T Ax$ (the eigenvector associated to this eigenvalue is found by changing the min to an argmin). Hence, if we write $f(x) = -x^T Ax$, and realize that the unit sphere $S^{n-1} = \{x \in \mathbb{R}^n : ||x||^2 = 1\}$ can be turned into a manifold, this problem can be written as both $\min_{x \in \mathbb{R}^n, ||x||^2=1} f(x)$, and $\min_{x \in S^{n-1}} f(x)$. That is, we can see the problem either as a constrained problem in a Euclidean space, or as an unconstrained problem on a manifold. We will see that the manifold perspective has its own advantages.

This problem, written in Pymanopt, requires nothing more than those three simple ingredients:

```python
import pymanopt
from pymanopt.manifolds import Sphere
from pymanopt.solvers import SteepestDescent
import autograd.numpy as np

# (1) Instantiate the manifold M
manifold = Sphere(100)

# (2) Define the cost function f
A = np.random.randn(100,100)
A = A + A.T # to make A symmetric
@pymanopt.function.Autograd
def cost(x): return - x.T @ A @ x

# (3) Instantiate a Pymanopt solver
solver = SteepestDescent()

# (4) Optimize
problem = pymanopt.Problem(manifold, cost)
Xopt = solver.solve(problem)
```

Source Code 1. Largest eigenvalue using Pymanopt.

Our main goal for this article is to understand what the SteepestDescent solver is doing. If we stare at that GD iterate for Euclidean spaces once again, $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, we find that it doesn't make sense in the manifold settings. For one, how is the gradient even defined? Remember that the gradient in Euclidean spaces is defined as the vector of partials $\nabla f(x) = (\frac{\partial f}{\partial x^1}(x), \dots, \frac{\partial f}{\partial x^n}(x))$, where the ith partial derivative $\frac{\partial f}{\partial x^i}$ is directional derivative of f in the direction of the standard basis vector e_i. Formally,

$$\frac{\partial f}{\partial x^i}(x) = \lim_{t \to 0} \frac{f(\overbrace{x + te_i}^{!?}) - f(x)}{t}$$

Note the "!?" overbrace. There is no linear structure on a manifold, so not only is there no such thing as a standard basis vector e_i, the operations of addition and scalar multiplication in $x + te_i$ are not even defined! Fortunately, there is a way around this problem. A manifold constrains the directions in which we are able to move in the ambient Euclidean space. The allowed directions are those tangent to the manifold, which is a linear space! We will be able to define these tangent spaces abstractly so as not to make reference to the ambient space, and the gradient, the direction of "fastest increase", will then live on this space.

Our last problem when trying to mimic the GD update $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$ is that x_k lives on the manifold whereas $\nabla f(x_k)$ lives on the tangent space, so we cannot add them. The last missing piece will be a retraction function (Absil & Malick, 2012) that moves from the tangent space back into the manifold. See Figure 5 for a graphical depiction of this concept.

With these concepts in hand, we now have our first algorithm on manifolds.

Algorithm 1 Riemannian Gradient Descent

Input: Riemannian Manifold M, cost function f, step sizes α_k, convergence criterion ϵ.
Initialize $p_0 \in M$, $k = 0$.
repeat
  $g_k = \text{grad} f|_{p_k}$ # Compute the gradient of f at p_k
  $p_{k+1} = \text{Retraction}_{p_k}(-\alpha g_k)$ # Step in direction $-g_k$
  $k := k + 1$
until $f(p_k) - f(p_{k-1}) < \epsilon$
Output: p_k

2. Manifold

Before going anywhere, we must start by having a solid foundation about the very space we are optimizing over. We start by defining manifolds, then explain what smooth manifolds and their tangent spaces are, and finally get to Riemannian manifolds, which add an inner product on the tangent spaces of smooth manifolds.

Formally, a manifold is a topological space that is locally

homeomorphic to $\mathbb{R}^n$. In order to understand this statement, we need to first define what a topological space is, and what a homeomorphism is.

A topological space is a 2-tuple structure $(X, \mathcal{O})$, where X is a set (called the underlying space), and $\mathcal{O} \subseteq \mathcal{P}(X)$ is a subset of the power set of $\mathcal{M}$ (called the topology on X). The sets of $\mathcal{O}$ are called open sets. A topology on top of a set permits defining the notion of continuity of functions. Given two topological spaces $(X, \mathcal{O}_X)$ and $(Y, \mathcal{O}_Y)$, a function $f : X \to Y$ is continuous if $f^{-1}(U) \in \mathcal{O}_X$ for all $U \in \mathcal{O}_Y$. In words, f is continuous if the pre-image of every open set is open. In this respect, a topology is the bare minimum structure that, when added to a set, permits talking about continuity of functions. Now, given two topological spaces, we can compare them, and ask whether they are "equivalent" in some sense. This notion is made formal by homeomorphisms. A homeomorphism is a bijective function between two topological spaces that is continuous and whose inverse is also continuous. Hence, if $f : (X, \mathcal{O}_X) \to (Y, \mathcal{O}_Y)$ is a homeomorphism, then the bijection makes underlying sets X and Y equivalent, and the continuity of f and its f^{-1} make the topologies $\mathcal{O}_X$ and $\mathcal{O}_Y$ equivalent. This is why two spaces are said to be equivalent, in the topological sense, if there is a homeomorphism between them. Topologies also permit defining limits of sequences. But that is really all that they can be used for: talking about limits and continuity. For readers wanting to refresh their topological notions or wanting more details, we suggest looking at (Munkres, 2000).

In our case however, we are interested in doing optimization, and methods such as steepest descent on $\mathbb{R}^n$ require access to both derivatives, as well as to an inner product. We do not have either of these on general topological spaces, and so we will need to add some further structure to the underlying space X. The only problem is that inner products can only be defined on vector spaces, and not on general topological spaces. Indeed, inner products are bi-**linear** functions, that is, linear in both of their arguments. But the whole point of manifold optimization is to generalize vector spaces, and do optimization without the linear structure of $\mathbb{R}^n$! The solution to this conundrum will be to define the inner product not on the space X itself, but on a related space, called the tangent space to X. This added structure will also permit taking derivatives.

Formally, an n-dimensional manifold is a 3-tuple structure $(\mathcal{M}, \mathcal{O}, \mathcal{A})$, where $(\mathcal{M}, \mathcal{O})$ form a topological space, and $\mathcal{A}$ is a set of maps, $\phi_U : U \to \mathbb{R}^n$, one for every open set $U \in \mathcal{O}$, which maps this open set to $\mathbb{R}^n$ homeomorphically. Here, it is implicit that $\mathbb{R}^n$ is a topological space with the standard topology induced by the dot product. A is called an atlas, and its maps ϕ_U are called (coordinate) charts, because we can use them to describe points on the manifold using coordinates, just like we do in vector spaces. The only difference is that on a manifold, these coordinates are local only, whereas they are global in vector spaces. Figure 4 presents two open sets with their charts. At this point, we notice that points p of a Manifold can be part of more than one chart, each of which maps p to different coordinates in $\mathbb{R}^n$. We can ask how these different representations are related. For every two charts ϕ_U and ϕ_V whose domains intersect $U \cap V \neq \emptyset$, the map $\phi_U \circ \phi_V^{-1} : \phi_V(U \cap V) \to \phi_U(U \cap V)$ that relate the two charts are called transition maps. Figure 4 depicts two transition maps, labeled $\tau_{\alpha,\beta}$ and $\tau_{\beta,\alpha}$. The properties of these transition maps give structure to the manifold. For instance, smooth manifolds are manifolds such that these transition maps are infinitely differentiable. We will explain why this is important, keeping our optimization goal in mind. Readers interested in a more gradual and comprehensive treatment of manifolds should consult a proper reference, such as (Loring, 2008).

We remind ourselves that we have a cost function $f : M \to \mathbb{R}$ defined on the manifold, and standing at a point p on the manifold, want to move in the direction in which f locally increases the most. But "locally increases the most" just means "whose local derivative is greatest". So we need to be able to take directional derivatives on the manifold. But we saw in the previous section that this was not permitted on manifolds, because we cannot add or subtract points of the manifold.

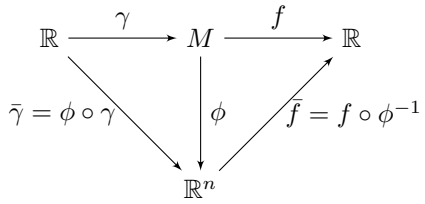

Figure 2. The structure around a manifold: curves, functions, and charts.

In order to resolve this issue, we will need to define curves (also called paths) on the Manifold. A curve on M is a function $\gamma : [a, b] \subseteq \mathbb{R} \to M$ which maps a closed interval of $\mathbb{R}$ into M. An example curve $\gamma(t)$ is depicted in Figure 5. A curve not only traces a path on M, but also permits differentiating functions on M. If $\gamma(0) = p$, then we can define

$$\frac{\partial f}{\partial \gamma}(p) := \frac{\mathrm{d} f \circ \gamma}{\mathrm{d}t}(0)$$

This is well defined because the function $f \circ \gamma$ is a function from $\mathbb{R}$ to $\mathbb{R}$. It is depicted in the top part of diagram in Figure 2. We call $\frac{\partial f}{\partial \gamma}(p)$ the directional derivative of f at p in the direction of γ. We note that many different curves will give the same directional derivative. This is because $\frac{\partial f}{\partial \gamma}(p)$ only considers how γ behaves infinitesimally close to p. We

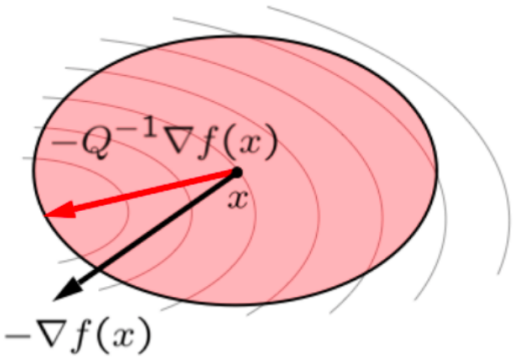

Figure 3. Steepest descent direction in the norm $||x||^2 = x^T Q x$. (Numerical Optimization slides)

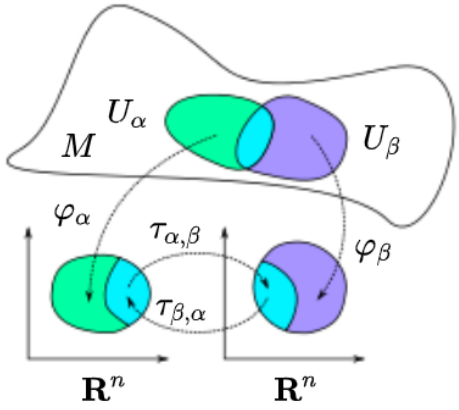

Figure 4. Two charts on a manifold, and their respective transition maps. (Wikipedia: Atlas (topology))

might want to follow our Euclidean intuition, and follow "straight" curves, as the γ in Figure 5 seems to be. However, defining "straightness" on manifolds is really non-trivial, and requires extra structure which we will only introduce later (see section 4.2).

At this point, in order to find the direction of steepest descent, we would need to calculate $\frac{\partial f}{\partial \gamma}(p)$ for curves going in every single direction. Once we found the steepest direction, we could move along it for a short distance, and then iterate. This would work, except that we would be left needing to answer the question "how many directions should I check before moving?" In $\mathbb{R}^n$, the gradient answers this question beautifully, in that we only need to compute n different partial derivatives in order to find the direction of steepest descent! But this requires a linear structure. Fortunately, because derivatives and gradients are a local phenomena, it will turn out that we can "linearize" the manifold around each point and profit from the exact same benefits.

We first need to define a little bit of terminology. For any function $f : M \to \mathbb{R}$, point $p \in M$, and local chart (U, ϕ_U) around p, we call f smooth (or infinitely differentiable) at p if the function $\bar{f}_U = f \circ \phi_U^{-1} : \mathbb{R}^n \to \mathbb{R}$ is smooth (or infinitely differentiable) at $\phi_U(p)$ (see Figure 2). This notion is well defined, in that it doesn't depend on the choice of local chart U, because of the earlier restriction that we made to smooth manifolds, that is manifolds whose transition maps $\phi_U \circ \phi_V^{-1}$ are smooth (remember Figure 4). That is, if $\bar{f}_U$ is smooth in one chart (U, ϕ_U), then so is $\bar{f}_V$ for all other charts (V, ϕ_V) around p. This is because $\bar{f}_V = f \circ \phi_V^{-1} = f \circ (\phi_U^{-1} \circ \phi_U) \circ \phi_V^{-1} = (f \circ \phi_U^{-1}) \circ (\phi_U \circ \phi_V^{-1})$. Hence, $\bar{f}_V$, being the composition of two smooth functions, is also smooth. We denote by C^∞ all functions $f : M \to \mathbb{R}$ which are smooth around every point on the manifold.

We define a tangent vector at p to be an operator $v_p : C^\infty(M) \to \mathbb{R}$ that takes functions and returns a real number and such that there is a curve γ with $\gamma(0) = p$ that realizes this operator, that is

$$v_p f := \dot{\gamma}(0) f := \frac{\partial f}{\partial \gamma}(p) = \frac{\mathrm{d} f \circ \gamma}{\mathrm{d} t}(0)$$

Note that for a given tangent vector v, many different curves γ will realize it. Indeed, tangent vectors are really equivalence classes of curves. The tangent space $T_p M$ to M at p is then defined as all tangent vectors v_p at p. We can turn tangent spaces into vector spaces, by defining

$$(a\dot{\gamma_1}(0) + b\dot{\gamma_2}(0)) f := a(\dot{\gamma_1}(0) f) + b(\dot{\gamma_2}(0) f)$$

To show that $T_p M$ is a vector space, we just need to show that there is a curve γ for which $\dot{\gamma} = (a\dot{\gamma_1}(0) + b\dot{\gamma_2}(0))$. This is done by taking a local chart (U, ϕ), pushing both of γ_1 and γ_2 into $\mathbb{R}^n$, adding them there, and then pulling them back: $\gamma(t) = \phi^{-1}(a\phi(\gamma_1(t)) + b\phi(\gamma_2(t)))$.

Now that we have defined the tangent spaces where most of the optimization action will be happening, we take a small step back to remind ourselves of our objective, which is to calculate the steepest descent direction. In inner product Euclidean spaces, the steepest descent direction d is found by

$$d = \underset{d:||d||=1}{\operatorname{argmin}} \langle \nabla f(x), d \rangle$$

The steepest descent, in norms other than L_2, will not be the gradient. See Figure 3 for a graphical depiction. So in order to find the steepest descent direction on tangent spaces, we will need access to an inner product, and we will also need to define $\nabla f(x)$. We start with the latter because we already have all of the necessary ingredients. Given a local chart (U, ϕ), we define $\nabla f(x) = (\frac{\partial \bar{f}}{\partial x^1}(x), \ldots, \frac{\partial \bar{f}}{\partial x^n}(x))$. Notice that the partials are with respect to $\bar{f} = f \circ \phi^{-1}$! It might seem like a problem that the gradient is defined with respect

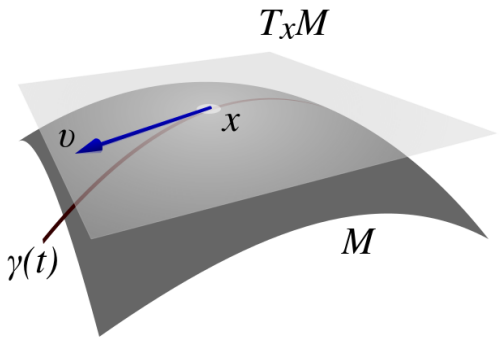

Figure 5. A Manifold M and its tangent plane T_xM at x. We also see a curve $\gamma(t) \in M$, with its associated tangent vector $v \in T_xM$. (Wikipedia: Tangent space)

to an arbitrary chart, as it will clearly be different in different charts, but we will be saved by the inner product which will also be different in different charts, and counteract the gradient.

The last missing structure that we need on the manifold is a Riemannian metric. A Riemannian metric[2] $g : M \times \cup_x T_xM \times \cup_x T_xM \to \mathbb{R}$ is an inner product defined on every tangent space. Given two vectors $v_x, w_x \in T_xM$, their inner product is denoted $g_x(v_x, w_x)$. Given a manifold, we can define g purely intrinsically, but in many cases of interest for optimization, g will be induced from the ambient embedding space. For an example, see section 3.

In order to understand Algorithm 1, we are now only missing one ingredient, the Retraction. As we mentioned before, a retraction $R : \cup_x T_xM \to M$ is a map from tangent spaces to the manifold. In particular, given a vector $v_x \in T_xM$, we write $R_x(v_x)$ to refer to the point where v_x is sent by the retraction. As an example, in Figure 5, $R_x(v)$ might send to one point along the curve γ. Following geodesics (so called "straight lines" on manifolds, see section 4.2) is probably the most famous kind of retraction. However, it is computationally expensive as it requires solving a nonlinear ordinary differential equation. Fortunately, the majority of interesting manifold optimization problems lie in matrix spaces, in which effective projections are often possible. Again, we leave an example of this to section 3.

3. The Sphere as a Basic Example

We will now look in more details at the eigenvalue example from source code 1, and in particular the Riemannian Gradient Descent method from Algorithm 1. We can rewrite that algorithm more compactly as

$$x_{k+1} = \text{Retraction}_{x_k}(-\alpha_k \text{grad} f(x_k))$$

Hence, all that we need to do is to define the gradient and the retraction on the sphere. It turns out the easiest way to define the Riemannian metric on the sphere is actually to view it as an embedded manifold of $\mathbb{R}^n$, and to make use of the ambient Euclidean metric.

In this regard, the Riemannian gradient on the tangent plane turns out to simply be the orthogonal projection of the classical gradient:

$$\text{grad} f(x) = (I - xx^\top)\nabla f(x)$$

As for the retraction, a simple projection works well:

$$\text{Retraction}_x(u) = \frac{x+u}{||x+u||}$$

Finally, we note that the eigenvalue example from source code 1 used the cost function $f(x) = -x^T Ax$, but nothing of what we just wrote depends on this fact. Indeed, the above Riemannian Gradient Descent algorithm on the Sphere would work for any cost function. Table 1 summarizes all of the details for the Sphere.

4. Convergence guarantees

Apart from its beautiful and insightful geometric interpretation, manifold optimization also comes with local convergence guarantees to first-order critical points of f. The first-order necessary optimality conditions are simply $\text{grad} f(x) = 0$, which is much less work than having to go through Lagrange multipliers to establish KKT conditions. Readers interested in learning more about this are invited to read section 4 of (Absil et al., 2008).

Yet, manifolds of interest are typically non convex, so that very little is known when it comes to computing global optimizers. Given the nature of this course, called Structured Prediction And Convex Optimization, we thought it would be worthwhile do to a bit of work in trying not to completely give up convex-like guarantees. A sneak-peek at Figure 6 will explain what we mean. Although we lost convexity, it will be possible to define a new notion, that of geodesic convexity, which will have all the same guarantees but on manifolds. Properly explaining geodesic convexity however requires an even broader coverage of differential geometry, and we will hence only be able to dip our toes into the subject. We start with a quick review of convexity and convex optimization, and then follow-up with geodesic convexity.

[2] Note that it is called a "metric" for historical reasons, but it isn't a metric!

	Manifold (S^{n-1})	Embedding ($\mathbb{R}^n$)
cost	$f(x) = x^T Ax,\ x \in S^{n-1}$	$\bar{f}(x) = x^T Ax,\ x \in \mathbb{R}^n$
metric	induced metric	$\bar{g}(v, w) = v^T w$
tangent space	$v \in \mathbb{R}^n : x^T v = 0$	$\mathbb{R}^n$
normal space	$v \in \mathbb{R}^n : v = \alpha x$	$\emptyset$
projection onto tangent space	$P_x v = (I - xx^T)v$	identity
gradient	grad $f(x) = P_x \text{grad} \bar{f}(x)$	grad $\bar{f}(x) = 2Ax$
retraction	$R_x(v) = \frac{x+v}{\|\|x+v\|\|}$	$R_x(v) = x + v$

Table 1. Optimization on the unit sphere. Table reproduced from (Absil et al., 2008).

4.1. Convexity and Convex Optimization

In standard form, a convex optimization problem is written as

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0 \quad i = 1, \ldots, p \\ & h_j(x) = 0 \quad j = 1, \ldots, q \end{aligned} \tag{1}$$

where f and g_i are convex, and h_j are affine. Following Table 2, we will break down the concepts underlying this formulation.

First, we need a space in which to perform optimization. Here, we have $x \in \mathbb{R}^n$. In section 2, we will generalize this space to smooth manifolds.

Then, we need to be able to be able to define functions on this space, and to take their derivative[3].

With this in hand, we can now define straight lines, which are curves $\gamma(t) : \mathbb{R} \to \mathbb{R}^n$ whose derivative $\gamma'(t)$ is constant, or in other words whose second derivative is the zero vector: $\gamma''(t) = 0$. We often simply define the line segment between p and q as being $\gamma_{qp}(t) = q + t(p - q)$, for $t \in [0, 1]$. Notice that this agrees with the above definition since $\gamma''_{qp}(t) = 0$. This new definition however will permit generalization to the manifold setting.

Convex sets are then defined as sets for which straight lines between any two points in the set, are entirely contained inside the set.

Convex functions can similarly be defined as those whose epigraph is a convex set, but this definition won't generalize as easily. We instead prefer the classical definition: a function f is convex if for any two points a and b in its domain, the straight line between $f(a)$ and $f(b)$ never dips below f, or equivalently, $\gamma_{f(a)f(b)}(t) = f(a) + t(f(b) - f(a)) \geq f(a + t(b - a)) = f(\gamma_{ab}(t))$ for $t \in [0, 1]$.

Convex functions have a single global minimum. This can be argued as follows. The first order definition of convexity (which can be proved starting from the 0th order definition above), is that for all x, y, $f(x) + \nabla f(x)(y - x) \leq f(y)$. If two points are local minima with different values, say $x*$ and $y*$ such that $f(x*) \geq f(y*)$. Then by definition of being a local minimum, we have $\nabla f(x*) = 0$, and so $f(x*) \leq f(y*)$. Exchanging roles of $x*$ and $y*$ shows that $f(x*) = f(y*)$. This shows that all global minima must have the same value. Furthermore, the constraints in problem 1 are convex functions, and the pre-image of convex sets of convex functions is convex. Hence, the feasible region is a convex set, so the every local minimum is a global minimum.

4.2. Geodesic Convexity

Following Figure 6, just like manifold optimization should be thought of as a subset of constrained optimization, geodesic convexity should be thought of as extending the now ubiquitous concept of convexity, which only applies in Euclidean spaces, to the larger domain of Riemannian manifolds. Not only does it promise to extend convex optimization's outreach to a whole new set of problems, but also to rethink established problems, such as Gaussian mixture models, and in so doing sometimes outperform prevailing methods (Hosseini & Sra, 2019).

Inside manifold optimization, we find geodesically convex optimization, which itself contains convex optimization. Convex optimization is already well established and recognized, it's importance often summarized by R. Tyrrell Rockafellar's enlightening quote: "The great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity."[4] Indeed, convex optimization is a very rich class of problems that maintain the crucial property that local minima are global minima, while still being almost as fast as linear programming on modern hard-

[3] We will only concentrate on twice differentiable functions, though subgradient methods have also been generalized to manifolds (Ferreira & Oliveira, 1998)

[4] SIAM Review, 1993.

Convexity	Geodesic Convexity
Euclidean Space	Riemannian Manifold
Derivative	Covariant Derivative
Straight Line	Geodesic
Convex Set	Geodesically Convex Set
Convex Function	Geodesically Convex Function
Local = Global	Local = Global
Algorithms	Riemannian Algorithms

Table 2. Making explicit the often implicitly assumed concepts underlying convexity, and generalizing them to Riemannian Manifolds. The dependency structure amongst the concepts can be viewed as an inverted stack: a concept depends on all concepts above it.

ware. Geodesically convex optimization is its younger, not yet established cousin, promising to expand the class of problems that we can tackle while maintaining the "local is global" property.

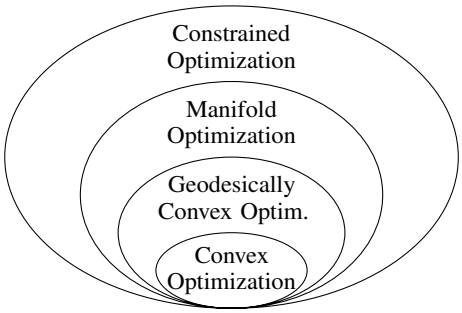

Figure 6. Hierarchy of optimization concepts referred to in this report.

A comparison of convexity and geodesic convexity is presented in Table 2, which is largely inspired from a talk given by Nisheeth Vishnoi at the Institute for Advanced Studies (Vishnoi, 2018a) and his related notes (Vishnoi, 2018b). The "local is global" property of these two classes of problems is included in the second to last row in the table. In Euclidean spaces, the convexity assumption, necessary for this property to hold, is often tersely written as $\nabla^2 f \succeq 0$. This terseness, however, hides a lot of underlying, often implicitly defined, concepts, such as "straight line". In order to generalize the concept to that of geodesic convexity, we need to make explicit every one of these implicit concepts, which this table does.

The first row is the generalization from Euclidean spaces to Riemannian manifolds that we have spent the majority of this report explaining. Explaining the second and third row properly would require another report of equal length, and we refer interested readers to consult chapter 5 of (Absil et al., 2008). The covariant derivative ∇[5], also called affine connection, gives us the ability to take directional derivatives not only of functions $f : M \to \mathbb{R}$ on M, but also of vector fields $\mathbb{X} : M \to \cup_x T_x M$, that is functions which associate to every point $p \in M$ a vector of the tangent space at x. This is much harder to do because it requires comparing vectors from different tangent spaces that aren't apriori related.

With this defined, geodesics, so called "straight lines" on manifolds, are then defined as curves γ for which $\nabla_{\dot\gamma(t)}\dot\gamma(t) = 0$ for all t. Intuitively, this says that geodesics are curves whose velocity ($\dot\gamma$) doesn't change, similarly to how we defined straight lines in $\mathbb{R}^n$ to be those whose second derivative $\gamma''(t) = 0$! The classical example of a geodesic is the equator on a sphere. Note however, that there can be more than one geodesic between two points. For example, there are infinitely many geodesics between the north and south pole.

With these concepts in hand, generalizing convexity to manifolds is almost automatic. A geodesically convex set is a set $A \subseteq M$ such that given two points $p, q \in A$, there exists a geodesic connecting p and q which entirely lies inside A. A geodesically convex function is a function such that for any two points a and b in its domain, the geodesic between $f(a)$ and $f(b)$ never dips below f. And that's it. Geodesic convexity has made some small breakthroughs (Vishnoi, 2018b), but its real power remains to be shown, as there still remain a lot of open questions related to it.

5. Conclusion

In conclusion, Manifold optimization is an alternative to constrained optimization. It's theory is now mature, and there is standard open-source software that makes it accessible to nonspecialists. Yet, it remains a fruitful ground for further research, and further comparisons to standard optimization algorithms.

Acknowledgements

We would like to thank Simon Lacoste-Julien for teaching this amazing class, and for giving us the freedom to work on a project not directly related to the class material.

References

Absil, P.-A. and Malick, J. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22: 135–158, 2012.

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*. Princeton University

[5]The covariant derivative uses the same symbol as the gradient because it is a generalization of the gradient.

Press, Princeton, NJ, 2008. ISBN 978-0-691-13298-3.

Boumal, N., Mishra, B., Absil, P.-A., and Sepulchre, R. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014. URL http://www.manopt.org.

Edelman, A., Arias, T. A., and Smith, S. T. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

Ferreira, O. P. and Oliveira, P. R. Subgradient algorithm on riemannian manifolds. *Journal of Optimization Theory and Applications*, 97(1):93–104, 1998. doi: 10.1023/A:1022675100677. URL https://doi.org/10.1023/A:1022675100677.

Ghojogh, B., Karray, F., and Crowley, M. Eigenvalue and generalized eigenvalue problems: Tutorial. *arXiv preprint arXiv:1903.11240*, 2019.

Hosseini, R. and Sra, S. An alternative to em for gaussian mixture models: Batch and stochastic riemannian optimization. *Mathematical Programming*, pp. 1–37, 2019.

Loring, W. T. An introduction to manifolds, 2008.

Munkres, J. R. Topology second edition prentice hall. 2000.

Renze, J., Rowland, T., and Weisstein, E. W. Compact manifold. From MathWorld–A Wolfram Web Resource. URL https://mathworld.wolfram.com/CompactManifold.html. Accessed: 2020-04-27.

Townsend, J., Koep, N., and Weichwald, S. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016. URL http://jmlr.org/papers/v17/16-177.html.

Vishnoi, N. An introduction to geodesic convexity. https://video.ias.edu/OCIT/2018/0607-NisheethVishnoi, 2018a. Accessed: 2020-04-27.

Vishnoi, N. K. Geodesic convex optimization: Differentiation on manifolds, geodesics, and convexity. *arXiv preprint arXiv:1806.06373*, 2018b.