Manopt

A Matlab toolbox to make optimization on manifolds feel as simple as unconstrained optimization

A project of the RANSO group
Nicolas Boumal and Bamdev Mishra
P.-A. Absil, Y. Nesterov and R. Sepulchre
What is the minimal framework you need for steepest descent optimization?
To optimize, we only need the search space to be a Riemannian manifold

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

We need...

- A notion of directions along which we can move tangent space, tangent vector
- A notion of steepest descent direction inner product, gradient
- A means of moving along a direction Geodesics, retractions
The theory is mature at this point.

What’s been missing is matching software.
Manopt

A Matlab toolbox to make optimization on manifolds feel as simple as unconstrained optimization

With generic solvers,
a library of manifolds and diagnostics tools
Low-rank matrix completion

\[
\min_{X \in M} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2
\]

\[
M = \{X \in \mathbb{R}^{m \times n} : \text{rank}(X) = r\}
\]

Find a matrix \(X\) of rank \(r\) which matches \(A\) as well as possible on a subset of entries.
Independent component analysis

$$\min_{X \in M} \sum_{i=1,\ldots,N} \|\text{offdiag}(X^T C_i X)\|^2$$

$$M = \{X \in \mathbb{R}^{n \times p} : \text{ddiag}(X^T X) = I_p\}$$

Find a demixing matrix $X$ with unit-norm columns which simultaneously diagonalizes given $C_i$’s as well as possible.
Distance matrix completion

\[
\min_{X \in M} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2
\]

\[
M = \{X \in \mathbb{R}^{n \times n} : q_1, \ldots, q_n \in \mathbb{R}^p \text{ and } X_{ij} = \|q_i - q_j\|^2\}
\]

Find a Euclidean distance matrix \(X\) which matches \(A\) as well as possible on a subset of entries.
Estimation of rotations

\[
\min_{R_1, \ldots, R_N \in M} \sum_{(i,j) \in \Omega} \left\| R_i R_j^T - H_{ij} \right\|^2
\]

\[M = \{ Q \in \mathbb{R}^{n \times n} : Q^T Q = I \text{ and } \det(Q) = +1 \}\]

Find rotation matrices \( R_i \) which match measurements of relative rotations \( H_{ij} \approx R_i R_j^T \) as well as possible.
Example code for dominant eigenvectors

$$\max_x \frac{x^T A x}{x^T x}$$
Example code for dominant eigenvectors

\[
\begin{align*}
\max_{\|x\|=1} & \quad x^T A x \\
M &= \{x \in \mathbb{R}^n : x^T x = 1\} \\
f(x) &= x^T A x \\
\nabla f(x) &= 2A x \\
\nabla f(x) &= (I - xx^T)\nabla f(x)
\end{align*}
\]
import manopt.solvers.trustregions.*;
import manopt.manifolds.sphere.*;
import manopt.tools.*;

% Generate the problem data.
n = 1000;
A = randn(n);
A = .5*(A+A');

% Create the problem structure.
manifold = spherefactory(n);
problem.M = manifold;

% Define the problem cost function and its gradient.
problem.cost = @(x) -x'*(A*x);
problem.grad = @(x) manifold.egrad2rgrad(x, -2*A*x);

% Numerically check gradient consistency.
checkgradient(problem);
Gradient check

Approximation error

Step size in the Taylor expansion
% Generate the problem data.
n = 1000;
A = randn(n);
A = .5*(A+A');

% Create the problem structure.
manifold = spherefactory(n);
problem.M = manifold;

% Define the problem cost function and its gradient.
problem.cost = @(x) -x'*(A*x);
problem.grad = @(x) manifold.egrad2rgrad(x, -2*A*x);

% Numerically check gradient consistency.
checkgradient(problem);

% Solve.
[x xcost info] = trustregions(problem);
## Optimization Results

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Function Value</th>
<th>Gradient Norm</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>k: 1</td>
<td>f: 1.571531e+000</td>
<td>grad: 4.456216e+001</td>
<td>negative curvature</td>
</tr>
<tr>
<td>k: 2</td>
<td>f: -2.147351e+001</td>
<td>grad: 3.053440e+001</td>
<td>negative curvature</td>
</tr>
<tr>
<td>k: 3</td>
<td>f: -3.066561e+001</td>
<td>grad: 3.142679e+001</td>
<td>negative curvature</td>
</tr>
<tr>
<td>k: 4</td>
<td>f: -3.683374e+001</td>
<td>grad: 2.125506e+001</td>
<td>exceeded trust region</td>
</tr>
<tr>
<td>k: 5</td>
<td>f: -4.007868e+001</td>
<td>grad: 1.389614e+001</td>
<td>exceeded trust region</td>
</tr>
<tr>
<td>k: 6</td>
<td>f: -4.237276e+001</td>
<td>grad: 9.687523e+000</td>
<td>exceeded trust region</td>
</tr>
<tr>
<td>k: 7</td>
<td>f: -4.356244e+001</td>
<td>grad: 5.142297e+000</td>
<td>exceeded trust region</td>
</tr>
<tr>
<td>k: 8</td>
<td>f: -4.412433e+001</td>
<td>grad: 2.860465e+000</td>
<td>exceeded trust region</td>
</tr>
<tr>
<td>k: 9</td>
<td>f: -4.438540e+001</td>
<td>grad: 3.893763e-001</td>
<td>reached target residual-kappa</td>
</tr>
<tr>
<td>k: 10</td>
<td>f: -4.442759e+001</td>
<td>grad: 4.116374e-002</td>
<td>reached target residual-kappa</td>
</tr>
<tr>
<td>k: 11</td>
<td>f: -4.442790e+001</td>
<td>grad: 1.443240e-003</td>
<td>reached target residual-theta</td>
</tr>
<tr>
<td>k: 12</td>
<td>f: -4.442790e+001</td>
<td>grad: 1.790137e-006</td>
<td>reached target residual-theta</td>
</tr>
<tr>
<td>k: 13</td>
<td>f: -4.442790e+001</td>
<td>grad: 3.992606e-010</td>
<td>dimension exceeded</td>
</tr>
</tbody>
</table>

Gradient norm tolerance reached.
Total time is 2.966843 [s] (excludes statsfun)
import manopt.solvers.trustregions.*;
import manopt.manifolds.sphere.*;
import manopt.tools.*;

% Generate the problem data.
n = 1000;
A = randn(n);
A = .5*(A+A');

% Create the problem structure.
manifold = spherefactory(n);
problem.M = manifold;

% Define the problem cost function and its gradient.
problem.cost = @(x) -x'*(A*x);
problem.grad = @(x) manifold.egrad2rgrad(x, -2*A*x);

% Numerically check gradient consistency.
checkgradient(problem);

% Solve.
[x xcost info] = trustregions(problem);

% Display some statistics.
semilogy([info.iter], [info.gradnorm], '.-');
Convergence of the trust-regions method

Gradient norm

Iteration #
Riemannian optimization is...

Well-understood

Theory is available for many algorithms

Useful

We covered a few fashionable problems

Easy

With Manopt, you simply provide the cost
Manopt is open source and documented

www.manopt.org

Welcome to Manopt!
A Matlab toolbox for optimization on manifolds

Optimization on manifolds is a powerful paradigm to address nonlinear optimization problems. With Manopt, it is easy to deal with various types of constraints that arise naturally in applications, such as orthonormality or low rank.

Download Get started

Maniwhat now?
Manifolds are mathematical sets with a smooth geometry, such as spheres. If you are facing a nonlinear (and possibly nonconvex) optimization problem with nice-looking constraints or invariance properties, Manopt may just be the tool for you. Check out the manifolds library to find out!

Key features
Manopt comes with a large library of manifolds and ready-to-use Riemannian optimization algorithms. It is well documented and includes diagnostics tools to help you get started quickly and make sure you make no mistakes along the way. It is designed to provide great flexibility in describing your cost function and incorporates an optional caching system for more efficiency.

May I use it?
Sure! This is open-source software. Check out the license and let us know how you use Manopt.