Spherical Harmonics Fitting

Kaibo Tang

July 3, 2024

1 Introduction

Spherical harmonics (SHs) comes in handy when we deal with complex-valued functions defined on the unit sphere. Sometimes we might wish to fit SHs up to a certain order to a spherical function, usually for the following reasons among many others:

1. We wish to reduce the dimension of the data.

2. We are looking for a smooth representation of the discretely sampled (and potentially noisy) data points defined on the unit sphere.

In addition, when the underlying data distribution on the unit sphere satisfies antipodal symmetry, we might opt to use only the SH of even orders to ensure the reconstructed signal also follows antipodal symmetry. For scenarios where the observed data on the unit sphere is very noisy, we might want to have a regularization factor λ which we can tune during SHs fitting to ensure we don't overfit on the noise.

In this note, I will discuss both theoretically and practically how we can perform SHs fitting. The theoretical part of this note bases heavily on the original paper by Maxime Descoteaux who proposed this method.[1] The practical part of this note is based on Spherical-Harmonics-Fitting,[2] a MATLAB toolbox developed by myself. The toolbox is a re-implementation of selected tools from DIPY.[3]

[1] 10.1002/mrm.21277
[2] mathworks.com/matlabcentral/fileexchange/168591-spherical-harmonics-fitting
[3] github.com/dipy/dipy

2 Theory

We assume the readers have familiarity with SHs and will skip the definitions of SHs. For a brief review, see DIPY's explanation.[4]

2.1 Univariate case

Let $f : S^2 \to \mathbb{F}$ be the spherical function of interest. For the rest of the note, we take $\mathbb{F} = \mathbb{R}$ but the same method applies to complex-valued spherical functions as well. Suppose we sample f N-times, and let $(\theta_i, \phi_i)_{i=1}^N$ denote the locations where the samples are taken, where θ_i denote the polar angle and ϕ_i denote the azimuthal angle of the i-th sample. Let $R = (l+1)(l+2)/2$ denote the number of SH coefficients up to a max order of l.

In addition, we define the following notations.

- Let $\mathbf{B} \in \mathcal{M}_{N\times R}$ be given by
$$\mathbf{B} = \begin{pmatrix} Y_1(\theta_1, \phi_1) & Y_2(\theta_1, \phi_1) & \cdots & Y_R(\theta_1, \phi_1) \\ \vdots & \vdots & \ddots & \vdots \\ Y_1(\theta_N, \phi_N) & Y_2(\theta_N, \phi_N) & \cdots & Y_R(\theta_N, \phi_N) \end{pmatrix}, \tag{1}$$
where Y_r denote the r-th SH basis function.
- Let the column vector $\mathbf{c} \in \mathbb{R}^R$ denote the SH coefficients.
- Let the column vector $\mathbf{s} \in \mathbb{R}^N$ denote the observed data at the sampled locations.

Without any regularization, a naive implementation of SH fitting would come in the form of
$$\min_{\mathbf{c}} \|\mathbf{Bc} - \mathbf{s}\|_2^2 . \tag{2}$$
And an naive implementation of a regularized version of this problem would be using Tikhonov regularization, i.e.,
$$\min_{\mathbf{c}} \|\mathbf{Bc} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{c}\|_2^2 . \tag{3}$$
However, Tikhonov regularization assigns penalty weight to lower and higher order coefficients equally. For the sake of preventing overfitting and recalling the famous statistical principle of parsimony, we might want to assign

[4] workshop.dipy.org/documentation/1.5.0/theory/sh_basis/

more penalty on higher order coefficients. In light of this observation, for a reconstructed spherical function $\hat{f} : S^2 \to \mathbb{R}$ given by

$$\hat{f} = \bar{\mathbf{B}}\mathbf{c}, \text{ where } \bar{\mathbf{B}} = \begin{pmatrix} Y_1 & Y_2 & \cdots & Y_R \end{pmatrix}, \tag{4}$$

Descoteaux et al., 2007 propose the following regularization:

$$E(\hat{f}) = \int_\Omega (\Delta \hat{f})^2 d\Omega \tag{5}$$

where $\int_\Omega$ denote surface integral over the unit sphere S^2, and Δ is the Laplace-Beltrami operator, which penalize deviation from smoothness of the reconstructed signal $\hat{f}$. Leveraging the fact[5] that the linear operator Δ satisfies

$$\Delta Y_l^m = -l(l+1)Y_l^m, \tag{6}$$

and that the SH basis functions are orthonormal,[6] we have

$$\begin{aligned}
E(\hat{f}) &= \int_\Omega (\Delta(\bar{\mathbf{B}}\mathbf{c}))^2 d\Omega & (7)\\
&= \int_\Omega \left(\sum_{r=1}^{R} c_r \Delta Y_r \right)^2 d\Omega & (8)\\
&= \int_\Omega \left(\sum_{r=1}^{R} c_r(-l_r(l_r+1)Y_r) \right)^2 d\Omega & (9)\\
&= \sum_{r=1}^{R} l_r^2 (l_r+1)^2 c_r^2 \int_\Omega |Y_r|^2 d\Omega & (10)\\
&= \sum_{r=1}^{R} l_r^2 (l_r+1)^2 c_r^2 & (11)\\
&= \|\mathbf{Lc}\|_2^2, & (12)
\end{aligned}$$

where $\mathbf{L} = \text{diag}(l_1(l_1+1), \cdots, l_R(l_R+1))$. Note that we apply (6) to get (9) from (8). To get from (9) to (10), observe that the "mixed" terms vanish due to orthogonality. To get from (10) to (11), recall the normality of Y_r.

Note that in this formulation, the weight for higher order coefficients (i.e., larger l_r) is also larger. As a result, compared to Tikhonov regularization,

[5] This is a well-known fact about SH but I don't have a reference on top of my head.

[6] This is another fact about SH that I don't have a reference for.

the regularizer here punishes higher order coefficients more than it does for lower order coefficients. Hence, the SH fitting problem with regularization now is given by

$$\min_{\mathbf{c}} \|\mathbf{Bc} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{Lc}\|_2^2 , \tag{13}$$

which is equivalent to solving the following equation in block matrix notation:

$$\begin{pmatrix} \mathbf{B} \\ \sqrt{\lambda}\mathbf{L} \end{pmatrix} \mathbf{c} = \begin{pmatrix} \mathbf{s} \\ 0 \end{pmatrix} . \tag{14}$$

2.2 Multivariate case

Solving this linear problem for a scalar-valued spherical function might not pose a significant computational barrier. However, for vector-, matrix-, or tensor-valued spherical functions, solving (14) individually for each $\mathbf{s}$ becomes intractable. Hence, similar to the implementation in DIPY, we solve the problem using Moore–Penrose pseudoinverse, i.e,

$$\hat{\mathbf{c}} = \begin{pmatrix} \mathbf{B} \\ \sqrt{\lambda}\mathbf{L} \end{pmatrix}^+ \begin{pmatrix} \mathbf{s} \\ 0 \end{pmatrix} . \tag{15}$$

Note that using pseudoinverse allows us to vectorize the problem. Consider a flattened tensor-valued spherical function $\mathbf{F} : S^2 \to \mathbb{R}^V$. Let

$$\mathbf{S} = \begin{pmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_V \end{pmatrix} \tag{16}$$

denote the observed and flattened data tensors of the tensor-valued spherical functions where each $\mathbf{s}_v$ denotes an observed data of scalar-valued spherical function. And let

$$\hat{\mathbf{C}} = \begin{pmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_V \end{pmatrix} \tag{17}$$

denote the fitted SH coefficients flattened in the same fashion as $\mathbf{S}$ where each $\mathbf{c}_v$ denotes the fitted SH coefficients for $\mathbf{s}_v$. Then, the SH fitting problem with regularization can be solved by

$$\hat{\mathbf{C}} = \begin{pmatrix} \mathbf{B} \\ \sqrt{\lambda}\mathbf{L} \end{pmatrix}^+ \begin{pmatrix} \mathbf{S} \\ 0 \end{pmatrix} .$$

Now, from the SH coefficients, we can then obtain the reconstructed tensor-valued (despite flattened) spherical function $\hat{\mathbf{F}} : S^2 \to \mathbb{R}^V$ by

$$\hat{\mathbf{F}} = \bar{\mathbf{B}}\hat{\mathbf{C}}. \tag{18}$$

3 Method

In this section, we discuss how the above mentioned strategy is implemented in practice. We will demonstrate the consistency between Spherical-Harmonics-Fitting[7] and the original Python implementation of this method.[8]

For case study, we consider the following problem that we might encounter in the realm of diffusion MRI. Specifically, suppose we have a Cartesian grid of shape $H \times W \times L$ in 3D space. At each grid location, there is an underlying orientation distribution function (ODF) that we wish to reconstruct. Let's say we sample N points on the unit sphere S^2. These points can be identified by their polar angles θ and azimuthal angles ϕ, i.e.,

```
theta = ...;  % shape (N,)
phi = ...;  % shape (N,)
```

We make N observations along the directions defined by the N points for each ODF. And the observations can be represented by a tensor:

```
odf = ...;  % shape (H, W, L, N)
```

Now, we can perform SH fitting and obtain the SH coefficients at each grid location. In this case, we fit even order SHs (by default) up to order 8 with no smoothing (by default):

```
sh = sf_to_sh(odf, theta, phi, 8);
```

At this point, we might wish to performance inference using the fitted SH coefficients, i.e., predict the ODFs evaluated along other directions. Let's say we sample M points this time on the unit sphere given by

```
new_theta = ...;  % shape (M,)
new_phi = ...;  % shape (M,)
```

We can then perform inference by calling the following:

```
sf = sh_to_sf(sh, new_theta, new_phi, 8);
```

[7] mathworks.com/matlabcentral/fileexchange/168591-spherical-harmonics-fitting
[8] github.com/dipy/dipy

An interested reader might wish to try the code themselves. See the example script.[9]

For the sake of completeness, a Python script using DIPY that performs the same task is also provided below.

```
from dipy.core.sphere import Sphere
from dipy.reconst.shm import sf_to_sh, sh_to_sf

theta = ...  # shape (N,)
phi = ...  # shape (N,)
sphere = Sphere(theta=theta, phi=phi)

odf = ...  # shape (H, W, L, N)

sh = sf_to_sh(odf, sphere, 8)

new_theta = ...  # shape (M,)
new_phi = ...  # shape (M,)
new_sphere = Sphere(theta=new_theta, phi=new_phi)

sf = sh_to_sf(sh, new_sphere, 8)
```

4 Caveats

There are some clarifications about both the technical and practical aspect of this method that need to be made.

Remark 4.1 (Regarding antipodal symmetry)**.** *When dealing with spherical functions that satisfy antipodal symmetry, instead of fitting the full set of SHs, we might wish to fit using only the SHs of even orders. This is because even order SHs are symmetric (and thus even).*

Remark 4.2 (Regarding θ and ϕ)**.** *Different packages adopt different conventions for polar and azimuthal angles.*

1. *MATLAB and DIPY use θ for polar angle and ϕ for azimuthal angle.*

2. *SciPy uses θ for azimuthal angle and ϕ for polar angle.*

In this package, we follow the first convention.

[9] github.com/kvttt/SphericalHarmonicsFitting/blob/main/test.m

5 Package logo

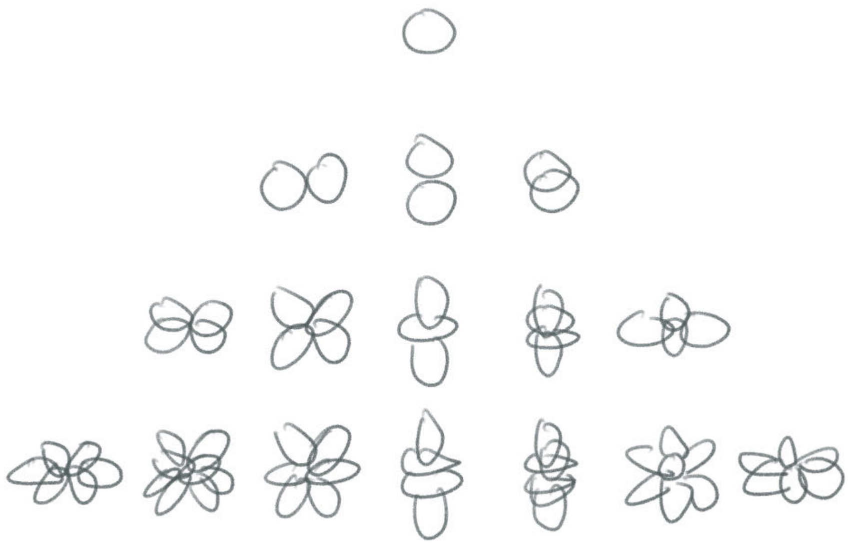

Figure 1: Package logo of Spherical-Harmonics-Fitting showing all the SHs up to a maximum order of 3 (hand-drawn by the author).