Estimating Riemann Curvature Tensor on Point Cloud Data in High Dimension

1 Estimation Step

The algorithm to estimate the Riemann curvature tensor can be divided into three stages:

1. Preprocessing: Construct $\varepsilon-$NN or $K-$NN neighborhood graph for the data point, Decide the intrinsic dimension of the data manifold d and find bases for the tangent space $T_p\mathcal{M}$ at every point p on the manifold using local Principal Component Analysis(PCA).

2. Resampling: Randomly choose two points in the neighborhood and calculate the Gaussian curvature of this section(which is the sectional curvature) and repeat this process for H times.

3. Estimating: Solve an optimization problem to find the values of every component $Rm(X, Y, Z, W)$ of the Riemann curvature tensor.

1.1 Local PCA for tangent space estimation

As tradiational manifold learning method works, the input data for a manifold learning algorithm is a set of point $\mathcal{D} = \{x_i\}_{i=1}^N \subset \mathcal{M} \subset \mathbb{R}^D$. We preprocess the data by first constructing a neighborhood graph $\mathcal{G} = (V, E)$ where V are the vertices and E the edges of $\mathcal{G}$. Here in the graph we define the neighborhood of a point p as the set of points whose distance to p is less than $\sqrt{\varepsilon}$. And we only connect one point with its neighbors in the graph. Note that this graph is symmetric naturally.

Here we follow the standard technique that many tangent based manifold learning adopt to estimate the tangent space at every point p on the manifold such as [1] [2] etc.

Let us denote the number of neighboring number of p by N_p and the neighbors of p to be $x_1, \cdots, x_{N_p}$. We always assume that ε is large enough so that $N_p > d$, which is the instrinsic dimension of the underlying data manifold. But also ε should be small enough that $N_i \ll N$.

Then we combine the neighborhood matrix

$$X_p = [x_{i_1} - p, \cdots, x_{N_p} - p] \tag{1}$$

Or we can re centralize it to the mean point of the neighborhoods. Then we construct a weight matrix D_i, which is an $N_i \times N_i$ diagonal matrix and

$$D_p(j, j) = \sqrt{K\left(\frac{\parallel p - x_j \parallel^2}{\sqrt{\varepsilon}}\right)} \tag{2}$$

where $K(\cdot)$ is a kernel. or weight function. For example we can choose the heat kernel with $K(u) = \exp(-u^2)$. This weight allign more weights to the nearer points to p. Then we multiply neighborhood matrix with weight matrix and get

$$B_p = X_p D_p \tag{3}$$

And we can canduct a SVD decomposition of $B_p = U_p \Sigma_p V_p^T$ and get the singular values of B_p by $\sigma_{p,1} \geq \sigma_{p,2} \geq \cdots$. We can write the columns of $D \times N_p$ matrix U_p as

$$U_p = [\boldsymbol{u}_{p_1}, \boldsymbol{u}_{p_2}, \cdots, \boldsymbol{u}_{p_{N_p}}] \tag{4}$$

And then we take the first d columns and can get our estimation of the base of the tangent space

$$O_p = [\boldsymbol{u}_{p_1}, \boldsymbol{u}_{p_2}, \cdots, \boldsymbol{u}_{p_d}] \tag{5}$$

And [1] have more specific discussion of local PCA for interesting readers.

1.2 Resampling to get sectional curvatures

For two points s, t in the neighborhood of point p, we suppose that γ_s and γ_t are the geodesics connecting p to s and t in the manifold and $\boldsymbol{a}, \boldsymbol{b}$ are the tangent vectors of γ_s and γ_t at point p. Then we can form a 2-manifold $ii(\boldsymbol{a}, \boldsymbol{b})$ as the section. Since γ_s and γ_t curves locally minimizing distance from p to s and t on the whole manifold, they should also be geodesics on $ii(\boldsymbol{a}, \boldsymbol{b})$. Then we imagine an additional geodesic connecting s and t on $ii(\boldsymbol{a}, \boldsymbol{b})$

Now let's turn to the geodesic triangle Δ_{pst} on the 2-manifold. From the Gauss-Bonnet formula, we have the following approximation of the Gaussian curvature of the 2-manifold $ii(\boldsymbol{a}, \boldsymbol{b})$

$$\widehat{K}(\boldsymbol{a}, \boldsymbol{b}) \approx \frac{\alpha_p + \alpha_s + \alpha_t - \pi}{\text{Area}(\Delta_{pst})} \tag{6}$$

at point p. For this apporximation we naively suppose that K is almost constant on the small neighborhood of p on this section plane.

We know that the sectional curvature is actually related to the Riemann curvature tensor by

$$K(\boldsymbol{a}, \boldsymbol{b}) = \frac{Rm(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{b}, \boldsymbol{a})}{|\boldsymbol{a}|^2|\boldsymbol{b}|^2 - \langle \boldsymbol{a}, \boldsymbol{b}\rangle^2} \tag{7}$$

Therefore considering the linear expansion of $\boldsymbol{a}, \boldsymbol{b}$ on the local tangent space

$$\boldsymbol{a} = \sum_{i=1}^{d} a_i \boldsymbol{u}_{pi}, \quad \boldsymbol{b} = \sum_{i=1}^{d} b_i \boldsymbol{u}_{pi} \tag{8}$$

Since it is known that the Riemann curvature tensor is multilinear over $\mathbb{R}$, we have

$$Rm(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{b}, \boldsymbol{a}) = \sum_{i_1=1}^{d}\sum_{i_2=1}^{d}\sum_{i_3=1}^{d}\sum_{i_4=1}^{d} a_{i_1} b_{i_2} b_{i_3} a_{i_4} Rm(\boldsymbol{u}_{i_1}, \boldsymbol{u}_{i_2}, \boldsymbol{u}_{i_3}, \boldsymbol{u}_{i_4}) \tag{9}$$

Therefore with 7 and 9,

$$K(\boldsymbol{a}, \boldsymbol{b}) \left(\sum_{i=1}^{d} a_i^2 \sum_{j=1}^{d} b_j^2 - (\sum_{k=1}^{d} a_k b_k)^2 \right) = \sum_{i_1=1}^{d}\sum_{i_2=1}^{d}\sum_{i_3=1}^{d}\sum_{i_4=1}^{d} a_{i_1} b_{i_2} b_{i_3} a_{i_4} Rm(\boldsymbol{u}_{i_1}, \boldsymbol{u}_{i_2}, \boldsymbol{u}_{i_3}, \boldsymbol{u}_{i_4}) \tag{10}$$

And we substitute our estimation of K there, we have a linear equation for the components $R_{ijkl} = Rm(widehatboldsymbolu_i, widehatboldsymbolu_j, widehatboldsymbolu_k, widehatboldsymbolu_l)$.

Now let's repeat this process for H times. We naively use the projection of $s-p, t-p$ onto the tangent space $T_p\mathcal{M}$ to approximate $\boldsymbol{a}, \boldsymbol{b}$.

$$\widehat{\boldsymbol{a}} = O_p^T(s-p) = \sum_{i=1}^{d} \widehat{a}_i \widehat{\boldsymbol{u}}_i, \quad \widehat{\boldsymbol{b}} = O_p^T(s-p) = \sum_{i=1}^{d} \widehat{b}_i \widehat{\boldsymbol{u}}_i \tag{11}$$

For the $l-$th time we get an equation

$$\widehat{K}(\widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l) \left(\sum_{i=1}^{d} (\widehat{a}_i^l)^2 \sum_{j=1}^{d} (\widehat{b}_j^l)^2 - (\sum_{k=1}^{d} \widehat{a}_k^l \widehat{b}_k^l)^2 \right) = \sum_{i_1=1}^{d}\sum_{i_2=1}^{d}\sum_{i_3=1}^{d}\sum_{i_4=1}^{d} \widehat{a}_{i_1}^l \widehat{b}_{i_2}^l \widehat{b}_{i_3}^l \widehat{a}_{i_4}^l Rm(\widehat{\boldsymbol{u}}_{i_1}, \widehat{\boldsymbol{u}}_{i_2}, \widehat{\boldsymbol{u}}_{i_3}, \widehat{\boldsymbol{u}}_{i_4}) \tag{12}$$

Putting all this equation together we will have a linear system to solve for components of Riemann curvature tensor. And according to the symmetrical properties of the Riemann curvature tensor, we should add the following restrictions which hold for arbitrary combination of (i, j, k, l) in $\{1, 2, \cdots, d\}^4$

$$Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l) = -Rm(\widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l) = -Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_l, \widehat{\boldsymbol{u}}_k) = Rm(\widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l, \widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j) \tag{13}$$

$$Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l) + Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l, \widehat{\boldsymbol{u}}_j) + Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_l, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_k) = 0 \tag{14}$$

1.3 Optimization to solve the linear system

Unfortunately direct estimation and solution to the linear system cannot give a stable result. Therefore we are turning to an optimization problem to solve for our Riemann curvature tensor. Before that we introduce some notations out of simplicity. We use $\otimes$ to denote the kroecker product of matrices and vec to denote an operator which maps $n \times n$ matrix

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

into a n^2 vector

$$\text{vec}(\boldsymbol{A}) = [a_{11}, a_{12}, \cdots, a_{1n}, a_{21}, a_{22}, \cdots, a_{nn}]^T \tag{15}$$

Let $\boldsymbol{R}$ denote a vector representing the Riemann curvature tensor. First for every fixed i, j we use a $d \times d$ matrix $(\boldsymbol{R}_{ij})_{kl}$ to denote the matrix

$$\begin{pmatrix} Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_1, \widehat{\boldsymbol{u}}_1) & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_1, \widehat{\boldsymbol{u}}_2) & \cdots & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_1, \widehat{\boldsymbol{u}}_d) \\ Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_2, \widehat{\boldsymbol{u}}_1) & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_2, \widehat{\boldsymbol{u}}_2) & \cdots & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_2, \widehat{\boldsymbol{u}}_d) \\ \vdots & \vdots & \vdots & \vdots \\ Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_d, \widehat{\boldsymbol{u}}_1) & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_d, \widehat{\boldsymbol{u}}_2) & \cdots & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_d, \widehat{\boldsymbol{u}}_d) \end{pmatrix} \tag{16}$$

Then we put all these $\boldsymbol{R}_{ij}$ together as a $d^2 \times d^2$ block matrix

$$\widetilde{\boldsymbol{Rm}} = \begin{pmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} & \cdots & \boldsymbol{R}_{1d} \\ \boldsymbol{R}_{21} & \boldsymbol{R}_{22} & \cdots & \boldsymbol{R}_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{R}_{d1} & \boldsymbol{R}_{d2} & \cdots & \boldsymbol{R}_{dd} \end{pmatrix} \tag{17}$$

And finally we construct our vector $\boldsymbol{Rm} = \text{vec}(\widetilde{\boldsymbol{Rm}})$. We will use R_{ijkl} to denote the $kl-$ element in the matrix $\boldsymbol{R}_{ij}$, which is $Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l)$

Then rewrite 12 we can get a loss function

$$E_d(l) = \left(\widehat{K}(\widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l) - \frac{(\widehat{\boldsymbol{a}}^l \otimes (\widehat{\boldsymbol{b}}^l \otimes (\widehat{\boldsymbol{b}}^l \otimes \widehat{\boldsymbol{a}}^l)))^T \boldsymbol{Rm}}{|\widehat{\boldsymbol{a}}^l|^2 |\widehat{\boldsymbol{b}}^l|^2 - |\langle \widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l \rangle|^2} \right)^2 \tag{18}$$

Therefore our goal is to solve the linear system by considering the following optimization problem since directly solving for the system may be very unstable numerically.

$$\begin{aligned} \textbf{min} \quad & \text{median}_l E(l) \\ \textbf{s.t.} \quad & R_{ijkl} = -R_{jikl} = -R_{ijlk} = R_{klij} \\ & R_{ijkl} + R_{iklj} + R_{iljk} = 0 \end{aligned} \tag{19}$$

It is not easy enough to solve this constrained optimization problem 19 However, fortunately all the constraints are linear equality constraints. And we are able to only look for the independent components of the Riemann curvature tensor as another vector $\boldsymbol{R}$. Then we can construct a stretching matrix $\boldsymbol{S}$ and multiply it to the left of $\boldsymbol{R}$ to get $\boldsymbol{Rm}$. Here we give an example of how to construct $\boldsymbol{R}$ and $\boldsymbol{S}$.

The independent components in Riemann curvature tensor including several cases:

(a) For every $i \in \{1, 2, \cdots, d\}$, $R_{iiii} = 0$, therefore they are not needed in $\boldsymbol{R}$.

(b) There are two different indexes in (i, j, k, l), the only independent and nonzero component is R_{ijij}. There are $\binom{d}{2} = d(d-1)/2$ of them.

(c) There are three different indexes in (i, j, k, l), we assume that depending on which number (minimum, median, maximum) is the repeated index, there are three independent values. Totally there are $3\binom{d}{3} = d(d-1)(d-2)/2$ of them.

(d) Four indexes are all different, let $\sigma(\cdot)$ representing the ascending order index of $\cdot$ in the four indexes, then we can divide them into three subgroups:

i) $|\sigma(i) - \sigma(j)| = |\sigma(k) - \sigma(l)| = 1$

ii) $|\sigma(i) - \sigma(j)| = |\sigma(k) - \sigma(l)| = 2$

iii) $|\sigma(i) - \sigma(j)| \neq |\sigma(l) - \sigma(k)|$

And when we know two values of R_{ijkl} falling in two of the three cases, we can have all values of components with permutations of $\{i, j, k, l\}$. All the independent components add up to $\frac{2}{3}\frac{d(d-1)(d-2)(d-3)}{8}$

Therefore we have $\frac{1}{12}d^2(d^2-1)$ independent components of Riemann curvature tensor in total. We will use the following notations:

$$\begin{aligned}
f_2(i,j) &= \sum_{i_1=1}^{i-1}(d-i_1) + (j-i) = \frac{1}{2}(2d-i)(i-1) + (j-i) \quad (i<j) \\
f_3(i,j,k) &= \sum_{i_1=1}^{i-1}\sum_{i_2=i_1+1}^{d-1}(d-i_2) + \sum_{i_2=i+1}^{j-1}(d-i_2) + (k-j) \\
&= \frac{1}{6}\left(3d^2i - 3d^2 - 3di^2 + 3d + i^3 - i\right) - \frac{1}{2}(i-j+1)(2d-i-j) - j + k \\
f_4(i,j,k,l) &= \sum_{i_1=1}^{i-1}\sum_{i_2=i_1+1}^{d-2}\sum_{i_3=i_2+1}^{d-1}(d-i_3) + \sum_{i_2=i+1}^{j-1}\sum_{i_3=i_2+1}^{d-1}(d-i_3) + \sum_{i_3=j+1}^{d-1}(d-i_3) + (l-k) \\
&= -\frac{1}{6}(i-j+1)\left(3d^2 - 3di - 3dj - 3d + i^2 + ij + 2i + j^2 + j\right) - \frac{1}{2}(j-k+1)(2d-j-k) - k + l \\
&+ \frac{1}{24}\left(4d^3i - 4d^3 - 6d^2i^2 - 6d^2i + 12d^2 + 4di^3 + 6di^2 - 2di - 8d - i^4 - 2i^3 + i^2 + 2i\right)
\end{aligned}$$

Then we briefly state our construction of $\boldsymbol{R}$ as following (we always suppose $i<j<k<l$):

$$\begin{aligned}
&\boldsymbol{R}_{f_2(i,j)} = R_{ijij} \\
&\boldsymbol{R}_{\binom{d}{2}+3(f_3(i,j,k)-1)+1} = R_{ijki} \\
&\boldsymbol{R}_{\binom{d}{2}+3(f_3(i,j,k)-1)+2} = R_{jikj} \\
&\boldsymbol{R}_{\binom{d}{2}+3(f_3(i,j,k)-1)+3} = R_{kijk} \\
&\boldsymbol{R}_{\binom{d}{2}+3\binom{d}{3}+2(f_4(i,j,k,l)-1)+1} = R_{ijkl} \\
&\boldsymbol{R}_{\binom{d}{2}+3\binom{d}{3}+2(f_4(i,j,k,l)-1)+2} = R_{ikjl}
\end{aligned} \tag{20}$$

And then for each row in the stretching matrix $\boldsymbol{S}$, we assign one or two elements to 1 or -1 according to the symmetric properties and bianchi identities. For example with the case $d=2$ we have only 1 independent component in the riemann curvature tensor, so we have

$$\boldsymbol{R} = R_{1212}, \boldsymbol{S} = [0,0,0,0,0,1,-1,0,0,-1,1,0,0,0,0,0]^T \tag{21}$$

And then the problem 19 can be transfered into a non constraint optimization problem with a newly defined loss function

$$E_d(l) = \left(\widehat{K}(\widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l) - \frac{(\widehat{\boldsymbol{a}}^l \otimes (\widehat{\boldsymbol{b}}^l \otimes (\widehat{\boldsymbol{b}}^l \otimes \widehat{\boldsymbol{a}}^l)))^T \boldsymbol{S}\boldsymbol{R}}{|\widehat{\boldsymbol{a}}^l|^2 |\widehat{\boldsymbol{b}}^l|^2 - |\langle \widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l \rangle|^2}\right)^2 \tag{22}$$

and we want to find the $\boldsymbol{R}$ in $\mathbb{R}^{(d(d-1)/12)}$ that minimize the loss function $E(l)$. Such optimization problems are actually not easy to solve this optimization problem but at least we could use L-BFGS algorithm to find the solution.

Let's look at a special case in 2-dimension surfaces. In this scenario, the tangent space also has dimension 2. Therefore the Gaussian curvature is the same no matter how the two neighbor data points are chosen. Let $\boldsymbol{u}_1, \boldsymbol{u}_2$ be the orthonormal base of $T_p\mathcal{M}$ at point p, then we randomly choose a^l, b^l in the neighborhood of p, then 11 could be reduced to

$$\widehat{\boldsymbol{a}}^l = \widehat{a}_1\widehat{\boldsymbol{u}}_1 + \widehat{a}_2\widehat{\boldsymbol{u}}_2, \quad \widehat{\boldsymbol{b}} = \widehat{b}_1\widehat{\boldsymbol{u}}_1 + \widehat{b}_2\widehat{\boldsymbol{u}}_2 \tag{23}$$

Then we con have the $2^4 = 16-$vector

$$\widehat{\boldsymbol{a}^l} \otimes (\widehat{\boldsymbol{b}^l} \otimes (\widehat{\boldsymbol{b}^l \otimes \boldsymbol{a}^l})) = [\widehat{a}_1\widehat{b}_1\widehat{b}_1\widehat{a}_1, \widehat{a}_1\widehat{b}_1\widehat{b}_1\widehat{a}_2, \widehat{a}_1\widehat{b}_1\widehat{b}_2\widehat{a}_1, \cdots, \widehat{a}_2\widehat{b}_2\widehat{b}_2\widehat{a}_2] \tag{24}$$

Combining with our stretching matrix S we can rewrite the loss function as

$$\begin{aligned} E_2(l) &= \left(\widehat{K}(\widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l) - \frac{(\widehat{\boldsymbol{a}}^l \otimes (\widehat{\boldsymbol{b}}^l \otimes (\widehat{\boldsymbol{b}}^l \otimes \widehat{\boldsymbol{a}}^l)))^T \boldsymbol{SR}}{|\widehat{\boldsymbol{a}}^l|^2|\widehat{\boldsymbol{b}}^l|^2 - |\langle \widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l \rangle|^2} \right)^2 \\ &= \left(\widehat{K}(\widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l) + \frac{(\widehat{a}_1^l\widehat{b}_2^l - \widehat{a}_2^l\widehat{b}_1^l)^2 R_{1212}}{(\widehat{a}_2^l\widehat{b}_1^l - \widehat{a}_1^l\widehat{b}_2^l)^2} \right)^2 \\ &= \left(\widehat{K}(\widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l) + R_{1212} \right)^2 \end{aligned} \tag{25}$$

We can see that in the case of $2-$dimensions our optimization problem will degenerate to choose the number that minimuze the negative number of the estimated Gaussian curvature.

References

[1] H.T. Wu A.Singer. Vector diffusion maps and the connection laplacian. *arXiv:1102.0075v1*, Feb 2011.

[2] H.Zha Z.Zhang. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. In *International Conference on Intelligent Data Engineering and Automated Learning*, 2003.