Estimating Riemann Curvature Tensor on Point Cloud Data in High Dimension

张翰宇 2014012168

1 概要

今年暑假期间，我在美国西雅图的华盛顿大学统计系进行研修. 我在 Marina Meila 教授的指导下，对数据流形的曲率估计这一问题进行了较为深入的研究. 在高维数据处理中，我们经常把数据点看作是在一个潜在的数据流形上进行均匀采样得到的点集. 在之前的流形学习算法，例如 LLE，ISOMAP,LTSA 等算法中，流形学习的主要目标都是对高维数据进行降维. 然后再用我们常见的数据分析方法进行处理. 但 Marina Meila 教授和我的这个课题，开始关注数据流形内蕴的几何结构. 我们首先想要从数据中获得的就是这个潜在的数据流形的曲率的信息. 黎曼曲率张量是刻画一个黎曼流形局部"弯曲程度"的重要几何量. 它是一个 4 阶张量，在黎曼几何中扮演着极其重要的角色. 可以说，如果能够成功地将它计算出来，我们对于这个潜在的数据结构的形状就有了基本的把握. 为此我们目前已经得到了一个可以计算任意维度 d 的黎曼流形的曲率张量的算法. 我们主要利用的是曲率张量的线性性质和 Gauss-Bonnet 公式，将一个点和近邻的两个数据点进行组合，进行一次估计. 在重复多次后，将曲率张量的计算问题转化成了一个优化问题. 我们将设计出来的算法在 4 维球面，3 维平坦环面，2 维环面和 2 维伪球面上进行了模拟实验，得到了较准确的结果. 现在这个课题进入了理论证明阶段，我们将要证明算法的渐近无偏性，并分析复杂度.

2 Estimation Step

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

1. 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. Choose all possible 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. Solve an optimization problem to find the values of every component $Rm(X, Y, Z, W)$ of the Riemann curvature tensor.

2.1 Local PCA for tangent space estimation

As traditional 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] [4] 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

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

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.

2.2 Resampling to get sectional curvatures

For two points a, b in the neighborhood of point p, we suppose that γ_a and γ_b are the geodesics connecting p to a and b 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: the image of the exponential map of the plane in the tangent space spanned by $\{\boldsymbol{a}, \boldsymbol{b}\}$. 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. The Gauss-Bonnet formula will be simpler since we are considering geodesic triangle:

$$\iint_{\Delta pab} KdA = \alpha_p + \alpha_a + \alpha_b - \pi \tag{6}$$

Here α_p is the smaller angle between $\boldsymbol{a}, \boldsymbol{b}$ on the tangent plane. And α_a, α_b are similarly defined. Therefore, we have the following approximation of the Gaussian curvature of the 2-manifold $ii(\boldsymbol{a}, \boldsymbol{b})$

$$\widehat{K}(\boldsymbol{a}, \boldsymbol{b}) \approx \frac{\alpha_p + \alpha_a + \alpha_b - \pi}{\text{Area}(\Delta_{pab})} \tag{7}$$

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

$$K(\boldsymbol{a}, \boldsymbol{b}) = \frac{Rm(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{b}, \boldsymbol{a})}{|\boldsymbol{a}|^2|\boldsymbol{b}|^2 - \langle \boldsymbol{a}, \boldsymbol{b} \rangle^2} \tag{8}$$

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

$$\boldsymbol{a} = \sum_{i=1}^{d} a_i \boldsymbol{u}_{pi}, \quad \boldsymbol{b} = \sum_{i=1}^{d} b_i \boldsymbol{u}_{pi} \tag{9}$$

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

$$Rm(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{b}, \boldsymbol{a}) = \sum_{i_1=1}^{d} \sum_{i_2=1}^{d} \sum_{i_3=1}^{d} \sum_{i_4=1}^{d} a_{i_1} b_{i_2} b_{i_3} a_{i_4} Rm(\boldsymbol{u}_{i_1}, \boldsymbol{u}_{i_2}, \boldsymbol{u}_{i_3}, \boldsymbol{u}_{i_4}) \tag{10}$$

Therefore with 8 and 10,

$$K(\boldsymbol{a}, \boldsymbol{b}) \left(\sum_{i=1}^{d} a_i^2 \sum_{j=1}^{d} b_j^2 - (\sum_{k=1}^{d} a_k b_k)^2 \right) = \sum_{i_1=1}^{d} \sum_{i_2=1}^{d} \sum_{i_3=1}^{d} \sum_{i_4=1}^{d} a_{i_1} b_{i_2} b_{i_3} a_{i_4} Rm(\boldsymbol{u}_{i_1}, \boldsymbol{u}_{i_2}, \boldsymbol{u}_{i_3}, \boldsymbol{u}_{i_4}) \tag{11}$$

And we substitute our estimation of K there, we have a linear equation for the components $R_{ijkl} = Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l)$.

Now let's repeat this process for every possible combination of $(\boldsymbol{a}, \boldsymbol{b})$ in neighborhood of $\boldsymbol{p}$ and satisfying that $\| \boldsymbol{a} - \boldsymbol{b} \| < \varepsilon$. We naively use the projection of $s - p, t - p$ onto the tangent space $T_p\mathcal{M}$ to approximate $\boldsymbol{a}, \boldsymbol{b}$.

$$\widehat{\boldsymbol{a}} = O_p^T(s - p) = \sum_{i=1}^{d} \widehat{a}_i \widehat{\boldsymbol{u}}_i, \quad \widehat{\boldsymbol{b}} = O_p^T(s - p) = \sum_{i=1}^{d} \widehat{b}_i \widehat{\boldsymbol{u}}_i \tag{12}$$

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

$$\widehat{K}(\widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l) \left(\sum_{i=1}^{d} (\widehat{a}_i^l)^2 \sum_{j=1}^{d} (\widehat{b}_j^l)^2 - (\sum_{k=1}^{d} \widehat{a}_k^l \widehat{b}_k^l)^2 \right) = \sum_{i_1=1}^{d} \sum_{i_2=1}^{d} \sum_{i_3=1}^{d} \sum_{i_4=1}^{d} \widehat{a}_{i_1}^l \widehat{b}_{i_2}^l \widehat{b}_{i_3}^l \widehat{a}_{i_4}^l Rm(\widehat{\boldsymbol{u}}_{i_1}, \widehat{\boldsymbol{u}}_{i_2}, \widehat{\boldsymbol{u}}_{i_3}, \widehat{\boldsymbol{u}}_{i_4}) \tag{13}$$

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 $\{1, 2, \cdots, d\}^4$

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

$$Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l) + Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{u}}_l, \widehat{\boldsymbol{u}}_j) + Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_l, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_k) = 0 \tag{15}$$

2.3 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

$$\text{vec}(\boldsymbol{A}) = [a_{11}, a_{12}, \cdots, a_{1n}, a_{21}, a_{22}, \cdots, a_{nn}]^T \tag{16}$$

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

$$\begin{pmatrix} Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_1, \widehat{\boldsymbol{u}}_1) & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_1, \widehat{\boldsymbol{u}}_2) & \cdots & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_1, \widehat{\boldsymbol{u}}_d) \\ Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_2, \widehat{\boldsymbol{u}}_1) & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_2, \widehat{\boldsymbol{u}}_2) & \cdots & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_2, \widehat{\boldsymbol{u}}_d) \\ \vdots & \vdots & \vdots & \vdots \\ Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_d, \widehat{\boldsymbol{u}}_1) & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_d, \widehat{\boldsymbol{u}}_2) & \cdots & Rm(\widehat{\boldsymbol{u}}_i, \widehat{\boldsymbol{u}}_j, \widehat{\boldsymbol{u}}_d, \widehat{\boldsymbol{u}}_d) \end{pmatrix} \tag{17}$$

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

$$\widetilde{\boldsymbol{Rm}} = \begin{pmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} & \cdots & \boldsymbol{R}_{1d} \\ \boldsymbol{R}_{21} & \boldsymbol{R}_{22} & \cdots & \boldsymbol{R}_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{R}_{d1} & \boldsymbol{R}_{d2} & \cdots & \boldsymbol{R}_{dd} \end{pmatrix} \tag{18}$$

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 13 we can get a loss function

$$E_d(l) = \left(\widehat{K}(\widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l) - \frac{(\widehat{\boldsymbol{a}}^l \otimes (\widehat{\boldsymbol{b}}^l \otimes (\widehat{\boldsymbol{b}}^l \otimes \widehat{\boldsymbol{a}}^l)))^T \boldsymbol{Rm}}{|\widehat{\boldsymbol{a}}^l|^2 |\widehat{\boldsymbol{b}}^l|^2 - |\langle \widehat{\boldsymbol{a}}^l, \widehat{\boldsymbol{b}}^l \rangle|^2} \right)^2 \tag{19}$$

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.

$$\begin{aligned} \textbf{min} \quad & \rho(E(l)) \\ \textbf{s.t.} \quad & R_{ijkl} = -R_{jikl} = -R_{ijlk} = R_{klij} \\ & R_{ijkl} + R_{iklj} + R_{iljk} = 0 \end{aligned} \tag{20}$$

Here ρ can be choosed as Huber loss to get a stable result.

$$\rho(u) = \begin{cases} u^2, & |u| \le M \\ M(2|u| - M), & |u| > M \end{cases} \tag{21}$$

It is not easy to solve this constrained optimization problem 20 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:

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

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

$$\begin{aligned}
\boldsymbol{R}_{f_2(i,j)} &= R_{ijij} \\
\boldsymbol{R}_{\binom{d}{2}+3(f_3(i,j,k)-1)+1} &= R_{ijki} \\
\boldsymbol{R}_{\binom{d}{2}+3(f_3(i,j,k)-1)+2} &= R_{jikj} \\
\boldsymbol{R}_{\binom{d}{2}+3(f_3(i,j,k)-1)+3} &= R_{kijk} \\
\boldsymbol{R}_{\binom{d}{2}+3\binom{d}{3}+2(f_4(i,j,k,l)-1)+1} &= R_{ijkl} \\
\boldsymbol{R}_{\binom{d}{2}+3\binom{d}{3}+2(f_4(i,j,k,l)-1)+2} &= R_{ikjl}
\end{aligned} \tag{22}$$

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

$$\boldsymbol{R}=R_{1212}, \boldsymbol{S}=[0,0,0,0,0,1,-1,0,0,-1,1,0,0,0,0,0]^T \tag{23}$$

And then the problem 20 can be transfered into a non constraint optimization problem with a newly defined loss function

$$E_d(l)=\left(\widehat{K}(\widehat{\boldsymbol{a}}^l,\widehat{\boldsymbol{b}}^l)-\frac{(\widehat{\boldsymbol{a}}^l\otimes(\widehat{\boldsymbol{b}}^l\otimes(\widehat{\boldsymbol{b}}^l\otimes\widehat{\boldsymbol{a}}^l)))^T\boldsymbol{S}\boldsymbol{R}}{|\widehat{\boldsymbol{a}}^l|^2|\widehat{\boldsymbol{b}}^l|^2-|\langle\widehat{\boldsymbol{a}}^l,\widehat{\boldsymbol{b}}^l\rangle|^2}\right)^2 \tag{24}$$

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 iterative reweighted least squares 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 12 could be reduced to

$$\widehat{\boldsymbol{a}}^l=\widehat{a}_1\widehat{\boldsymbol{u}}_1+\widehat{a}_2\widehat{\boldsymbol{u}}_2,\quad \widehat{\boldsymbol{b}}=\widehat{b}_1\widehat{\boldsymbol{u}}_1+\widehat{b}_2\widehat{\boldsymbol{u}}_2 \tag{25}$$

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

$$\widehat{\boldsymbol{a}^l}\otimes(\widehat{\boldsymbol{b}^l}\otimes(\widehat{\boldsymbol{b}^l}\otimes\widehat{\boldsymbol{a}^l}))=[\widehat{a}_1\widehat{b}_1\widehat{b}_1\widehat{a}_1,\widehat{a}_1\widehat{b}_1\widehat{b}_1\widehat{a}_2,\widehat{a}_1\widehat{b}_1\widehat{b}_2\widehat{a}_1,\cdots,\widehat{a}_2\widehat{b}_2\widehat{b}_2\widehat{a}_2] \tag{26}$$

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

$$\begin{aligned}
E_2(l)&=\left(\widehat{K}(\widehat{\boldsymbol{a}}^l,\widehat{\boldsymbol{b}}^l)-\frac{(\widehat{\boldsymbol{a}}^l\otimes(\widehat{\boldsymbol{b}}^l\otimes(\widehat{\boldsymbol{b}}^l\otimes\widehat{\boldsymbol{a}}^l)))^T\boldsymbol{S}\boldsymbol{R}}{|\widehat{\boldsymbol{a}}^l|^2|\widehat{\boldsymbol{b}}^l|^2-|\langle\widehat{\boldsymbol{a}}^l,\widehat{\boldsymbol{b}}^l\rangle|^2}\right)^2 \\
&=\left(\widehat{K}(\widehat{\boldsymbol{a}}^l,\widehat{\boldsymbol{b}}^l)+\frac{(\widehat{a}_1^l\widehat{b}_2^l-\widehat{a}_2^l\widehat{b}_1^l)^2R_{1212}}{(\widehat{a}_2^l\widehat{b}_1^l-\widehat{a}_1^l\widehat{b}_2^l)^2}\right)^2 \\
&=\left(\widehat{K}(\widehat{\boldsymbol{a}}^l,\widehat{\boldsymbol{b}}^l)+R_{1212}\right)^2
\end{aligned} \tag{27}$$

We can see that in the case of 2−dimensions our optimization problem will degenerate to find value that minimizes the negative of the estimated Gaussian curvature.

Now we can summarize our estimation step into a curvature estimation algorithm

Algorithm 1 Curvature Tensor Estimation

Input: N data points in dimension D $\{x_i\}_{i=1}^N$, a point p that we want to estimated the curvature tensor

Output: The coefficients of Riemann curvature tensor at point p

1: Construct the $\varepsilon-$neighbor graph $\mathcal{G}$ and the weight matrix W using the heat kernel
2: Conduct local PCA algorithm to estimate local tangent space at p and all neighbors of p
3: **for** every pair of (a, b) in the neighborhood of p **do**
4: **if** b is in the neighborhood of a **then**
5: Estimate the gaussian curvature using Gauss-Bonnet formula
6: **end if**
7: **end for**
8: Minimizing loss function 24 to get the estimation.

3 Simulation

In this section results of some numeric simulation experiments are reported. We tested our algorithm on the following four models:

- 4-dimension sphere $\mathbb{S}^4$ embedded in 5 dimension Eulidean space.
- 3-dimension flat torus $\mathbb{T}^3$ embedded in 4 dimension Euclidean space.
- 2-dimension curved torus embedded in 3 dimension space
- 2-dimension pseudosphere embedded in 2 dimension space

For every model, we fixed the sample size 20000, and sampled uniformly distributed on it (specified later). Then we choose epsilon such that for each point the average number of neighbors is approximately 10^2. We randomly choose one point a time in every model, calculate the Riemann curvature there as we proposed and then repeat the whole process for 10000 times. The results are evaluated (specified later) basically by the mean absolute value of the Frobenius norm of our estimated $\widehat{\boldsymbol{R}}$ and the true value $\boldsymbol{R}$

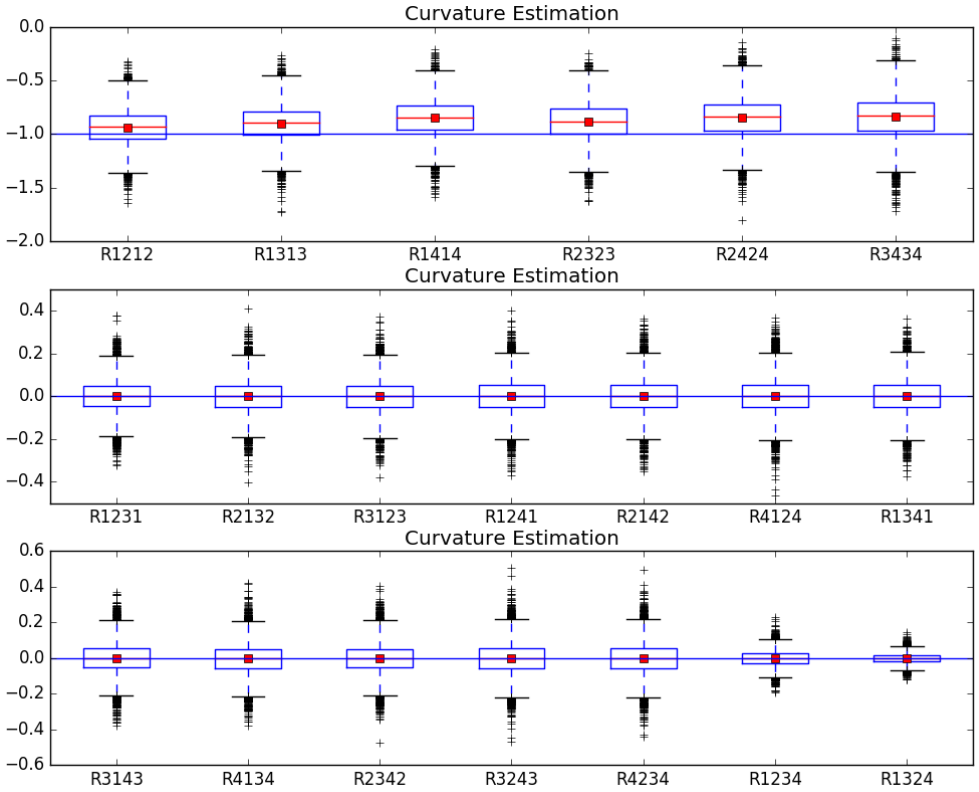
4d-Sphere in $\mathbb{R}^5$ The data is sampled as [3], we generate 4-dimension standard normal distribution random numbers and then normalise them to get uniform distributed data. The true value of Riemann curvature tensor of a 4d-sphere with radius R is

$$R_{ijkl} = \frac{1}{R^2}(g_{il}g_{jk} - g_{ik}g_{jl}) \tag{28}$$

for any basis. Therefore with $R = 1$,the true value of $\boldsymbol{R}$ is

$$\boldsymbol{R} = (\overbrace{-1, -1, \cdots, -1}^{6}, \overbrace{0, 0, \cdots, 0}^{10}) \tag{29}$$

In case that the average number of neighbors for every point is approximately 100, we choose $\epsilon = 0.40$ Then the distribution of repeated result of estimation can be shown in figures.



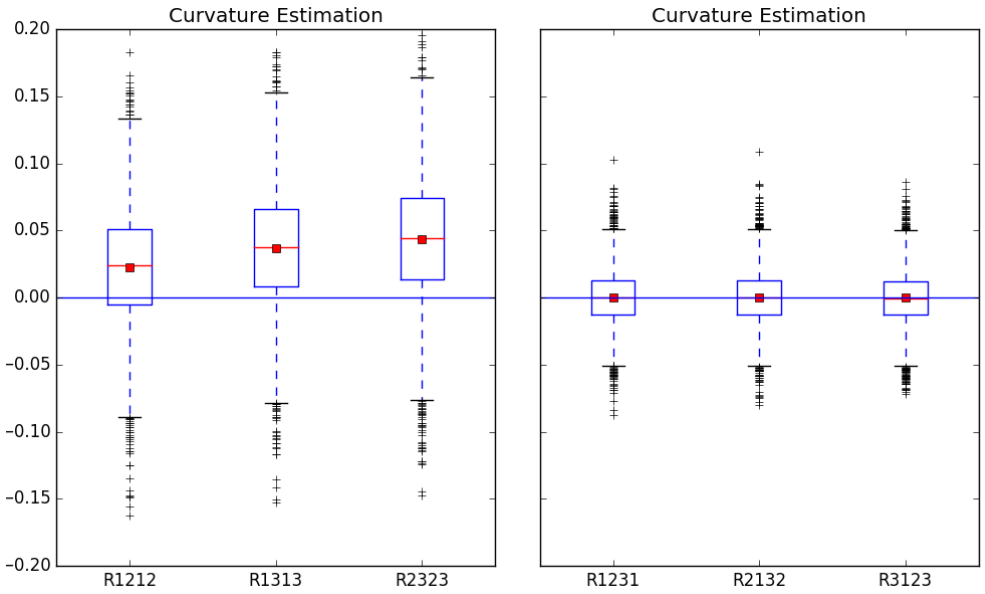
3d-Flat torus in $\mathbb{R}^4$ Here the 3d-flat torus is given by the parametrization

$$(\cos u, \sin u, \cos v, \sin v, \cos w, \sin w), \quad u, v, w \in [0, 2\pi) \tag{30}$$

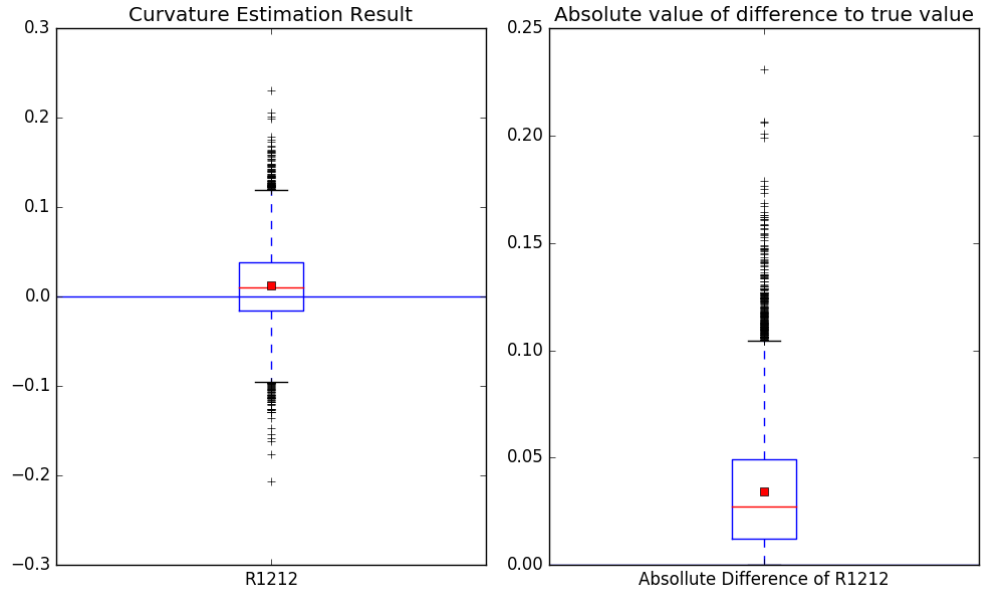
And the true value for every component is 0 according to standard calculation of Riemann curvature tensor. The data is sampled uniformly by first finding uniformly distributed sample on three perpendicular $\mathbb{S}^1$ circles and then generate there cartesian product.In our simulation, we select $R = 2, r = 1$. We choose epsilon $= 0.63$ to guarantee approximately 100 neighbors in average for every point. The true value of curvature tensor for flat torus is zero for every component. The estimation results are shown in figures.

2d-curved torus in $\mathbb{R}^3$ Here we test our algorithm on varying sectional curvature case. The curved torus is given by the parametrization

$$((R + r\cos\theta)\cos\phi, (R + r\cos\theta)\sin\phi, r\sin\theta), \quad 0 \leq \theta, \phi \leq 2\pi \tag{31}$$



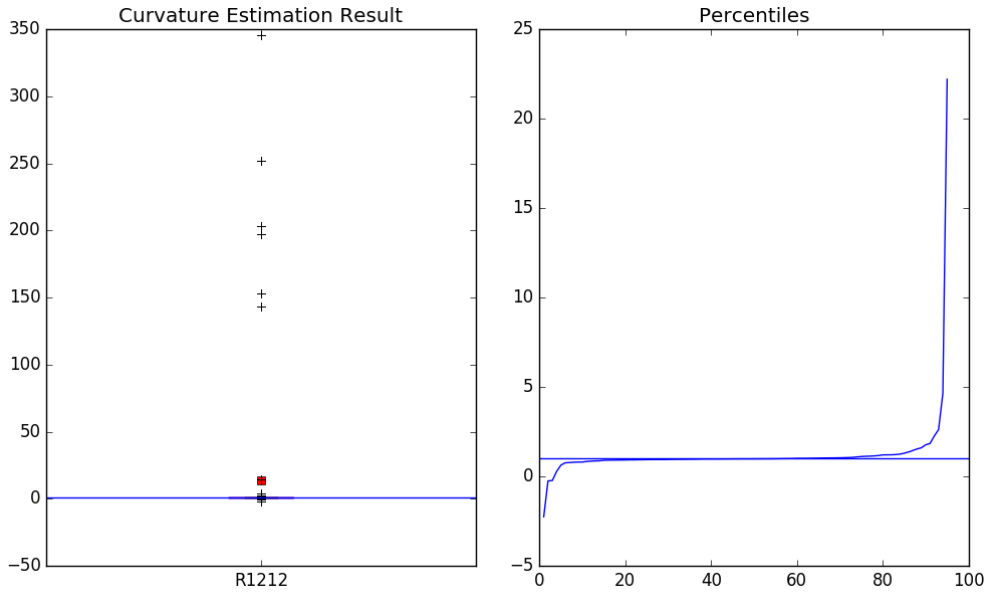
Sampling is conducted as [2] Example 1A. At point (θ, ϕ), since it is only two dimensional, the only one independent component of the curvature tensor is $R_{1212} = -\frac{\cos\theta}{R+r\cos\theta}$. Then we can see the estimated result as



2d-pseudosphere in $\mathbb{R}^3$ Pseusosphere is one representative of negative constant gaussian curvature, given by the parametrization

$$(\cos u \sin v, \sin u \sin v, \cos u + \log(\tan(\frac{v}{2}))), \quad u \in [0, 2\pi), v \in [0, \pi) \tag{32}$$

We choose $\epsilon = 0.2$ to have a proper number of average neighbors. The results are shown as



参考文献

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

[2] Persi Diaconis, Susan Holmes, and Mehrdad Shahshahani. *Sampling from a Manifold*, volume Volume 10 of *Collections*, pages 102–125. Institute of Mathematical Statistics, Beachwood, Ohio, USA, 2013.

[3] Mervin E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2(4):19–20, April 1959.

[4] 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.