Kernel density estimation with spherical data

Miaolei Bao

June 2, 2024

Statement: In the final project, I aim to study the method introduced in the paper, "Kernel Density Estimation with Spherical Data"[1], authored by Peter Hall et al. The proposed estimator will be implemented in R and validated using simulated and real-world data. The project intends to broaden the application of kernel density estimation to higher dimensions and can be applied to address real-world scientific questions associated with estimating density on spherical surfaces.

1 Introduction

Kernel density estimation (KDE) is a fundamental method in nonparametric estimation. In class, we have explored its application in 1D and 2D scenarios using a Gaussian kernel and discussed the extension to complex domains. In practice, most scientific problems are not only limited to two dimensions. Particularly in fields such as ecology, geology, and atmospheric science, collected data are often macroscopic, with longitude and latitude coordinates. Thus, expanding density estimation into 3D settings is crucial for real-world applications.

2 Methodology

In 1D and 2D cases, the KDE that we are familiar with, as defined in class, takes the following form (assuming $h_1 = \cdots = h_p = h$):

$$\hat{f}_n(x;h) = \frac{1}{nh^p}\sum_{i=1}^{n} K(\frac{x - X_i}{h})$$

Here $x - X_i$ represents the displacement in Euclidean space. To illustrate instead the displacement on sphere surfaces, we need a new measurement. Assume we have two unit vectors x and y, and the angle between them has a cosine value equal to $x^T y$. Replacing the displacement term in the above equation with this new measurement, we have:

$$\hat{f}_n(x;\kappa) = \frac{1}{n}c_0(\kappa)\sum_{i=1}^{n} K(\kappa x^T X_i)$$

where κ is the new smoothing parameter and $c_0(\kappa)$ is chosen so that $\hat{f}_n(x;\kappa)$ integrates to unity. Since $x^T X_i$ equals the cosine value of the angle between vectors x^T and X_i, it only takes values from -1 to 1. When the two vectors are close to each other, unlike in 1D and 2D settings where the displacement term approaches 0, here it tends towards unity. Therefore, unlike the Gaussian kernel commonly used before, a kernel with different characteristics, such as rapidly varying, should be employed in higher dimensions. One such function is $K(t) = e^t$.

According to the paper, the estimator $\hat{f}_n(x;\kappa)$ is asymptotically unbiased. An explicit formula for bias is derived, which further facilitates evaluation for squared-error and Kullback-Leibler losses. When optimal κ is chosen by cross-validation and sample size n goes to infinity, the asymptotic minimum squared-error loss and the asymptotic minimum Kullback-Leibler loss both do not depend on the kernel K. Therefore, all estimators of this class are equivalent to each other asymptotically and to the special estimator using $K(t) = e^t$.

The rate of convergence based on these two losses is both $n^{-\frac{2}{p+3}}$, where p is $\frac{1}{2}(q-3)$. Results from class show:

$$MSE(\hat{f}_n(x_0), f(x_0)) \le C^{\star\star} n^{\frac{-2\beta}{2\beta+q}}.$$

The optimal achievable convergence rate of L^2 norm for an estimator with up to second bounded derivatives (i.e., $\beta = 2$) is $n^{-\frac{4}{q+3}}$ in (q-1)-dimensional space, which is analogous to estimation on a q-dimensional sphere. $n^{-\frac{2}{p+3}} = n^{-\frac{4}{q+3}}$. We recover the same rate.

Therefore, this class of estimators demonstrates good overall properties, and based on the properties above, we assume it is reasonable to use the special estimator with $K(t) = e^t$ for the following implementation.

3 Implementation

3.1 Kernel

Considering practicality and computational simplicity, I only implemented the estimator for 3D spherical data in the final project using $K(t) = e^t$. In this case, we have $c_0(\kappa) = \kappa(4\pi sinh\kappa)^{-1}$ for:

$$\hat{f}_n(x;\kappa) = \frac{1}{n} c_0(\kappa) \sum_{i=1}^{n} K(\kappa x^T X_i)$$

Here X_i and x require data in Cartesian coordinates. I wrote a function for transforming data in spherical coordinates (polar coordinates) to Cartesian coordinates.

3.2 Date generation

For data simulation, I followed the procedure described in an article[2] about an R package "spherepc". I generated half-great circle, circular, and noisy zigzag data to assess the performance of the proposed estimator across different scenarios. The data were generated with various noise levels and in spherical coordinates, and subsequently transformed to Cartesian coordinates for analysis.

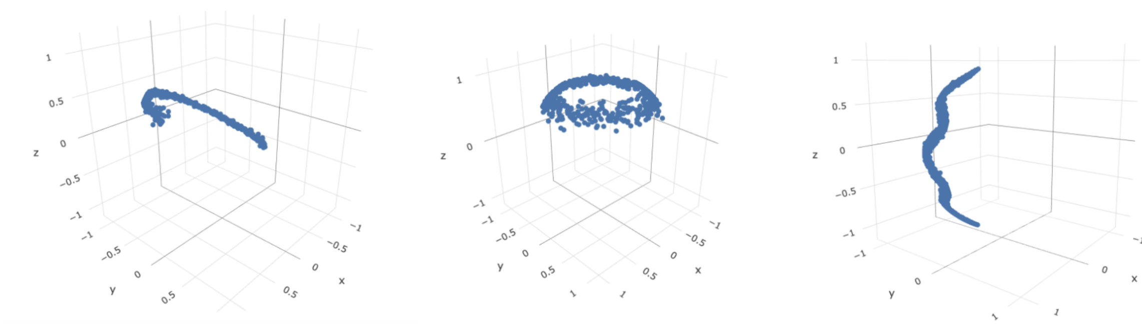

Figure 1: Visualization of generated data (half-great circle, circular, and noisy zigzag)

3.3 Density estimation

I created a spherical grid with 300 and 300 spherical coordinates, each coordinate uniformly ranging from -180 to 180, with a total of 90000 locations. Then I transformed the coordinates into Cartesian and used the KDE for prediction. For now, I used two values for the smoothing parameter κ, 1 and 100, to demonstrate that in 3D kernel estimation, a larger smoothing parameter corresponds to a smaller h in 1D and 2D settings.

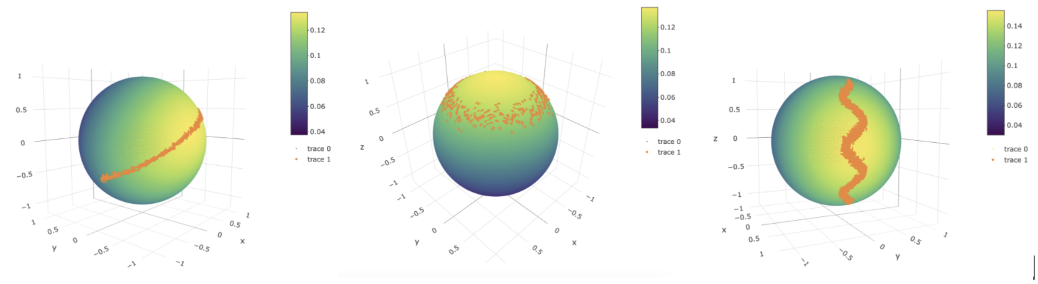

Figure 2: Visualization of prediction for $\kappa = 1$

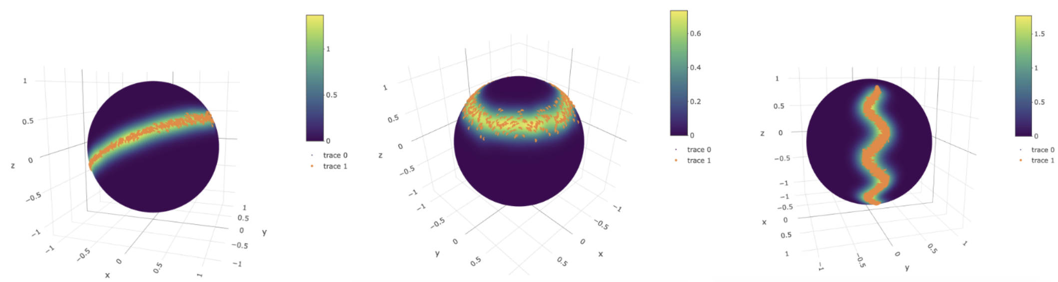

Figure 3: Visualization of prediction for $\kappa = 100$

$\hat{f}_n(x;\kappa)$ is depicted in Figure 2 and Figure 3 by the surface color. Light colors represent higher density, while darker colors represent lower density. We can observe that the estimation values are symmetric around data points and, in certain areas, similar values form circles. This occurs because any point on the surface forming a vector with the same angle with the observation vector has the same estimation.

3.4 Cross validation

In the final project, I implemented the Kullback-Leibler loss. The density estimate constructed by the leaving-one-out method is:

$$\hat{f}_i(x;\kappa) = \frac{1}{n-1} c_0(\kappa) \sum_{j \neq i} K(\kappa x^T X_j)$$

We have the following cross-validation score:

$$CV_{KL} = \frac{1}{n} \sum_{i=1}^{n} log \hat{f}_i(X_i;\kappa)$$

Here the optimal κ maximizes CV_{KL} since the Kullback-Leibler loss has the form:

$$\mathscr{L}_{KL}(\kappa) = \int_{\Omega} f(x) \mathbb{E}[log \frac{f(x)}{\hat{f}(x;\kappa)}] \omega(dx)$$

3.5 Example: earthquake

In addition, there is an earthquake dataset, containing 77 observations in the 'spherepc' package, which I used as a real-world example in this part. The data record significant earthquakes (8+ Mb magnitude) around the Pacific Ocean since 1900, collected by the U.S. Geological Survey. The data have longitude and latitude coordinates. For analysis, I went through the same procedure described before, but instead of using a pre-specified smoothing parameter κ, I used cross-validation to select the optimal *kappa*. The list of *kappa* is from:

```
10^seq(1, 3, length.out = 1000)
```

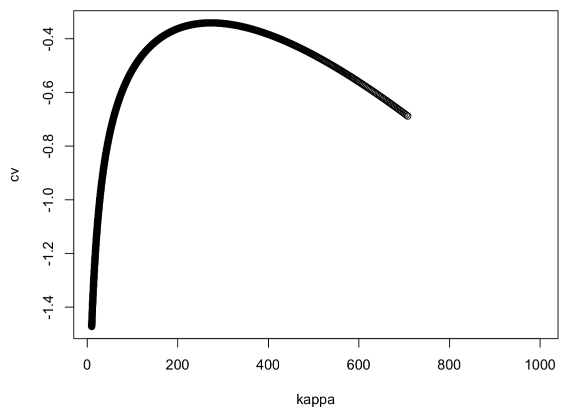

Figure 4: Visualization of cross-validation results

The optimal κ for earthquake data is 273.8025. The following plot shows part of the visualization based on the optimal κ.

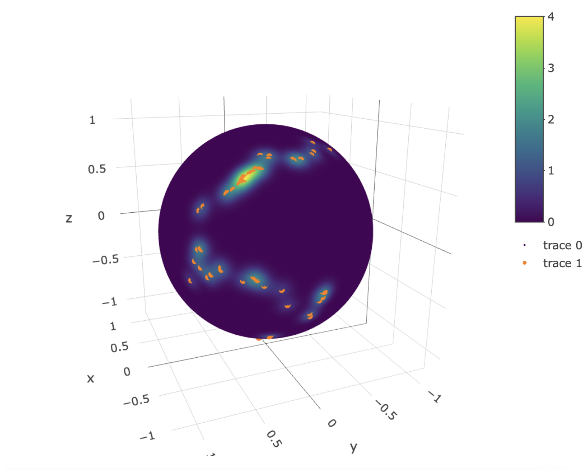

Figure 5: Kernel density estimation of earthquake data based on the optimal κ

4 Reflection

1. The authors used equal area projections of the unit sphere onto a disc in the paper. Compared with color patterns on the surface, as shown here in the report, the plots in the paper, while appearing crude, better demonstrated density peaks. Unfortunately, I don't have time to work on it for the project but might attempt it afterwards.

2. Another problem I encountered during the process is about the kernel $K(t) = e^t$. While increasing κ for the simulation data, I was unable to find the optimal κ. As κ grows large, the term $K(\kappa x^T X_j)$ in CV_{KL} increases exponentially and cannot be balanced by $c_0(\kappa)$. As a result, estimates at large κ result in infinite values, hindering further progress. Some curves had an increasing trend and were cut off due to "inf" values before reaching the optimal κ. Perhaps a kernel with less drastic changes would be better-performed.

References

[1] Peter Hall, G. S. Watson, and Javier Cabrera. Kernel density estimation with spherical data. *Biometrika*, 74(4):751–762, 1987.

[2] Jongmin Lee, Jang-Hyun Kim, and Hee-Seok Oh. spherepc: An r package for dimension reduction on a sphere. *The R Journal*, 14:167–181, 2022. https://doi.org/10.32614/RJ-2022-016.