Multivariate Kernel Density Estimation

Computational Statistics Course Project

Khang Nguyen *25C01035*
Nhat Tien Nguyen *25C01039*
Phuc Loc Nguyen *25C01041*

Master of Data Science
University of Science, VNU-HCM

February 1, 2026

Outline

1. Problem
2. Simplification by Product Kernel method
3. Radial Symmetry Kernel Asumnption

Problem Statement

- Kernel density estimation (KDE) is a flexible nonparametric method.

Problem Statement

- Kernel density estimation (KDE) is a flexible nonparametric method.
- Univariate KDE is well understood and easy to implement.

Problem Statement

- Kernel density estimation (KDE) is a flexible nonparametric method.
- Univariate KDE is well understood and easy to implement.
- In practice, data are often multivariate.

Problem Statement

- Kernel density estimation (KDE) is a flexible nonparametric method.
- Univariate KDE is well understood and easy to implement.
- In practice, data are often multivariate.
- This motivates extending KDE to p-dimensional spaces.

The Curse of Dimensionality

The Challenge

High-dimensional space ($p > 3$) is vastly different from univariate space. Data becomes incredibly **sparse**.

The Curse of Dimensionality

The Challenge

High-dimensional space ($p > 3$) is vastly different from univariate space. Data becomes incredibly **sparse**.

Key Implications:

- Points have very few near neighbors.
- "Empty space" phenomenon: For $p = 10$, $> 50\%$ of mass lies in the extreme tails ($> 1.6\sigma$).
- Visualization is difficult without dimension reduction.

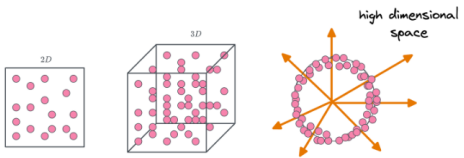


Figure: Sparseness in High Dimensions

General Multivariate Estimator

Let $\mathbf{x}$ be a p-dimensional random vector. The estimator is:

General Multivariate KDE (Eq. 10.40)

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right) \tag{1}$$

Core Components:

General Multivariate Estimator

Let $\mathbf{x}$ be a p-dimensional random vector. The estimator is:

General Multivariate KDE (Eq. 10.40)

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right) \tag{1}$$

Core Components:

- $\mathbf{H}$ (Bandwidth Matrix): Controls **shape & orientation** of the kernel.

General Multivariate Estimator

Let $\mathbf{x}$ be a p-dimensional random vector. The estimator is:

General Multivariate KDE (Eq. 10.40)

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right) \tag{1}$$

Core Components:

- $\mathbf{H}$ (Bandwidth Matrix): Controls **shape & orientation** of the kernel.
- $|\mathbf{H}|$ (Volume Factor): Normalizes the density to ensure $\int \hat{f} = 1$.

General Multivariate Estimator

Let $\mathbf{x}$ be a p-dimensional random vector. The estimator is:

General Multivariate KDE (Eq. 10.40)

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right) \tag{1}$$

Core Components:

- $\mathbf{H}$ (Bandwidth Matrix): Controls **shape & orientation** of the kernel.
- $|\mathbf{H}|$ (Volume Factor): Normalizes the density to ensure $\int \hat{f} = 1$.
- K (Kernel Function): A symmetric probability density (e.g., Gaussian).

General Multivariate Estimator

Let $\mathbf{x}$ be a p-dimensional random vector. The estimator is:

General Multivariate KDE (Eq. 10.40)

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right) \tag{1}$$

Core Components:

- $\mathbf{H}$ (Bandwidth Matrix): Controls **shape & orientation** of the kernel.
- $|\mathbf{H}|$ (Volume Factor): Normalizes the density to ensure $\int \hat{f} = 1$.
- K (Kernel Function): A symmetric probability density (e.g., Gaussian).

Constraint: Optimizing a full $\mathbf{H}$ requires estimating $O(p^2)$ parameters.

Practical Approach: Product Kernels

To simplify complexity, we assume **independence** between dimensions.

- This implies the Bandwidth Matrix $\mathbf{H}$ is **diagonal**.

The Product Estimator

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\prod_{j=1}^{p}\frac{1}{h_j}K\left(\frac{x_j - X_{ij}}{h_j}\right)}_{\text{Product of univariate kernels}} \tag{2}$$

Key Advantages:

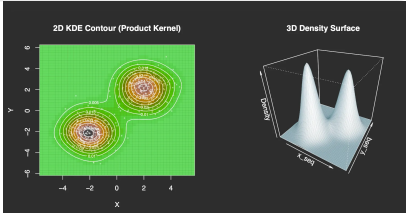


Figure: Consequence: Axis-aligned Contours

Practical Approach: Product Kernels

To simplify complexity, we assume **independence** between dimensions.

- This implies the Bandwidth Matrix $\mathbf{H}$ is **diagonal**.

The Product Estimator

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\prod_{j=1}^{p}\frac{1}{h_j}K\left(\frac{x_j - X_{ij}}{h_j}\right)}_{\text{Product of univariate kernels}} \tag{2}$$

Key Advantages:

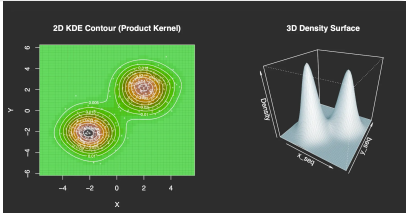


Figure: Consequence: Axis-aligned Contours

Practical Approach: Product Kernels

To simplify complexity, we assume **independence** between dimensions.

- This implies the Bandwidth Matrix $\mathbf{H}$ is **diagonal**.

The Product Estimator

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\prod_{j=1}^{p}\frac{1}{h_j}K\left(\frac{x_j - X_{ij}}{h_j}\right)}_{\text{Product of univariate kernels}} \tag{2}$$

Key Advantages:

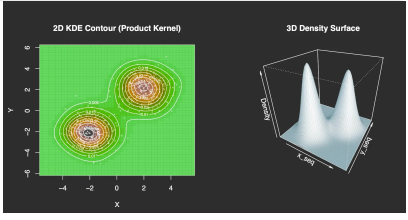


Figure: Consequence: Axis-aligned Contours

Practical Approach: Product Kernels

To simplify complexity, we assume **independence** between dimensions.

- This implies the Bandwidth Matrix $\mathbf{H}$ is **diagonal**.

The Product Estimator

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\prod_{j=1}^{p}\frac{1}{h_j}K\left(\frac{x_j - X_{ij}}{h_j}\right)}_{\text{Product of univariate kernels}} \tag{2}$$

Key Advantages:

- **Parameter Reduction:** Reduces unknowns from $O(p^2)$ (matrix) to $O(p)$ (vector).

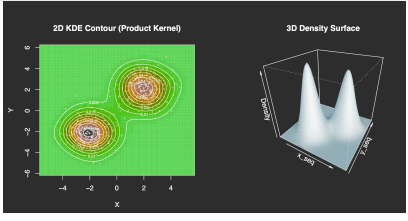


Figure: Consequence: Axis-aligned Contours

Practical Approach: Product Kernels

To simplify complexity, we assume **independence** between dimensions.

- This implies the Bandwidth Matrix $\mathbf{H}$ is **diagonal**.

The Product Estimator

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\prod_{j=1}^{p}\frac{1}{h_j}K\left(\frac{x_j - X_{ij}}{h_j}\right)}_{\text{Product of univariate kernels}} \tag{2}$$

Key Advantages:

- **Parameter Reduction:** Reduces unknowns from $O(p^2)$ (matrix) to $O(p)$ (vector).
- **Computational Efficiency:** No expensive matrix inversion needed.

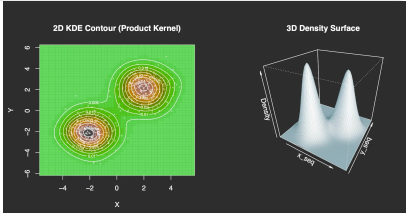


Figure: Consequence: Axis-aligned Contours

Practical Approach: Product Kernels

To simplify complexity, we assume **independence** between dimensions.

- This implies the Bandwidth Matrix $\mathbf{H}$ is **diagonal**.

The Product Estimator

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{\prod_{j=1}^{p} \frac{1}{h_j} K\left(\frac{x_j - X_{ij}}{h_j}\right)}_{\text{Product of univariate kernels}} \tag{2}$$

Key Advantages:

- **Parameter Reduction:** Reduces unknowns from $O(p^2)$ (matrix) to $O(p)$ (vector).
- **Computational Efficiency:** No expensive matrix inversion needed.
- **Flexibility:** Allows distinct bandwidth h_j for each dimension.

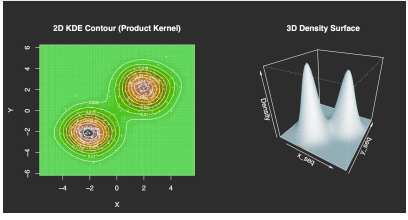


Figure: Consequence: Axis-aligned Contours

Bandwidth Selection (Scott's Rule)

To minimize the error (AMISE), we select the bandwidth h_j for each dimension:

Multivariate Scott's Rule

$$h_j = \hat{\sigma}_j \left(\frac{4}{n(p+2)} \right)^{\frac{1}{p+4}} \tag{3}$$

Interpretation:

Bandwidth Selection (Scott's Rule)

To minimize the error (AMISE), we select the bandwidth h_j for each dimension:

Multivariate Scott's Rule

$$h_j = \hat{\sigma}_j \left(\frac{4}{n(p+2)} \right)^{\frac{1}{p+4}} \tag{3}$$

Interpretation:

- $\hat{\sigma}_j$: The standard deviation (scale) of variable j.

Bandwidth Selection (Scott's Rule)

To minimize the error (AMISE), we select the bandwidth h_j for each dimension:

Multivariate Scott's Rule

$$h_j = \hat{\sigma}_j \left(\frac{4}{n(p+2)} \right)^{\frac{1}{p+4}} \tag{3}$$

Interpretation:

- $\hat{\sigma}_j$: The standard deviation (scale) of variable j.
- n: Sample size. More data $\rightarrow$ smaller bandwidth (finer detail).

Bandwidth Selection (Scott's Rule)

To minimize the error (AMISE), we select the bandwidth h_j for each dimension:

Multivariate Scott's Rule

$$h_j = \hat{\sigma}_j \left(\frac{4}{n(p+2)} \right)^{\frac{1}{p+4}} \tag{3}$$

Interpretation:

- $\hat{\sigma}_j$: The standard deviation (scale) of variable j.
- n: Sample size. More data $\rightarrow$ smaller bandwidth (finer detail).
- p: Number of dimensions.
  - As $p \uparrow$, data becomes sparse.
  - We need **wider bandwidths** to cover the gaps.

Alternative Simplification: Radial Symmetry

Instead of multiplying univariate kernels, what if we use a single **Radially Symmetric** kernel K for the whole p-dimensional space?

The Spherical Estimator (Eq. 10.42)

We use a **single scalar bandwidth** h for all dimensions:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^p} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right) \tag{4}$$

(Where K depends only on the distance $||\mathbf{x} - \mathbf{X}_i||$, e.g., Multivariate Epanechnikov)

Alternative Simplification: Radial Symmetry

Instead of multiplying univariate kernels, what if we use a single **Radially Symmetric** kernel K for the whole p-dimensional space?

The Spherical Estimator (Eq. 10.42)

We use a **single scalar bandwidth** h for all dimensions:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^p} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right) \tag{4}$$

(Where K depends only on the distance $||\mathbf{x} - \mathbf{X}_i||$, e.g., Multivariate Epanechnikov)

- **Geometry:** The kernel shape is a perfect **Hyper-Sphere** (Ball).

Alternative Simplification: Radial Symmetry

Instead of multiplying univariate kernels, what if we use a single **Radially Symmetric** kernel K for the whole p-dimensional space?

The Spherical Estimator (Eq. 10.42)

We use a **single scalar bandwidth** h for all dimensions:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^p} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right) \tag{4}$$

(Where K depends only on the distance $||\mathbf{x} - \mathbf{X}_i||$, e.g., Multivariate Epanechnikov)

- **Geometry:** The kernel shape is a perfect **Hyper-Sphere** (Ball).
- **Simplicity:** Only 1 parameter (h) to tune!

Alternative Simplification: Radial Symmetry

Instead of multiplying univariate kernels, what if we use a single **Radially Symmetric** kernel K for the whole p-dimensional space?

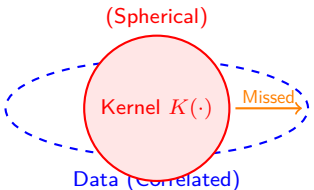
The Spherical Estimator (Eq. 10.42)

We use a **single scalar bandwidth** h for all dimensions:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^p} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right) \tag{4}$$

(Where K depends only on the distance $||\mathbf{x} - \mathbf{X}_i||$, e.g., Multivariate Epanechnikov)

- **Geometry:** The kernel shape is a perfect **Hyper-Sphere** (Ball).
- **Simplicity:** Only 1 parameter (h) to tune!



Alternative Simplification: Radial Symmetry

Instead of multiplying univariate kernels, what if we use a single **Radially Symmetric** kernel K for the whole p-dimensional space?

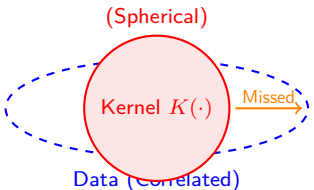
The Spherical Estimator (Eq. 10.42)

We use a **single scalar bandwidth** h for all dimensions:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^p} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right) \tag{4}$$

(Where K depends only on the distance $||\mathbf{x} - \mathbf{X}_i||$, e.g., Multivariate Epanechnikov)

- **Geometry:** The kernel shape is a perfect **Hyper-Sphere** (Ball).
- **Simplicity:** Only 1 parameter (h) to tune!



Implementation Workflow (Step-by-Step)

Standard Operating Procedure (SOP):

1. **Decompose:** Compute Eigen-decomposition of Covariance Matrix $\hat{\mathbf{\Sigma}} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$.

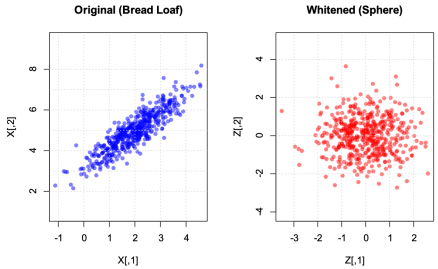


Figure: $\mathbf{X}$ (Correlated) $\rightarrow$ $\mathbf{Z}$ (Spherical)

Implementation Workflow (Step-by-Step)

Standard Operating Procedure (SOP):

1. **Decompose:** Compute Eigen-decomposition of Covariance Matrix $\hat{\mathbf{\Sigma}} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$.
2. **Transform (Whitening):** Rotate data to eliminate correlation.

$$\boxed{\mathbf{Z}_i = \mathbf{\Lambda}^{-1/2}\mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}})}$$

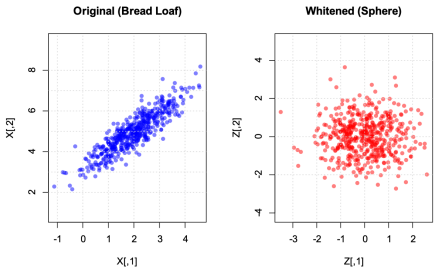


Figure: $\mathbf{X}$ (Correlated) $\rightarrow$ $\mathbf{Z}$ (Spherical)

Implementation Workflow (Step-by-Step)

Standard Operating Procedure (SOP):

1. **Decompose:** Compute Eigen-decomposition of Covariance Matrix $\hat{\mathbf{\Sigma}} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$.
2. **Transform (Whitening):** Rotate data to eliminate correlation.

$$\boxed{\mathbf{Z}_i = \mathbf{\Lambda}^{-1/2}\mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}})}$$

3. **Estimate:** Apply simple Product Kernel on the whitened data $\mathbf{Z}$.

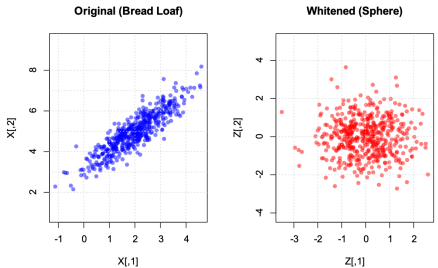


Figure: $\mathbf{X}$ (Correlated) $\rightarrow$ $\mathbf{Z}$ (Spherical)

Implementation Workflow (Step-by-Step)

Standard Operating Procedure (SOP):

1. **Decompose:** Compute Eigen-decomposition of Covariance Matrix $\hat{\mathbf{\Sigma}} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$.
2. **Transform (Whitening):** Rotate data to eliminate correlation.

$$\boxed{\mathbf{Z}_i = \mathbf{\Lambda}^{-1/2}\mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}})}$$

3. **Estimate:** Apply simple Product Kernel on the whitened data $\mathbf{Z}$.
4. **Reconstruct:** Adjust density scale to original space (Jacobian determinant).

$$\hat{f}_{\mathbf{X}}(\mathbf{x}) = \hat{f}_{\mathbf{Z}}(\mathbf{z}) \cdot |\hat{\mathbf{\Sigma}}|^{-1/2}$$

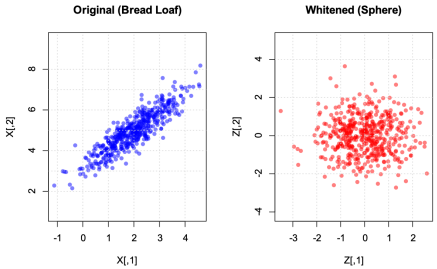


Figure: $\mathbf{X}$ (Correlated) $\rightarrow$ $\mathbf{Z}$ (Spherical)

Conceptual Comparison: PCA vs. Whitening (1/2)

The Shared DNA: Eigen-decomposition

$$\hat{\mathbf{\Sigma}} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$$ (**P**: Rotation Matrix, Λ: Scale Matrix)

Conceptual Comparison: PCA vs. Whitening (1/2)

The Shared DNA: Eigen-decomposition

$$\hat{\mathbf{\Sigma}} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$$ ($\mathbf{P}$: Rotation Matrix, Λ: Scale Matrix)

1. PCA (The Rotator)

Goal: Decorrelate (Align axes).

$$\mathbf{Y} = \underbrace{\mathbf{P}^T}_{\text{Rotate}}(\mathbf{x} - \bar{\mathbf{x}})$$

- **Shape:** Remains Elliptical.
- **Variance:** Preserved ($\mathbf{\Lambda}$).
- *Not enough for Product Kernels.*

Conceptual Comparison: PCA vs. Whitening (1/2)

The Shared DNA: Eigen-decomposition

$$\hat{\mathbf{\Sigma}} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T \qquad (\mathbf{P}\text{: Rotation Matrix, } \Lambda\text{: Scale Matrix})$$

1. PCA (The Rotator)

Goal: Decorrelate (Align axes).

$$\mathbf{Y} = \underbrace{\mathbf{P}^T}_{\text{Rotate}}(\mathbf{x} - \bar{\mathbf{x}})$$

- **Shape:** Remains Elliptical.
- **Variance:** Preserved ($\mathbf{\Lambda}$).
- *Not enough for Product Kernels.*

2. Whitening (The Squasher)

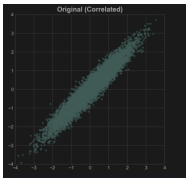
Goal: Isotropy (Make Spherical).

$$\mathbf{Z} = \underbrace{\mathbf{\Lambda}^{-1/2}}_{\text{Scale}}\underbrace{\mathbf{P}^T}_{\text{Rotate}}(\mathbf{x} - \bar{\mathbf{x}})$$

- **Shape:** Becomes Spherical.
- **Variance:** Normalized to 1 ($\mathbf{I}$).
- *Perfect for Product Kernels.*

Conceptual Comparison: Visual Geometry (2/2)

From Correlation to Isotropy: A Visual Journey

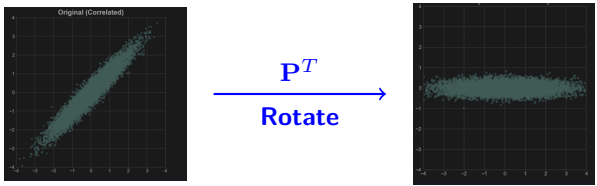


$\xrightarrow[\text{Rotate}]{\mathbf{P}^T}$

Original
(Correlated)

Conceptual Comparison: Visual Geometry (2/2)

From Correlation to Isotropy: A Visual Journey

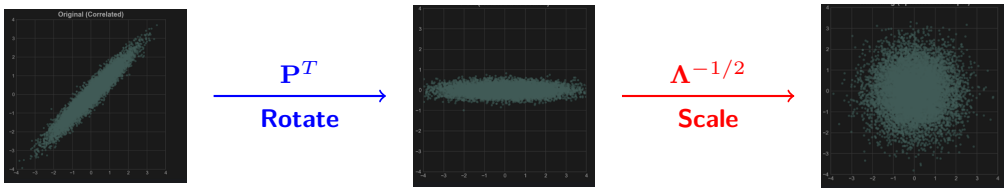


Original
(Correlated)

PCA Space
(Elliptical)

Conceptual Comparison: Visual Geometry (2/2)

From Correlation to Isotropy: A Visual Journey



Original
(Correlated)

PCA Space
(Elliptical)

Whitened
(Spherical)

*Note: Only the **Whitened** space is suitable for Product Kernels.*

Performance Comparison

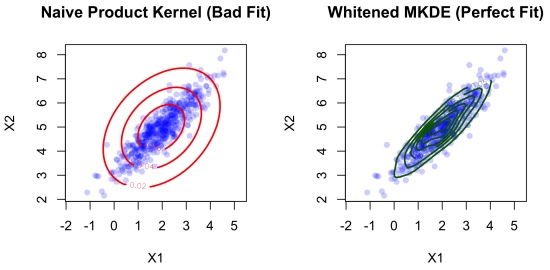


Figure: Naive (Left) vs. Whitened (Right)

Key Observations:

Performance Comparison

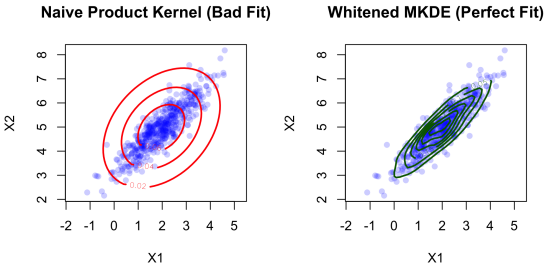


Figure: Naive (Left) vs. Whitened (Right)

Key Observations:

- **Naive Approach: Fail.** Constrained to axis-aligned shapes. Misses diagonal correlations.

Performance Comparison

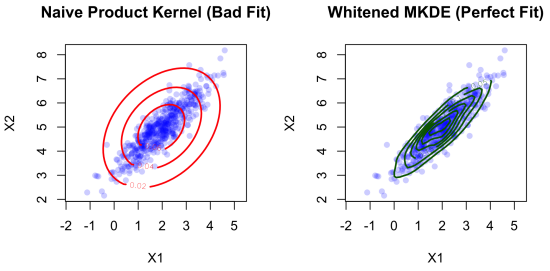


Figure: Naive (Left) vs. Whitened (Right)

Key Observations:

- **Naive Approach: Fail.** Constrained to axis-aligned shapes. Misses diagonal correlations.
- **Whitened MKDE: Success.** Contours adapt perfectly to the data's orientation.

Performance Comparison

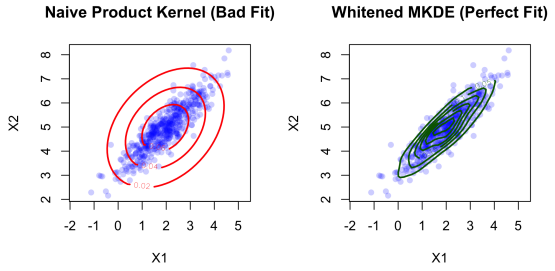


Figure: Naive (Left) vs. Whitened (Right)

Key Observations:

- **Naive Approach: Fail.** Constrained to axis-aligned shapes. Misses diagonal correlations.
- **Whitened MKDE: Success.** Contours adapt perfectly to the data's orientation.

Final Verdict

Whitening enables **Product Kernels** to model complex correlations with $O(n)$ efficiency.

Generalized Estimator (Theory)

Instead of performing manual transformation steps, we can mathematically express the estimator directly using the **Mahalanobis Distance**:

General Multivariate KDE Equation (10.44)

$$\hat{f}(\mathbf{x}) = \frac{|\hat{\mathbf{\Sigma}}|^{-1/2}}{nh^p} \sum_{i=1}^{n} K\left(\frac{(\mathbf{x} - \mathbf{x}_i)^T \hat{\mathbf{\Sigma}}^{-1} (\mathbf{x} - \mathbf{x}_i)}{h} \right) \tag{5}$$

Generalized Estimator (Theory)

Instead of performing manual transformation steps, we can mathematically express the estimator directly using the **Mahalanobis Distance**:

General Multivariate KDE Equation (10.44)

$$\hat{f}(\mathbf{x}) = \frac{|\hat{\mathbf{\Sigma}}|^{-1/2}}{nh^p} \sum_{i=1}^{n} K\left(\frac{(\mathbf{x} - \mathbf{x}_i)^T \hat{\mathbf{\Sigma}}^{-1} (\mathbf{x} - \mathbf{x}_i)}{h} \right) \tag{5}$$

1. Volume Correction:

- The term $|\hat{\mathbf{\Sigma}}|^{-1/2}$ adjusts the density scale.
- Ensures the total probability integrates to 1 after "stretching" space.

Generalized Estimator (Theory)

Instead of performing manual transformation steps, we can mathematically express the estimator directly using the **Mahalanobis Distance**:

General Multivariate KDE Equation (10.44)

$$\hat{f}(\mathbf{x}) = \frac{|\hat{\mathbf{\Sigma}}|^{-1/2}}{nh^p} \sum_{i=1}^{n} K\left(\frac{(\mathbf{x} - \mathbf{x}_i)^T \hat{\mathbf{\Sigma}}^{-1} (\mathbf{x} - \mathbf{x}_i)}{h} \right) \tag{5}$$

1. Volume Correction:

- The term $|\hat{\mathbf{\Sigma}}|^{-1/2}$ adjusts the density scale.
- Ensures the total probability integrates to 1 after "stretching" space.

2. Shape Adaptation:

- Uses **Mahalanobis Distance** instead of Euclidean.
- Captures the correlation structure (orientation) of the data.

Evolution: From Theory to Practice (1/2)

The Whitening Estimator is not a new invention. It is the **General Estimator** (Eq 10.40) where we replace the unknown parameter $\mathbf{H}$ with the **Data's Geometry** $\hat{\mathbf{\Sigma}}$.

1. General Theory

We need an unknown matrix $\mathbf{H}$:

$$K\left(\mathbf{H}^{-1}(\mathbf{x}-\mathbf{X}_i)\right)$$

Problem: Optimizing $\mathbf{H}$ is hard ($O(p^2)$).

2. Whitening Practice

Plug in Sample Covariance $\hat{\mathbf{\Sigma}}$:

$$K\left(\hat{\mathbf{\Sigma}}^{-1}(\mathbf{x}-\mathbf{X}_i)\right)$$

Solution: Use data to fix orientation.

Evolution: From Theory to Practice (1/2)

The Whitening Estimator is not a new invention. It is the **General Estimator** (Eq 10.40) where we replace the unknown parameter $\mathbf{H}$ with the **Data's Geometry** $\hat{\mathbf{\Sigma}}$.

1. General Theory

We need an unknown matrix $\mathbf{H}$:

$$K\left(\mathbf{H}^{-1}(\mathbf{x}-\mathbf{X}_i)\right)$$

Problem: Optimizing $\mathbf{H}$ is hard ($O(p^2)$).

$\Rightarrow$

2. Whitening Practice

Plug in Sample Covariance $\hat{\mathbf{\Sigma}}$:

$$K\left(\hat{\mathbf{\Sigma}}^{-1}(\mathbf{x}-\mathbf{X}_i)\right)$$

Solution: Use data to fix orientation.

The "Lost Sibling" Revealed (2/2)

Let's compare the **General MKDE** with the **Whitening Estimator** side-by-side.

1. The General Form (Eq. 10.40):

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right)$$

$\downarrow$ *Substitute* $\mathbf{H} \approx \hat{\mathbf{\Sigma}}^{1/2}$ *(Data-driven)*

2. The Whitening Form (Eq. 10.44):

$$\hat{f}(\mathbf{x}) = \frac{|\hat{\mathbf{\Sigma}}|^{-1/2}}{nh^p} \sum_{i=1}^{n} K\left(\frac{\hat{\mathbf{\Sigma}}^{-1}(\mathbf{x} - \mathbf{X}_i)}{h}\right)$$

The "Lost Sibling" Revealed (2/2)

Let's compare the **General MKDE** with the **Whitening Estimator** side-by-side.

1. The General Form (Eq. 10.40):

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right)$$

$\downarrow$ *Substitute* $\mathbf{H} \approx \hat{\mathbf{\Sigma}}^{1/2}$ *(Data-driven)*

2. The Whitening Form (Eq. 10.44):

$$\hat{f}(\mathbf{x}) = \frac{|\hat{\mathbf{\Sigma}}|^{-1/2}}{nh^p} \sum_{i=1}^{n} K\left(\frac{\hat{\mathbf{\Sigma}}^{-1}(\mathbf{x} - \mathbf{X}_i)}{h}\right)$$

A. Volume Term

$|\mathbf{H}|$ becomes $|\hat{\mathbf{\Sigma}}|^{1/2}$.
(Normalizes volume change).

The "Lost Sibling" Revealed (2/2)

Let's compare the **General MKDE** with the **Whitening Estimator** side-by-side.

1. The General Form (Eq. 10.40):

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right)$$

$\downarrow$ *Substitute* $\mathbf{H} \approx \hat{\mathbf{\Sigma}}^{1/2}$ *(Data-driven)*

2. The Whitening Form (Eq. 10.44):

$$\hat{f}(\mathbf{x}) = \frac{|\hat{\mathbf{\Sigma}}|^{-1/2}}{nh^p} \sum_{i=1}^{n} K\left(\frac{\hat{\mathbf{\Sigma}}^{-1}(\mathbf{x} - \mathbf{X}_i)}{h}\right)$$

A. Volume Term

$|\mathbf{H}|$ becomes $|\hat{\mathbf{\Sigma}}|^{1/2}$.
(Normalizes volume change).

B. Distance Term

Becomes **Mahalanobis**:
$(\mathbf{x} - \mathbf{X}_i)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{X}_i)$.

Intuition 1: The "Messi" Factor (Variance)

Scenario: La Liga

- **Mean:** 1G/A, 9km.
- **A:** 1G/A, **11km** (Easy).
- **Messi: 3G/A**, 9km (Hard).

Euclidean (Naive Fan)

Calculates geometric distance:

$d_A = \sqrt{0^2 + 2^2} = 2 \quad | \quad d_{\text{Messi}} = \sqrt{2^2 + 0^2} = 2$

Intuition 1: The "Messi" Factor (Variance)

Scenario: La Liga

- **Mean:** 1G/A, 9km.
- **A:** 1G/A, **11km** (Easy).
- **Messi: 3G/A**, 9km (Hard).

Euclidean (Naive Fan)

Calculates geometric distance:

$d_A = \sqrt{0^2 + 2^2} = 2 \quad | \quad d_{\text{Messi}} = \sqrt{2^2 + 0^2} = 2$

Wrong: "Equally impressive."

Intuition 1: The "Messi" Factor (Variance)

Scenario: La Liga

- **Mean:** 1G/A, 9km.
- **A:** 1G/A, **11km** (Easy).
- **Messi: 3G/A**, 9km (Hard).

Euclidean (Naive Fan)

Calculates geometric distance:
$d_A = \sqrt{0^2 + 2^2} = 2 \quad | \quad d_{\text{Messi}} = \sqrt{2^2 + 0^2} = 2$
Wrong: "Equally impressive."

Mahalanobis (The Coach)

Weights by Variance (σ^2): Running variance is high; Scoring variance is low.

Correct: Messi is the Outlier.

Intuition 2: The "All-Rounder" Anomaly (Correlation)

Scenario: Idol Show

- **Rule:** High Vocal $\leftrightarrow$ Avg Dance.
- **A (Orange):** Voc 9, Dance 6 (Normal).
- **B (All-Rounder):** Voc 9, Dance 9.

Euclidean (Blind to Context)

Measures straight line from Average (5,5).

- Sees B as only "slightly" further than A.
- **Fails** to see the skill trade-off pattern.

Intuition 2: The "All-Rounder" Anomaly (Correlation)

Scenario: Idol Show

- **Rule:** High Vocal $\leftrightarrow$ Avg Dance.
- **A (Orange):** Voc 9, Dance 6 (Normal).
- **B (All-Rounder):** Voc 9, Dance 9.

Euclidean (Blind to Context)

Measures straight line from Average (5,5).

- Sees B as only "slightly" further than A.
- **Fails** to see the skill trade-off pattern.

Mahalanobis (Structure Aware)

Uses **Correlation Matrix** (Ellipse shape).

- **A (Orange):** On the ellipse edge (Expected).
- **B (All-Rounder):** Far outside (The Unicorn).

Conclusion: B is a huge anomaly.

Appendix A: Beyond Gaussian (Fourier Approach)

Motivation: Standard KDE struggles with bandwidth matrices ($\mathbf{H}$) in high dimensions. Is there a better way?

The Fourier Integral Theorem (Identity)

Any square-integrable function $f(\mathbf{x})$ can be reconstructed **exactly** via:

$$f(\mathbf{x}) = \lim_{R \to \infty} \frac{1}{\pi^d} \int_{\mathbb{R}^d} \prod_{j=1}^{d} \frac{\sin(R(x_j - t_j))}{x_j - t_j} f(\mathbf{t}) d\mathbf{t} \tag{6}$$

The Estimator (Monte Carlo approx):

$$\hat{f}_{n,R}(\mathbf{x}) = \frac{1}{n\pi^d} \sum_{i=1}^{n} \prod_{j=1}^{d} \underbrace{\frac{\sin(R(x_j - X_{ij}))}{x_j - X_{ij}}}_{\text{Sinc Kernel}}$$

Key Difference

It uses the **Sinc Kernel** instead of Gaussian.

$\text{sinc}(u) = \frac{\sin(u)}{u}$

Appendix B: The "Magic" of Fourier KDE

The Big Question: How does it handle correlation without a matrix $\mathbf{H}$?

1. Gaussian KDE (Traditional)

To capture correlation, we **MUST** estimate a full matrix $\mathbf{H}$ (or $\mathbf{\Sigma}^{-1}$).

- $\prod K(x_j)$ (Diagonal) $\rightarrow$ Fails (Axis-aligned).
- Full $\mathbf{H} \rightarrow$ Expensive ($O(d^2)$).

Appendix B: The "Magic" of Fourier KDE

The Big Question: How does it handle correlation without a matrix $\mathbf{H}$?

1. Gaussian KDE (Traditional)

To capture correlation, we **MUST** estimate a full matrix $\mathbf{H}$ (or $\mathbf{\Sigma}^{-1}$).

- $\prod K(x_j)$ (Diagonal) $\rightarrow$ Fails (Axis-aligned).
- Full $\mathbf{H} \rightarrow$ Expensive ($O(d^2)$).

2. Fourier KDE (Advanced)

Theorem: A simple product of 1D Sinc kernels **automatically** recovers the joint density structure.

- **No Matrix needed!** Just a scalar R.
- Captures rotation/correlation natively via the Fourier domain.

Appendix B: The "Magic" of Fourier KDE

The Big Question: How does it handle correlation without a matrix $\mathbf{H}$?

1. Gaussian KDE (Traditional)

To capture correlation, we **MUST** estimate a full matrix $\mathbf{H}$ (or $\mathbf{\Sigma}^{-1}$).

- $\prod K(x_j)$ (Diagonal) $\rightarrow$ Fails (Axis-aligned).
- Full $\mathbf{H} \rightarrow$ Expensive ($O(d^2)$).

2. Fourier KDE (Advanced)

Theorem: A simple product of 1D Sinc kernels **automatically** recovers the joint density structure.

- **No Matrix needed!** Just a scalar R.
- Captures rotation/correlation natively via the Fourier domain.

Verdict: Fourier KDE bypasses the "Bandwidth Matrix Selection" headache entirely.

Appendix C: Performance Superiority

Why should we care? Because it converges faster.

Convergence Rate (MISE):

- **Standard KDE:**

$$O(n^{-4/(4+d)})$$

(Suffer from Curse of Dimensionality)

- **Fourier KDE (Supersmooth):**

$$O(n^{-1}(\log n)^k)$$

(Almost parametric rate $1/n$!)

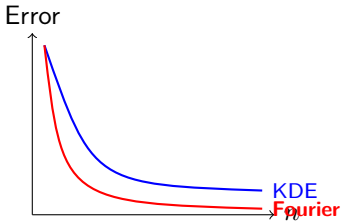


Figure: Faster error decay

Source: Ho, N., & Walker, S. G. (2021). Multivariate Smoothing via the Fourier Integral Theorem.

Thank You for Your Attention!

Q & A

We are open to any questions or feedback.

Project Source Code & Slides:

https://github.com/khang3004/Multivariate_KernelDensity_Estimators.git

Contact: 21C0103580@student.hcmus.edu.vn