Non-linear Independent Component Analysis with Diffusion Maps

Chuxiangbo Wang

May 3, 2023

Contents

1 Introduction

Independent Component Analysis (ICA) is a technique for separating mixed signals into their original sources, assuming that the sources are statistically independent. The algorithm involves preprocessing the data, initializing an unmixing matrix, optimizing it using various methods, stopping when a criterion is met, and estimating the sources. ICA has several advantages, including blind source separation, robustness to noise, and effectiveness with non-Gaussian sources. However, it also has some drawbacks, such as ambiguity in source ordering and scaling, reliance on the independence assumption, computational complexity for large data, and sensitivity to initialization.

The main idea is that suppose we possess two sets of unknown data, x_1 and x_2. After being processed through unknown non-linear functions of independent stochastic Itô processes, we obtain the observed data as y_1 and y_2, which are known to us. Our goal is to use y_1 and y_2 to reconstruct the original unknown data, x_1 and x_2. To do this, we apply Independent Component Analysis (ICA) and Diffusion Dap to the observed data, and then use the first two eigenvectors, ϕ_1 and ϕ_2, to recover x_1 and x_2, as shown in Figure 1.

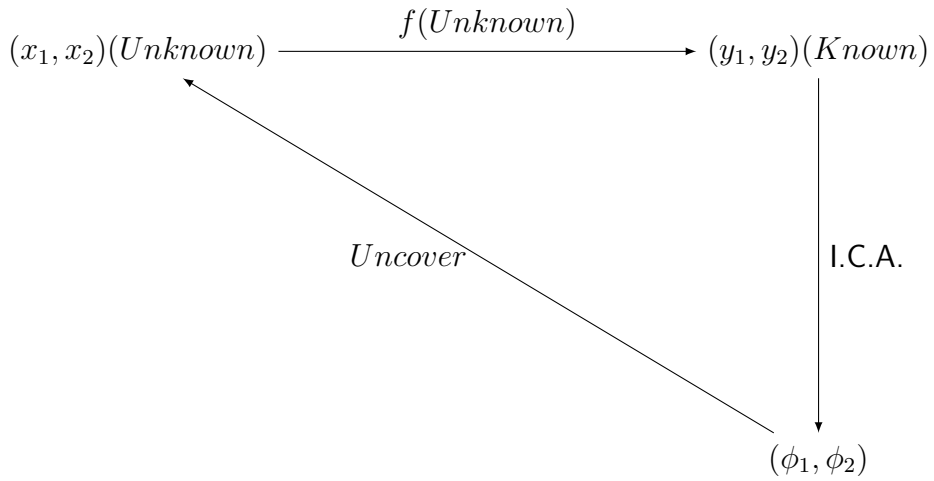

Figure 1: Illustration of the process from observable space (y_1, y_2) to independent components (ϕ_1, ϕ_2)

In Section 2 and 3, we will present definitions and significant theoretical results that are closely related to and utilized in our project. In Section 4, we offer a numerical example to demonstrate that by using the weight matrix created with the local Jacobian

$$W_{ij} = \exp\left(-\frac{\left\|J^{-1}(y^{(i)})(y^{(j)} - y^{(i)})\right\|^2 + \left\|J^{-1}(y^{(j)})((y^{(j)}) - (y^{(i)}))\right\|^2}{4\epsilon}\right).$$

we can recover the data x_1, x_2 in unknown space X. Finally, in Section 4, we will explain that if we lack knowledge of the nonlinear map 'f', we will apply a simulated Itô process to each point in the unobservable space. We will then calculate our

covariance matrix based on corresponding changes in the observable space 'Y' to construct the weight matrix

$$W_{ij} = \exp\left(-\frac{\delta^2}{d+2} * \frac{(y^{(j)} - y^{(i)})^T \left[C^\dagger + C^\dagger\right](y^{(j)} - y^{(i)})}{4\epsilon}\right).$$

for the diffusion map. This will show that the unobservable space is well-revealed by the method.

All the Matlab Code for numerical experiments can be accessed in OneDrive.

2 Background Knowledge

2.1 Independent Component Analysis

A popular algorithm for performing ICA is FastICA, which was introduced by Aapo Hyvärinen and Erkki Oja in their paper titled "Independent Component Analysis: Algorithms and Applications". The paper was published in the journal Neural Networks, volume 13, issue 4-5, pages 411-430, in the year 2000. The FastICA algorithm is discussed in detail in Section 6 of the paper, titled "FastICA Algorithm."

2.2 Itô process

The Itô process is a type of stochastic process, defined as a continuous-time process with a continuous trajectory that is adapted to a Brownian motion (or Wiener process) W_t. It can be written as the sum of a Riemann-Stieltjes integral with respect to the Wiener process and a Lebesgue-Stieltjes integral with respect to time t. Mathematically, it is defined as:

$$X_t = X_0 + \int_0^t \sigma_s \, dW_s + \int_0^t \mu_s \, ds \tag{1}$$

where:

- X_0 is the initial value of the process,
- σ_s is the diffusion coefficient, which can be a function of the process value X_s and time s,
- μ_s is the drift coefficient, which can also be a function of the process value X_s and time s,
- W_s is a Wiener process or Brownian motion, and
- the first integral is a Riemann-Stieltjes integral with respect to the Wiener process, while the second integral is a Lebesgue-Stieltjes integral with respect to time.

2.3 Diffusion Process

For Diffusion Map, a diffusion process is usually given by: We define the diffusion matrix L (it is also a version of graph Laplacian matrix):

$$L_{i,j} = k(x_i, x_j) \tag{2}$$

We then define the new kernel $L_{i,j}(\alpha)$:

$$L_{i,j}^{(\alpha)} = k^{(\alpha)}(x_i, x_j) = \frac{L_{i,j}}{(d(x_i)d(x_j))^{\alpha}} \tag{3}$$

or equivalently,

$$L^{(\alpha)} = D^{-\alpha} L D^{-\alpha} \tag{4}$$

where D is a diagonal matrix and

$$D_{i,i} = \sum_j L_{i,j}. \tag{5}$$

We apply the graph Laplacian normalization to this new kernel:

$$M = (D^{(\alpha)})^{-1} L^{(\alpha)}, \tag{6}$$

where $D^{(\alpha)}$ is a diagonal matrix and

$$D_{i,i}^{(\alpha)} = \sum_j L_{i,j}^{(\alpha)}. \tag{7}$$

Depending on the choice of α, the diffusion map converges to different representations of the data:

- For $\alpha = 0$, the diffusion map focuses on local structure, with the distances in the map approximating the geodesic distances on the data manifold.

- For $\alpha = 1/2$, the diffusion map approximates the Fokker-Planck diffusion, which balances local and global structure.

2.4 Fokker-Planck Operator

The Fokker-Planck Operator has form:

$$\frac{\partial P(x,t)}{\partial t} = -\nabla \cdot (b(x,t)P(x,t)) + \frac{1}{2}\nabla^2 (D(x,t)P(x,t)), \tag{8}$$

where:

- $P(x,t)$ is the probability density function,
- t is the time,
- $b(x,t)$ is the drift term,
- $D(x,t)$ is the diffusion coefficient,

3 Theorems and Jacobian-Covariance Relation

We have so far discussed the mathematical framework and the key concepts involved in our study. To further investigate, we will now introduce two theorems. The first theorem describes the relationship between the usage of the Jacobian in the weight matrix and the convergence of the Graph Laplacian to the Fokker-Planck operator. The second theorem provides the idea that with the ito process, the origional data can be uncovered by weight matrix using the jacobian, which can be approximated by the covariance matrix.

Theorem 1 *The Graph Laplacian converges to the Fokker-Planck operator when* $W_{ij} = \exp\left(-\frac{\left\|J^{-1}(y^{(i)})(y^{(j)}-y^{(i)})\right\|^2+\left\|J^{-1}(y^{(j)})((y^{(j)})-(y^{(i)}))\right\|^2}{4\epsilon}\right)$.

After establishing the convergence of the Graph Laplacian to the Fokker-Planck operator in Theorem 1, we now proceed to explore the relationship between the Jacobian and the covariance matrix under the assumtion of the Ito Process.

Theorem 2

$$J^{-1}(y)(\eta - y)^2 = \|\xi - x\|^2 + O(\|\xi - x\|^3)$$

where x,ξ be two points in the original space X (unknown) and y = f(x),η = f(ξ) be their mapping to the observable space Y (known).

The Theorem 2 shows that the distance in the unobserved space can be approximated by the Jacobian, which is essential for us to construct the weight matrix in order to recover the unknown data. The proof idea is that let x, ξ be two points in the original space X and $y = f(x), \eta = f(\xi)$ their mapping to the observable space Y. Let $g : Y \to X$ be the inverse mapping of $f : X \to Y$, that is, $g(f(x)) = x$ and $f(g(y)) = y$, $\forall x \in X$, $\forall y \in Y$. Expanding the functions $x = g(y)$ in a Taylor series at the point y gives the approximation:

$$\xi = x + g_j(y)(\eta - y) + \frac{1}{2}g_{jk}(y)(\eta - y)(\eta - y) + O|\eta - y|^3, \tag{9}$$

the expansion at the point η is similar.

Under the assumption of the Ito process and our knowledge about $JJ^T(x_1^{(i)}, x_2^{(i)}) = \frac{C^{(i)}}{\Delta t}, \quad i = 1, \ldots, N.$, where C is the covariance matrix, we can derive a second-order symmetric approximation for the Euclidean distance in the original space X by using the weight matrix constructed with the covariance matrix (Section 5).

4 Numerical Example

In this section, we assume that the non-linear map f is known to us, and we will utilize f to directly calculate the local Jacobian. The key idea is to demonstrate that, as opposed to the traditional diffusion map, which only approximates distances in the observable space, the diffusion map constructed using the local Jacobian can recover distances in the unobservable space. This ultimately aids in uncovering the unknowns.

To illustrate this point, we conduct a numerical experiment. We generate $N = 2000$ random points, uniformly distributed within the unit square, represented as (x_1^i, x_2^i), where $i = 1, \ldots, N$. These points are non-linearly mapped according to the following equations:

$$\begin{aligned} y_1 &= x_1 + x_2^3 \\ y_2 &= x_2 - x_1^3 \end{aligned} \tag{10}$$

This results in obtaining (y_1^i, y_2^i), which maps the unit square into a mushroom-shaped figure, as depicted in Figure 2.

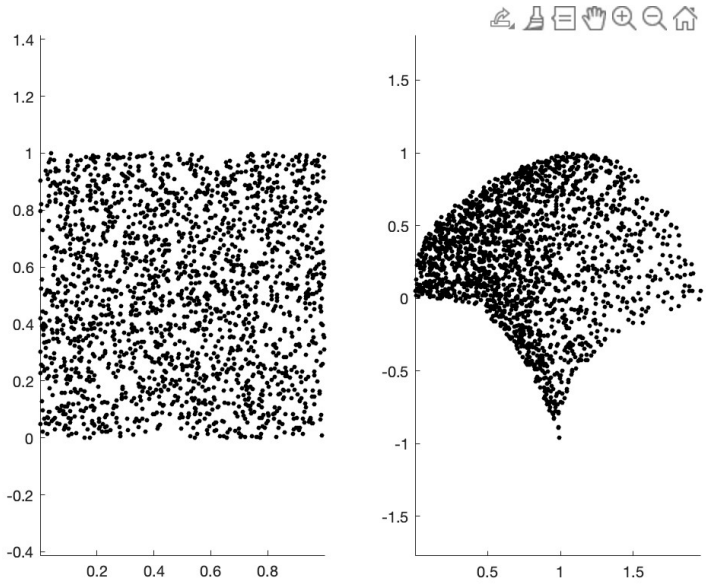

Figure 2: Non-linear mapping of the generated points

4.1 Eigenvalues

The row stochastic matrix $D^{-1}W$ is approximately equal to $\exp\left(-\frac{\epsilon}{2}L\right)$, where L is the Fokker-Planck operator (By Theorem 2.3.1). Since our manifold is the unit square with uniform density, L is just the Laplacian of the unit square with eigenvalues $\mu_{n,m} = \pi^2(n^2 + m^2)$. Figure 3 and 4 emphasis the comparison between the numerical eigenvalues and analytical eigenvalues.

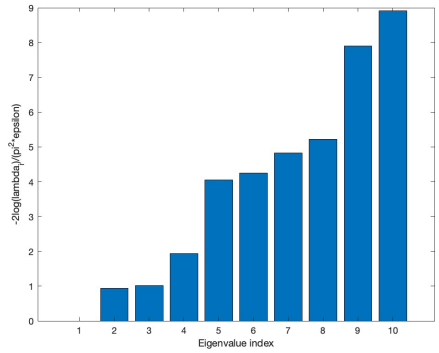

Figure 3: Numerical Eigenvalues

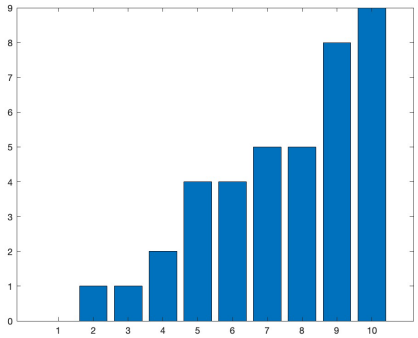

Figure 4: Analytical Eigenvalues

4.2 Anisotropic Diffusion vs Graph Laplacian

In Figure 5, we demonstrate the application of the classic diffusion map to the data, followed by the calculation of the second and the third eigenvectors ϕ_1 and ϕ_2. By coloring x_1 and x_2 using ϕ_1 and ϕ_2, and vice versa, we can clearly observe that the two eigenvectors preserve the distinctive mushroom shape.

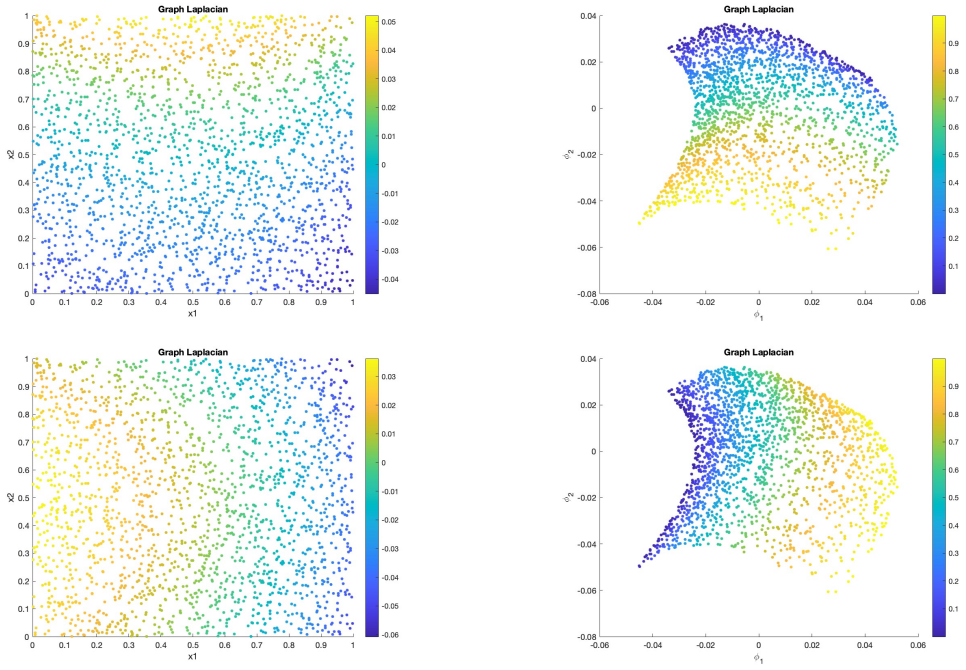

Figure 5: Graph Laplacian

We construct the weight matrix as

$$W_{ij} = \exp\left(-\frac{\left\|J^{-1}(y^{(i)})(y^{(j)} - y^{(i)})\right\|^2 + \left\|J^{-1}(y^{(j)})(y^{(j)} - y^{(i)})\right\|^2}{4\epsilon}\right), \tag{11}$$

and then apply the diffusion map. The result is displayed in Figure 6, where the square shape is recovered, but the density is not preserved.

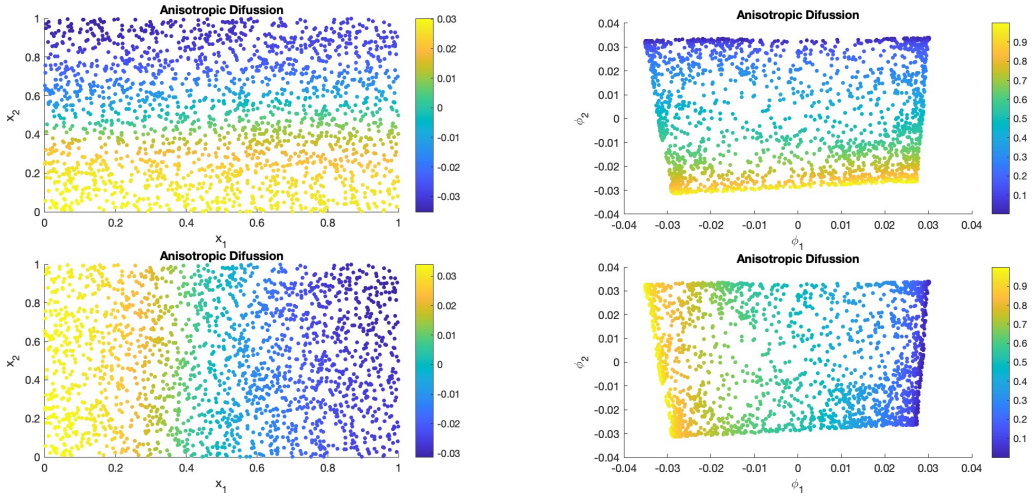

Figure 6: Anisotropic Diffusion

5 Using Covariance Matrix to Approximate the Jacobian

When the function f is unknown, we simulate the Itô process on unobserved variables and then use the changes in the observable space to construct the covariance matrix. We will use the covariance matrix to approximate the Jacobian by calculating the sample covariance $C^{(i)}$ of the point cloud and estimate the local Jacobian $J(x_1^{(i)}, x_2^{(i)})$ using:

$$JJ^T(x_1^{(i)}, x_2^{(i)}) = \frac{C^{(i)}}{\Delta t}, \quad i = 1, \ldots, N. \tag{12}$$

We construct the weight matrix W, using the covariance matrices as

$$W_{ij} = \exp\left(-\frac{\delta^2}{d+2} * \frac{(y^{(j)} - y^{(i)})^T \left[C_{(i)}^{\dagger} + C_{(i)}^{\dagger} \right] (y^{(j)} - y^{(i)})}{4\epsilon} \right).$$

where $C_{(i)}^{\dagger}$ is the pesudo inverse of $C^{(i)}$

5.1 Numerical Example

With the same setting as the numerical example in Section 3, we now assume that the non-linear map f is unknown. For each point $x^{(i)}$, we simulate the Itô process with $\Delta t = 0.001$ and generate a local burst of 1000 simulated points. We then observe the changes in the observable space and use this information to construct the covariance matrix for each point $y^{(i)}$. The pseudo-inverse of the covariance matrix is calculated using SVD. Finally, we compute the weight matrix with $\epsilon = 0.005$.

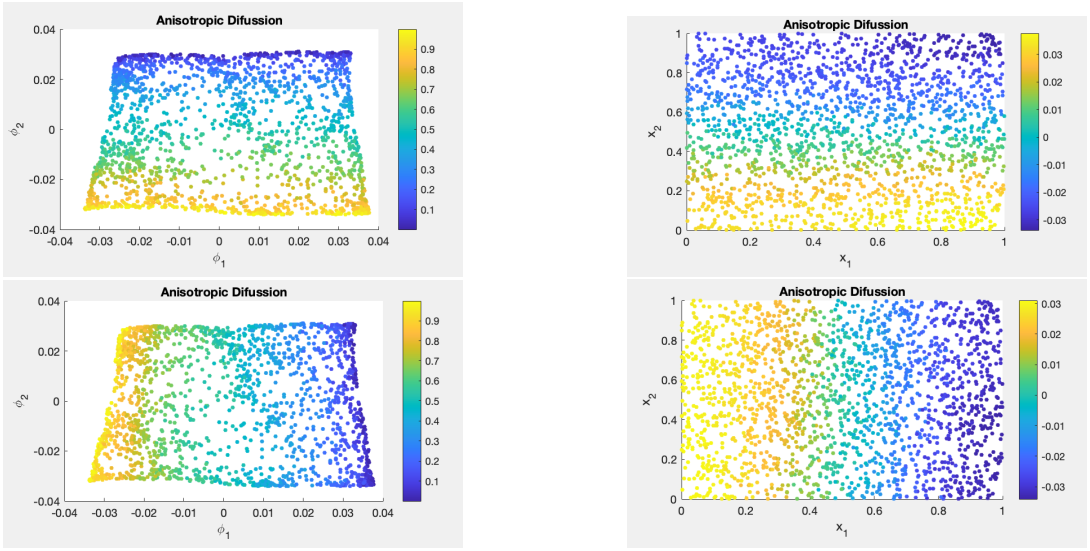

Figure 7: Data Recovered using Covariance Matrix

As evident from Figure 7, the square shape of x_1 and x_2 is successfully recovered by ϕ_1 and ϕ_2.

The dimensionality of the data can be adjusted when taking the conjugate inverse of the covariance matrix. For instance, if we have m-dimensional data embedded in an n-dimensional manifold, where $m < n$, after calculating the covariance matrix, we only take the first m principal components when taking the conjugate inverse. We then apply the remaining procedures to uncover the data.

References

[1] Amit Singer and Ronald R. Coifman. Non-linear independent component analysis with diffusion maps. *Applied and Computational Harmonic Analysis*, 2008.

[2] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[3] Wikipedia contributors. Diffusion map. Wikipedia, The Free Encyclopedia, 2021. URL: `https://en.wikipedia.org/wiki/Diffusion_map`

[4] Wikipedia contributors. Itô process. Wikipedia, The Free Encyclopedia, 2021. URL: `https://en.wikipedia.org/wiki/It_process`

[5] Wikipedia contributors. Fokker-Planck equation. Wikipedia, The Free Encyclopedia, 2021. URL: `https://en.wikipedia.org/wiki/Fokker-Planck_equation`