

Spectral Clustering of Spatiotemporal Datasets

A thesis

submitted to

Indian Institute of Science Education and Research Pune in partial
fulfillment of the requirements for the BS-MS dual degree programme

by

Augastya Srivastava



Indian Institute of Science Education and Research Pune
Dr Homi Bhabha Road,
Pashan, Pune 411008, INDIA

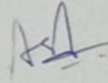
December 2024

Supervisor: Dr. Amit Apte (IISER Pune)
Expert: Dr. Moumanti Podder (IISER Pune)

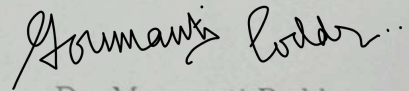
All rights reserved

Certificate

This is to certify that this dissertation entitled **Spectral Clustering of Spatiotemporal Datasets** towards the partial fulfillment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Augastya Srivastava at Indian Institute of Science Education and Research Pune under the supervision of Professor Amit Apte, Department of Data Science, during the academic year 2023-2024.



Dr. Amit Apte



Dr. Moumanti Podder

Committee:

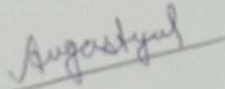
Supervisor: Dr. Amit Apte

Expert: Dr. Moumanti Podder

This thesis is dedicated to
my mother, Suman Srivastava.

Declaration

I hereby declare that the matter embodied in the report entitled " Spectral Clustering of Spatiotemporal Datasets " are the results of the work carried out by me at the Department of Mathematics, Indian Institute of Science Education & Research (IISER) Pune, under the supervision of Dr. Amit Apte, and the same has not been submitted elsewhere for any other degree. Wherever others contribute, every effort is made to indicate this clearly, with due reference to the literature and acknowledgment of collaborative research and discussions.



Augastya Srivastava
20191201

Abstract

The aim of this project is to use the spectral clustering algorithm to detect important geological and climate features by utilizing the properties of the graph Laplacian. Spectral clustering uses similarity measures between data points to construct the graph Laplacian and then finds clusters using the eigenvectors corresponding to its largest eigenvalues.

Our work has found the relationships between connectivity parameters that are used to define pairwise similarity values between data points and the eigenvalues of the graph Laplacian. These relationships are then used to cluster spatiotemporal datasets, including some high-dimensional datasets.

We have also tried to account for the phase differences that exist between two otherwise similar high-dimensional time series by using Dynamic Time Warping (DTW) and Uniform Manifold Approximation and Projection (UMAP).

The relationships we find between the connectivity parameters and the graph Laplacian's eigenvalues can be used to study eigengaps which are important to perform cluster analysis and detect the number of clusters that can possibly be obtained without having an estimate beforehand.

Acknowledgements

I would like to thank my supervisor, Dr Amit Apte, for providing his guidance and insights at every step of the project. He also provided me with the resources necessary to run the computationally expensive algorithms required for this project. I am grateful for his patience and the valuable lessons he taught me.

I am also grateful to Dr Mihir Hasabnis for his collaborative work on this project and for providing some unique ideas that helped the project's progress.

I would also like to thank my expert member, Dr Moumanti Podder, for her advice and suggestions. It was during one of her courses that I developed a great interest in Statistics. If it wasn't for her lessons and support, I would never have entered the world of Data Science.

I am thankful to the Department of Mathematics, IISER Pune, for providing me with this great opportunity to do research.

I would like to thank my friends and family for their emotional support. And I thank God for His love and for my spiritual growth throughout the project.

Contents

1	Introduction	17
1.1	Spectral clustering	18
1.2	Earlier applications of spectral clustering	19
1.2.1	Speech separation	19
1.2.2	Image clustering	20
1.2.3	Community detection	20
1.3	The datasets	20
1.3.1	Events	20
1.3.2	Time series	20
1.3.3	Maps	21
2	Methodology	22
2.1	The Spectral Clustering Algorithm	22
2.2	Why does it work?	23
2.2.1	The ideal case	24
2.2.2	The general case	25
2.3	Constructing a graph	26
2.4	Choosing the parameter σ	29
2.5	Dynamic time warping (DTW)	33
2.5.1	Examples	35
2.6	UMAP	37
2.6.1	A correlation algorithm	38

3	Application and Results	41
3.1	Earthquake dataset	41
3.2	OLR dataset	45
3.2.1	Time series dataset	45
3.2.2	Maps dataset	51
3.3	Rainfall data	52
3.4	OLR vs Rainfall datasets	53
3.4.1	Histogram of pairwise distances	53
3.4.2	UMAP	53
4	Conclusion	57
4.1	Parameter σ	57
4.2	Spectral clustering has detected geological and climate features.	58
4.3	OLR time-series data could not be clustered.	58
4.4	Future Work	58
	References	60

List of Figures

2.1	Eigengaps of the ideal Laplacian with 3 blocks.	27
2.2	Eigengaps of Laplacian with 3 blocks for perturbed similarity matrices, with varying standard deviation.	27
2.3	Clustering concentric annuli, for $\sigma = 0.5, 1$	30
2.4	Clustering concentric annuli, for $\sigma = 1.5, 2$	31
2.5	Clustering 5 balls with randomly added noise.	32
2.6	Top 100 eigenvalues of Laplacian for uniform points with no clusters.	33
2.7	DTW optimal warping paths for different symmetric slope constraints.	35
2.8	Application of UMAP on a synthetic dataset.	38
2.9	Result of the correlation algorithm on synthetic dataset.	39
3.1	Eigenvalues of the Laplacian for the earthquake dataset	42
3.2	Earthquake clusters over Asia-Pacific, $\sigma = 468, k = 17$	43
3.3	Earthquake clusters over Asia-Pacific, $\sigma = 468, k = 26$	44
3.4	L1 Norm distance matrices for (a) Hourly time-series and (b) Time-series averaged over every 12 hours. And (c) The difference between the two matrices divided by the first matrix.	46
3.5	Eigenvalues of graph Laplacian, for different values of σ , OLR time-series dataset.	48
3.6	Top 100 eigenvalues of the Laplacian matrix for $\sigma = 2\sigma_0$, OLR time-series data.	49

3.7	The 19 clusters of OLR time-series data, using l1-norm distance function, and $\sigma = 2\sigma_0$, shown in three different different plots for clarity. .	50
3.8	Clustering of OLR 2010 maps to detect monsoon period.	51
3.9	Top 100 eigenvalues of the Laplacian for the rainfall time series data .	52
3.10	Rainfall time series data clusters, using L1 norm. The scatter plot show the number of data points in each cluster.	54
3.11	Histograms of pairwise distances for (a) OLR time-series data, L1 norm distance function, (b) Rainfall time-series data, L1 norm distance function, (c) Concentric annuli, Euclidean distance, and (d) Synthetic data containing 5 well-separated balls each containing uniformly distributed points, Euclidean distance.	55
3.12	UMAP projections on \mathbb{R}^2	56

Chapter 1

Introduction

In this project, we have used unsupervised machine-learning algorithms to detect geological structures and climate zones through various low and high-dimensional spatial or spatio-temporal datasets. The main technique we have used is spectral clustering. Although the algorithms are pretty straightforward, the choice of multiple parameters involved in these algorithms is often tricky, and we have spent a significant part of the project on that. In order to choose parameters, we used some methods from Luxborg's 2007 tutorial on spectral clustering [1] while also trying to find new ways. We manipulate the Laplacian matrix and try to find relationships between its eigenvalues and the values of the connectivity parameters. Once we establish a relationship, we use it to detect geographical features without using any additional information from Earth and climate science.

Spatio-temporal datasets are often of very high dimensions. Both spectral clustering and UMAP use spectral graph theory to project these high-dimensional datasets into lower-dimensional space before doing any clustering. Therefore, we first create some low-dimensional synthetic datasets and then generalize our findings to spatio-temporal datasets.

Another challenge to this project was the choice of an appropriate similarity measure

for clustering such datasets. We mostly used L1-norm distance function and then converted it into a similarity measure by using Gaussian function 2.1. But for some datasets, the phase difference between otherwise similar data points was too much to be ignored. Therefore, we tried DTW distance function as well.

We begin with an introduction to spectral clustering and spatio-temporal datasets and then discuss some of the earlier applications of spectral clustering. But first, we need to understand the Laplacian matrix. After we have seen what spectral clustering is, we will see how it has been used in the past to cluster real-life datasets in section 1.2. Then we look at the description of the datasets that we will use for our project in section 1.3.

1.1 Spectral clustering

Spectral clustering uses spectral graph theory to cluster a given dataset. Let $S = \{x_1, x_2, \dots, x_n\}$ be a set of data points in \mathbb{R}^d . Our goal is to find a partition of S containing sets $C_i = \{x_{i_1}, x_{i_2}, \dots\}$, $i = 1, 2, \dots, k$ such that the inter-cluster similarity is minimized and intra-cluster similarity is maximized while also making sure the clusters are not too small. This problem of balancing the cluster ‘size’ becomes NP-hard. See Wagner and Wagner (1993) [2] for more discussions.

So, instead of finding perfect clusters, we try to approximate our clusters using the spectral algorithm in which we convert the given dataset into a graph and find its clusters using eigenvectors of the graph Laplacian [3]. This is a density-based algorithm as opposed to KMeans, and it also has a way of determining the number of clusters that can be obtained. The difficulty is converting the dataset into a graph and determining what similarity measure we should use between data points. The full algorithm is discussed in chapter 2.

Similarity measures often depend on what are called the ‘connectivity parameters.’ A significant part of the project was the selection of such parameters. The crucial part of judging the choices of our parameters was to analyze the eigenvalues of

the graph Laplacian for reasons discussed in chapter 2. The number of clusters is determined by the eigengap in the spectral plot of the graph Laplacian. If there is an eigengap between k th and $(k + 1)$ th eigenvalues, we try to find k clusters in the graph. Variation of the connectivity parameters can give us a significant eigengap for a dataset, but it is not always guaranteed, as shown in chapter 3.

To see if the eigengap is stable, we plotted the eigengaps against the size of the dataset, first for the ideal matrix and then for the perturbed matrix. The ideal matrix is the Laplacian of a graph where points from different clusters are infinitely far apart. Thus, the ideal matrix is a block matrix. Then, we perturb it by connecting points from different clusters to each other.

Once the number of clusters is determined, the next step is to project the dataset onto a k -dimensional space where k is the number of clusters. This method is important for both spectral clustering and UMAP (for UMAP, the dimension of the projected space is determined by the manifold embedded in the original dataspace). The manner in which these projections are made is discussed in chapter 2.

1.2 Earlier applications of spectral clustering

1.2.1 Speech separation

One of the most common applications of spectral clustering is to identify words or speakers. It has been used to separate speakers' speeches recorded by a single microphone by Bach and Jordan (2006) [4]. Their technique differs slightly from ours as they have first generated training samples, but we do not use this approach.

Speeches can be treated as time series, but they often have phase differences. A different similarity measure is sometimes used to compare them correctly. It is called dynamic time warping, and its methodology is discussed in chapter 2. However, this metric is very time-consuming and unsuitable for large datasets.

1.2.2 Image clustering

Spectral clustering has proven effective in clustering images of digits drawn with different hand-writings [5]. Therefore, it might be used to separate images of different objects without requiring a training sample.

1.2.3 Community detection

Real-life communities can be detected with spectral clustering if the right attributes are chosen to calculate the similarity matrices. This has been done for diving students according to their academic performances by Mouden *et al* (2019) [6].

1.3 The datasets

We have used four different types of datasets in this project. The first three are spatiotemporal datasets, which are described using geographical coordinates and/or time coordinates along with any other attribute we might want to include while finding similarity values between data points [7].

1.3.1 Events

An event dataset is a low-dimensional dataset that has spatial and temporal coordinates only. An example of this is an earthquake dataset [8] that we have used in our project. The dataset contains spatial coordinates and the dates of 782 earthquakes that occurred from 2001 to 2022, with a magnitude of at least 6.5. It is dense in space but very sparse in time; therefore, clustering the dataset in both dimensions is difficult.

1.3.2 Time series

A time series contains numerical values that a given variable takes over a selected time interval. Each time series is associated with geographical coordinates such that when we cluster the dataset, it is technically a clustering of locations.

Time series are often high-dimensional datasets; therefore, finding lower-dimensional manifolds through trial and error is difficult. We also need to find good similarity measures to compare two time series as we need to consider the existing phase difference. If we wish to shorten the length of a time series by averaging, we need to ensure the adjacency matrix is not changed drastically.

We have used two such datasets in this project. The first dataset contains the time series of outgoing longwave radiation values over South Asia. It is obtained from ERA5 hourly data on single levels available on climate.copernicus.eu. We have OLR values at 32,361 locations between 60°E to 100°E and -10°N to 40°N after every 0.25°. It is obtained through reanalysis, which combines model data with real observations using laws of physics. The values measure the intensity of longwave radiation emitted from Earth's atmosphere. The lower the absolute value, the more cloudy it is at that location.

The second time series dataset is daily rainfall values interpolated into grids of 1°×1° over India created by Rajeevan *et al* (2006) [9].

1.3.3 Maps

The maps are snapshots of measurements referenced by geographical coordinates at any given time. We have used the OLR maps dataset for clustering. This is a 32,361-dimensional dataset containing hourly OLR values over South Asia.

Chapter 2

Methodology

This chapter explains the theory behind the methods that we are going to use for our datasets. It also has a description of all the algorithms we apply and why we apply them.

We begin with the main spectral clustering algorithm that we are going to follow throughout the project. The next section describes why the algorithm should work and the results of some of the experiments we performed to test the theory behind the algorithm. After that, we describe how we can determine the values of the connectivity parameters and how our choices determine the clustering results using some experiments on synthetic datasets.

Finally, we end this chapter after discussing the theory behind UMAP and DTW and how to manipulate their algorithms to get good clustering results.

2.1 The Spectral Clustering Algorithm

The algorithm for spectral clustering uses special properties of the Laplacian matrix derived from spectral graph theory [10]. The algorithm below was presented by Andrew *et al* (2001) [11].

Let us suppose we have a set of n data points $S = \{x_1, x_2, \dots, x_n\}$ in \mathbb{R}^l , and we would like to cluster them in k subsets. The number k can be determined by looking at the eigenvalues of the Laplacian, which we will discuss later.

1. Form a similarity matrix (also known as affinity matrix) $A \in \mathbb{R}^{n \times n}$ where $A_{ij} = s(i, j)$ when $i \neq j$, and $A_{ii} = 0$. $s(i, j)$ is a similarity function that takes values between 0 and 1. Andrew *et al* (2001) use the Gaussian similarity function given as,

$$s(i, j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right). \quad (2.1)$$

We will also use some other methods to find similarities between data points as we deal with different types of datasets.

2. Create a diagonal matrix D such that D_{ii} is the sum of the i^{th} row of A .
3. Construct the Laplacian matrix $L = D^{-1/2}AD^{-1/2}$. There are other methods of constructing a Laplacian, as mentioned in Luxborg (2007) [1].
4. Find v_1, v_2, \dots, v_k the eigenvectors of L corresponding to the top eigenvalues of L . They are chosen to be orthonormal to each other in case of repeated eigenvalues. Now form a matrix $V = [v_1 \ v_2 \ \dots \ v_k] \in \mathbb{R}^{n \times k}$.
5. Normalize each row of V so that the rows have unit length. We will call the normalized matrix Y .
6. Consider each row of Y as a data point in \mathbb{R}^k and cluster them into k clusters using K-means algorithm [12].
7. Think of the i^{th} row of Y as corresponding to x_i and cluster the original data points as the rows of Y have been clustered.

2.2 Why does it work?

In order to understand why the spectral algorithm works, Andrew *et al* [2001] first consider an ideal dataset where all the clusters are infinitely far apart from each

other and then generalize the results using matrix perturbation theory [13].

2.2.1 The ideal case

In the ideal case, where the points from different clusters are all far apart, the similarity matrix \hat{A} and the Laplacian \hat{L} are both block diagonal matrices. Let us assume that there are n data points and 3 clusters in the dataset. Then, the matrices will look like the following:

$$\hat{A} = \begin{bmatrix} A^{(11)} & 0 & 0 \\ 0 & A^{(22)} & 0 \\ 0 & 0 & A^{(33)} \end{bmatrix}; \quad \hat{L} = \begin{bmatrix} L^{(11)} & 0 & 0 \\ 0 & L^{(22)} & 0 \\ 0 & 0 & L^{(33)} \end{bmatrix}. \quad (2.2)$$

Using simple linear algebra, it can be shown that $\hat{L}^{(ii)}$ has a strictly positive principal eigenvector with eigenvalue 1, and the second eigenvalue is strictly less than 1. The set of eigenvalues of \hat{L} is the union of the sets of eigenvalues of its blocks, and the set of its eigenvectors is the union of the sets of the eigenvectors of its blocks with zeros added appropriately to make them n -dimensional. In other words, the vector space spanned by the eigenvectors of \hat{L} is the direct sum of the vector spaces associated with the span of eigenvectors of each block.

And hence, we have:

$$\hat{V} = \begin{bmatrix} v_1^{(1)} & 0 & 0 \\ 0 & v_1^{(2)} & 0 \\ 0 & 0 & v_1^{(3)} \end{bmatrix} \quad (2.3)$$

where $v_1^{(i)}$ is the principal eigenvector of $\hat{L}^{(ii)}$. Finally, we normalize the rows of \hat{V}

to get

$$\hat{Y} = \begin{bmatrix} \vec{1} & \vec{0} & \vec{0} \\ \vec{0} & \vec{1} & \vec{0} \\ \vec{0} & \vec{0} & \vec{1} \end{bmatrix}. \quad (2.4)$$

If we apply K-Means on the rows of \hat{Y} , it will cluster together points around three unit vectors in \mathbb{R}^k . So, we have seen that the algorithm works for the ideal case. Now, we look at how and when it will work for a general case.

2.2.2 The general case

In the general case, the similarity between points of different clusters is non-zero, and therefore, the adjacency matrix and the Laplacian matrix are not block diagonals. The Laplacian for the general case is $L = \hat{L} + E$ where E is a perturbation matrix.

The question is when the eigenvectors of the general Laplacian and the ideal Laplacian will be close enough so that we get clusters similar to the ideal case? The answer to that is given by the Davis-Kahan theorem from matrix perturbation theory as explained by Luxborg(2007) [1].

Davis-Kahan Theorem. *Let $A, H \in \mathbb{R}^{n \times n}$ be symmetric matrices, and let $\|\cdot\|$ be the Frobenius norm or the two-norm for matrices, respectively. Consider $\tilde{A} := A + H$ as a perturbed version of A . Let $S_1 \subset \mathbb{R}$ be an interval. Denote by $\sigma_{S_1}(A)$ the set of eigenvalues of A which are contained in S_1 , and by V_1 the eigenspace corresponding to all those eigenvalues (more formally, V_1 is the image of the spectral projection induced by $\sigma_{S_1}(A)$.) Denote by $\sigma_{S_1}(\tilde{A})$ and \tilde{V}_1 the analogous quantities for \tilde{A} . Define the distance between S_1 and the spectrum of A outside of S_1 as*

$$\delta = \min\{|\lambda - s|; \lambda \text{ eigenvalue of } A, \lambda \notin S_1, s \in S_1\}.$$

Then the distance $d(V_1, \tilde{V}_1) := \|\sin \Theta(V_1, \tilde{V}_1)\|$ between two subspaces V_1 and \tilde{V}_1 is

bounded by

$$d(V_1, \tilde{V}_1) \leq \frac{\|H\|}{\delta}.$$

According to the theorem, the distance between the eigenspace corresponding to the first k eigenvalues of L and the eigenspace corresponding to the first k eigenvalues of \hat{L} has an upper bound of $\|E\|/\delta$ where δ is the eigengap between the k^{th} and the $(k+1)^{th}$ eigenvalues of L . For more details about the theorem, refer to Section V.3 of Stewart and Sun (1990) [13].

If we have a graph Laplacian L such that we can find a large enough eigengap δ after its k^{th} eigenvalue, then we know there is a block diagonal matrix \hat{L} which is similar to L , and the clusters in the graph are clearly separable.

We created an ideal adjacency matrix with all the elements on the block diagonals (except for the diagonal elements) being 1 and the rest being zero. The matrix has 3 blocks. Therefore, the dataset will have 3 clusters. Then, we plot the eigengap between the third and the fourth leading eigenvalues of the corresponding Laplacian. To compare it with non-ideal cases, we add small perturbations to the ideal adjacency matrix. Let \hat{A}_{ij} ($i \neq j$) be any element of the ideal adjacency matrix. We add a small error ϵ_{ij} , which is generated from a normal distribution. Then, we vary the standard deviation to create multiple perturbed adjacency matrices. For each value of standard deviation, we vary the number of data points and plot it against the eigengap to see if the eigengaps stabilize with an increasing number of data points. The figure for eigengaps of the ideal Laplacian matrix is shown in figure 2.1.

The plots of eigengaps for perturbed similarities are shown in Fig 2.2.

2.3 Constructing a graph

The first challenge while attempting to perform spectral clustering is to choose how to convert the given dataset into a graph. If you think of the dataset as a graph with similarity between two data points representing the weight of the edge connecting

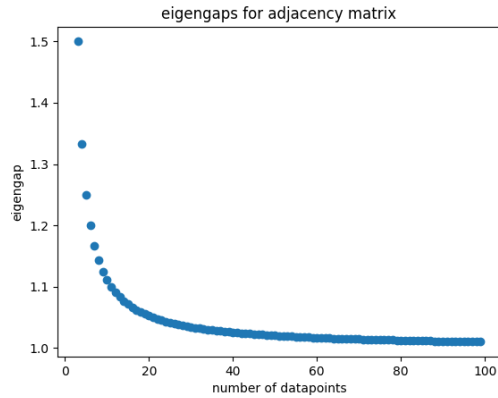


Figure 2.1: Eigengaps of the ideal Laplacian with 3 blocks.

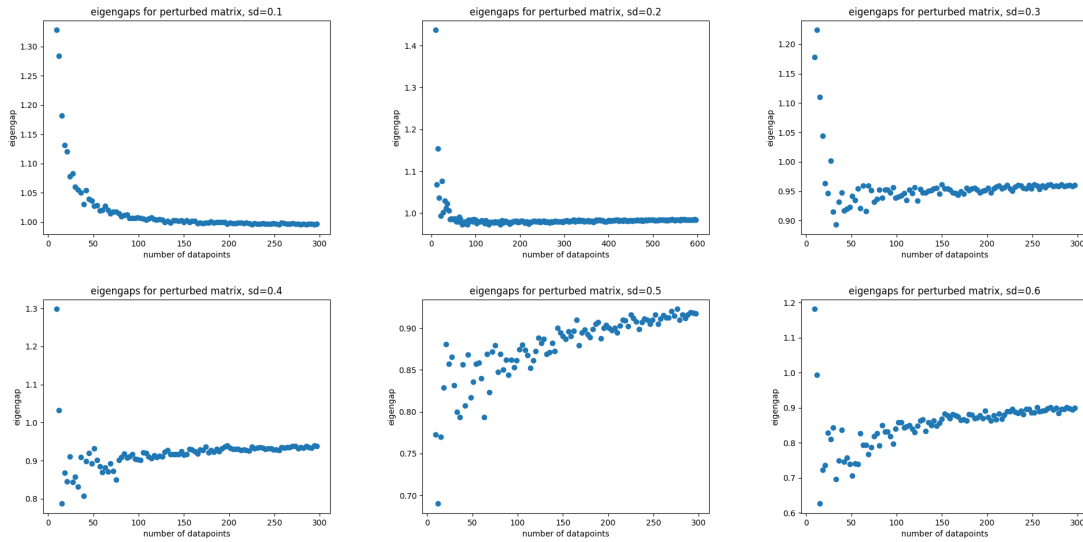


Figure 2.2: Eigengaps of Laplacian with 3 blocks for perturbed similarity matrices, with varying standard deviation.

those two points, then the problem is finding an appropriate method of constructing this graph. There are multiple ways we can do this. Some of them are given in Luxborg (2007) [1]:

- **ϵ -neighborhood graphs.** To generate this type of graph, choose a cutoff

parameter ϵ such that if the similarity between two points is more than ϵ , they are connected with an edge of weight zero. Otherwise, there is no edge added between them. This graph should be constructed only if it is known that all the clusters are similarly dense. Otherwise, no single value of ϵ can identify clusters. Therefore, we do not use it for spatiotemporal clustering.

- **k-nearest neighbor graphs.** In this graph, two vertices are connected if either one of them is among the k nearest neighbors of the other. This type of graph does not face the same problem as the ϵ -neighborhood graph, as it can work with clusters of different densities. However, we cannot use it for our project either, as it increases inter-cluster similarity if the clusters have different densities but are close to each other in some regions.
- **Mutual k-nearest neighbor graphs** In this graph, two vertices are connected if and only if both of them are among each other's k nearest neighbors. This graph is far more sparse than k-nearest neighbor graphs.
- **Fully connected graph** This graph is most commonly used in our project. It connects each vertex with all other vertices, and each edge is given a weight equal to the similarity between the two corresponding vertices. Andrew *et al* (2001) [11] use gaussian similarity function given by eq 2.1.

We use fully connected graphs and mutual k-nearest neighbor graphs for spatiotemporal datasets as they do not require any strict assumptions about the dataset. Luxborg (2007) [1] has illustrated the first three types of graphs on synthetic datasets.

The similarity function is a function $s(i, j)$ that determines the weight assigned to the edge connecting vertices i and j . For a fully connected graph, we use the following similarity function:

$$s(i, j) = \exp\left(-\frac{d(i, j)^2}{\sigma^2}\right). \quad (2.5)$$

Here, $d(i, j)$ is a distance function between data points corresponding to vertices i and j . Here, we need to choose an appropriate distance function as well as a good value of σ so that we can obtain meaningful clusters. For most datasets, we can

use the L1 norm or the L2 norm as our distance function, but sometimes, another technique called dynamic time warping (DTW) is useful. We will discuss DTW later. But first, we discuss how to choose a value of σ .

2.4 Choosing the parameter σ

In order to determine a method of choosing σ , first, we create a synthetic dataset and see how the changing values of σ change the results of spectral clustering. Our synthetic dataset consists of two concentric two-dimensional annuli with one thousand points in each annulus. To choose an initial value of σ , we use a thumb rule given by Luxborg (2007) [1]. The rule states that the value of σ should be of the order of the mean distance of a point to its k -th nearest neighbor where $k \sim \log(n) + 1$. For our dataset, this value of σ is around 0.5. The result is shown in figures 2.3 and 2.4. As we increase the value of σ , the rows of Y come all together due to increasing similarity values between data points. Also, the eigengap between the first eigenvalue and the second eigenvalue keeps increasing with increasing σ .

We find that varying σ once we have obtained an initial estimate using the above-mentioned thumb rule is quite useful in finding a good eigengap to estimate the number of clusters. As we have seen, increasing the value of σ decreases the number of clusters that can be obtained.

To test it, we use another synthetic dataset. This dataset contains a set of five balls, each containing 700 points. Then, we add random noise to the dataset to test if the technique of varying σ can detect clusters hidden between noise. The initial estimate of σ is 0.129. Let us call it σ_0 . Using this estimate, we find the Laplacian matrix and plot its eigenvalues shown in Fig 2.5. The eigengap is obtained after 10th eigenvalue even though we need 5 clusters. This is happening because the noise has caused some perturbations to our adjacency matrix. If we proceed with the spectral clustering algorithm using 10 as the number of our clusters, we get meaningless clusters that might ruin our cluster analysis. Therefore, we try to vary σ to see if we can get a value of σ such that it gives an eigengap after 5 eigenvalues of the Laplacian.

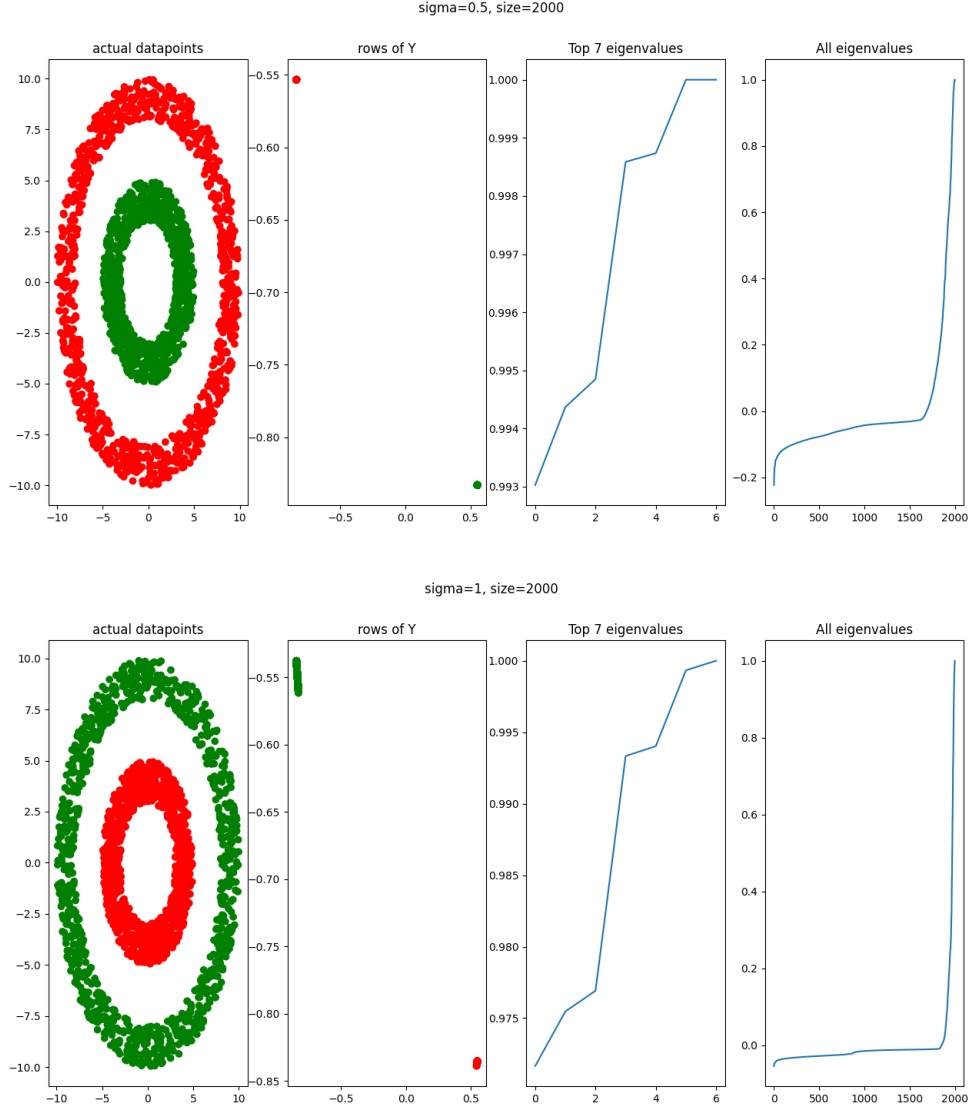


Figure 2.3: Clustering concentric annuli, for $\sigma = 0.5, 1$.

For $\sigma = 5\sigma_0$, the eigengap is present between the 5th and the 6th eigenvalues (see Fig 2.5b). Therefore, we use the corresponding Laplacian to cluster our dataset.

The clusters obtained by using σ_0 are not able to separate balls clearly, and we have some clusters formed purely due to noise, as can be seen in Fig 2.5c. But we find

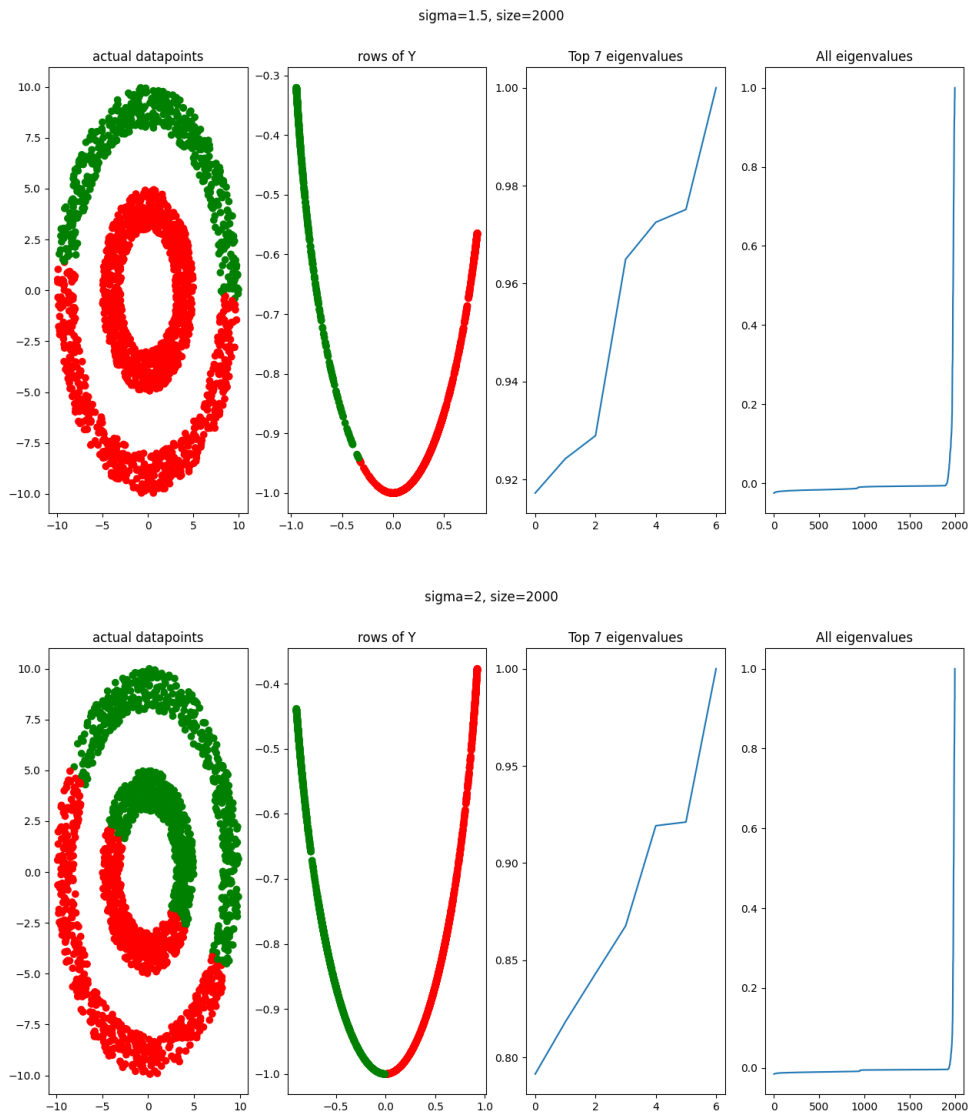
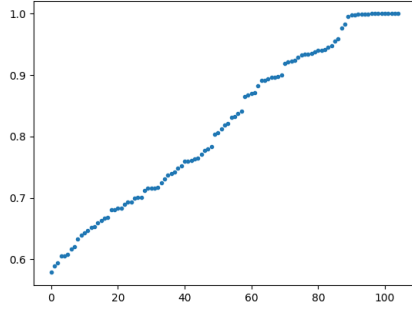


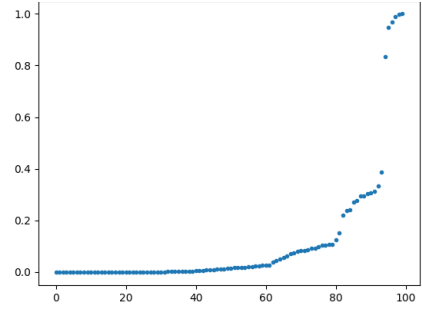
Figure 2.4: Clustering concentric annuli, for $\sigma = 1.5, 2$.

that as we vary σ , the clusters become very well separated and meaningful with little effect of noise. This is an effective technique to find clusters, and we will use it extensively in chapter 3.

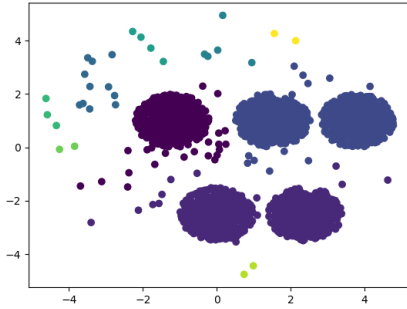
If the initial estimate of σ gives us too many clusters or clusters that are not mean-



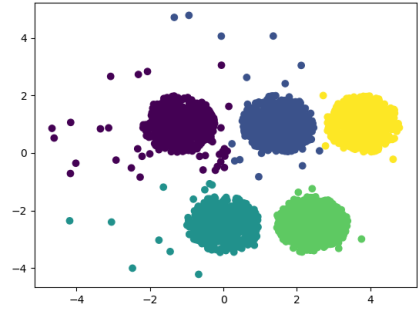
(a) Top 100 eigenvalues of Laplacian for $\sigma = \sigma_0$.



(b) Top 100 eigenvalues of Laplacian for $\sigma = 5\sigma_0$.



(c) Clusters obtained for $\sigma = \sigma_0$.



(d) Clusters obtained for $\sigma = 5\sigma_0$.

Figure 2.5: Clustering 5 balls with randomly added noise.

ingful, we should increase the values of σ in order to increase the pairwise similarity between data points. This will move the eigengap to the right, i.e., the number of clusters obtainable from the algorithm will decrease. If, on the other hand, the initial estimate of σ is giving very few clusters, or if a large eigengap is present after the largest eigenvalue itself, then decreasing the value of σ might help, but in most probability, there are no clusters obtainable if the eigengap lies between the first and the second eigenvalues.

At last, we try to look at how the eigenvalues will behave with the changing of σ if there are no clusters that can be obtained. For this, we create a synthetic dataset containing only points uniformly distributed on a disk with no identifiable patterns.

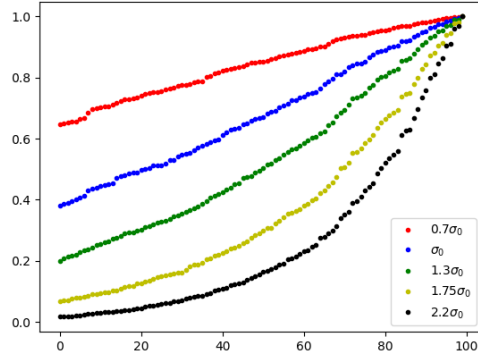


Figure 2.6: Top 100 eigenvalues of Laplacian for uniform points with no clusters.

Let the initial estimate of σ be σ_0 obtained using the above-mentioned thumb rule. We plot the eigenvalues for various values of σ to see how the eigenvalues behave (see Fig 2.6). There is no significant eigengap anywhere for smaller values of σ , and as we increase σ , we see that the eigengap between the first and the second eigenvalue starts increasing. This implies that even our manipulation of the σ value does not find a cluster.

Note that if the value of σ is decreased too much, the eigengap is almost always present after a very large number of eigenvalues, but the clusters thus obtained are not meaningful and contain a very small number of data points as was seen in Fig 2.5c. Another problem with σ being too small is that the pairwise similarities become so small that the machine used to run the algorithm may record the similarities as zero.

2.5 Dynamic time warping (DTW)

So far, we have used the L1 norm for measuring distances between two data points. This distance is converted to similarity using the Gaussian similarity function as shown in eq 2.5. However, if our data points are in the form of a time series, then there might be a phase difference that the L1 norm will not consider. Therefore, a better method of measuring the distance between two time series is to use DTW, first

introduced in 1959 [14], as its algorithm considers the phase difference that might exist between two otherwise similar time series.

Let us say we have two time series, each of length n . In the DTW algorithm, a cost matrix $C \in \mathbb{R}^{n \times n}$ is created where C_{ij} is the distance between the i^{th} element of the first time series and the j^{th} element of the second time series. The task is to find a path $P = (p_1, p_2, \dots, p_k)$ from C_{11} to C_{nn} where $p_i = (x_i, y_i)$ and x_i is an element of the first time-series, while y_i is an element of the second time series such that the cost of the path is minimized. The cost of a path is the sum of distances between elements of two time series corresponding to each p_i . Senin (2009) [15] has created a review of DTW, which contains a detailed algorithm of DTW.

One important part of DTW is a slope constraint, introduced by Sakoe and Chiba (1978) [16]. This constraint controls the slope of the optimal path so that it is neither too steep nor too gentle. If this constraint is not applied, the resulting path might end up comparing one small part of the first time series to the entire second time series, creating an unrealistic distance measure. Let us say we put the slope constraint at n/m . This would imply that any path that the DTW algorithm looks at must be such that any m consecutive movements along the same time series should be followed by at least n movements along the diagonal. We also need the distance measure to be symmetric for the spectral clustering to work. Therefore, the slope constraints will be symmetric for both time series.

The programming language Python has a package called ‘dtw-python’ that contains some in-built slope constraints that we will use while dealing with time-series datasets. For more information about using dtw in Python, please see Giorgino (2009) [17]. If the length of each time series is n , then the algorithm’s complexity is $O(n^2)$ [15]. Therefore, for high-dimensional datasets, DTW is much slower than using the L1 norm.

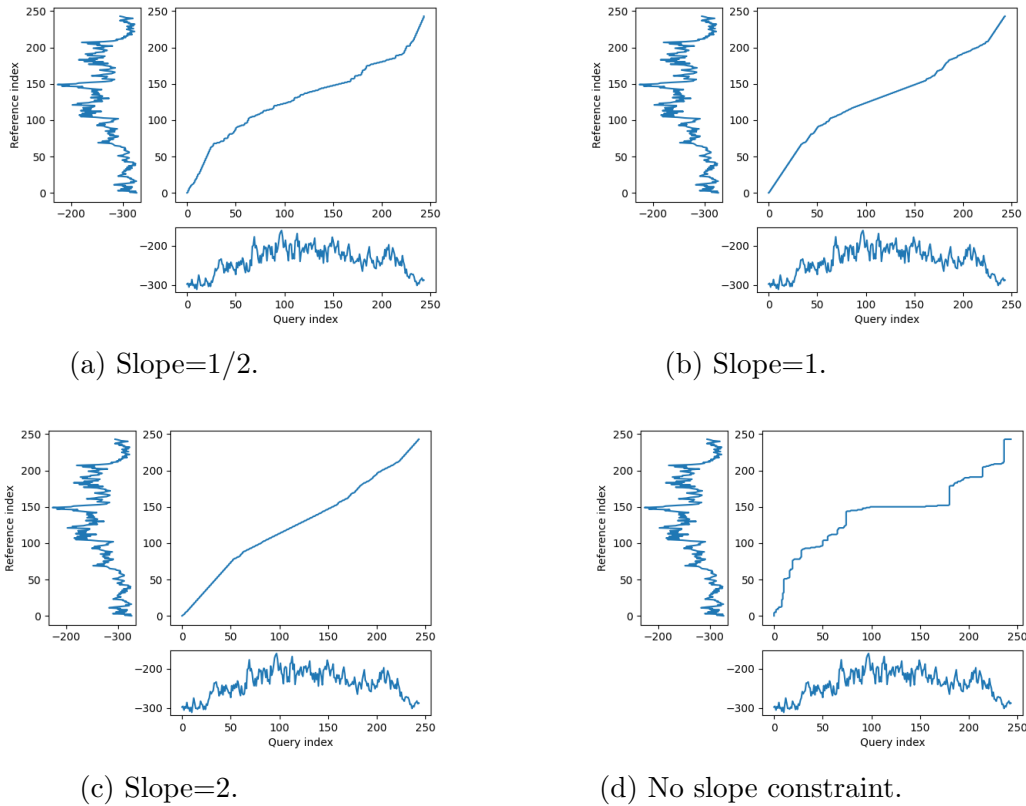


Figure 2.7: DTW optimal warping paths for different symmetric slope constraints.

2.5.1 Examples

In order to show the differences between different values of the sakoe-chiba slope constraint, we take two OLR time series from the dataset mentioned in section 1.3. The first time series T_1 contains OLR values at the coordinates 26.5°N , 81.25°E , while the second time series T_2 contains OLR values at the coordinates 26.75°N , 71°E . The OLR values are obtained by averaging over every 12 hours from June 1 to September 30, 2010. Therefore, the length of each time series is 244.

The symmetric constraints we use are labeled as P05 (slope=1/2), P1 (slope=1), P2 (slope=2), and P0 (no constraint) in the ‘dtw-python’ package that was used to get these illustrations.

Constraints	$d_{dtw}(T_1, T_2)$	$d_{dtw}(T_3, T_2)$	$d_{dtw}(T_3, T_1)$	$\frac{d_{dtw}(T_3, T_2)}{d_{dtw}(T_3, T_1)}$
P0	8.022	12.029	7.117	1.69
P05	17.585	38.533	11.202	3.44
P1	22.441	46.748	15.271	3.061
P2	29.482	53.627	19.132	2.803

Table 2.1: Symmetric constraints values for comparing three time series.

As we see in Figure 2.7, the higher the slope constraint, the more the optimal warping path is similar to the diagonal. This means that using DTW with high slope constraints will be similar to using the L1 norm as our distance measure. The result when there is no slope constraint is highly unrealistic as the algorithm compares very few elements of the second time series (elements around its 150th element) with almost half of the first time series. It is also very slow as it has to optimize over a far greater number of possible warping paths compared to having some slope constraints.

If clustering using the L1 norm gives some meaningful clustering with only a part of the dataset not clustered well, we can use higher slope constraints, but if the L1 norm clustering is very poor or meaningless, lower values of slope constraints need to be used. However, we do not use the P0 (no constraint) algorithm because it gives low distances between data points that are supposed to be very different. To see how zero slope constraint ruins our similarity matrix, we take three time series obtained similarly as T_1 and T_2 were. Let's call them T_1, T_2 and T_3 . The time series T_1 and T_2 are the same as the ones defined above. The third time series T_3 is associated with the location at coordinates $21^\circ\text{N}, 79.25^\circ\text{E}$.

The locations associated with time series T_1 and T_3 receive similar rainfall patterns during the monsoon period, while the location associated with T_2 is a dry region receiving very different levels of rainfall than the other two. Therefore, we expect DTW to give a lower distance between T_1 and T_3 than between T_2 and T_3 .

Table 2.1 shows how different constraint values distinguish between a pair of similar

time series (with phase difference) and a pair of asimilar time series. The values for when there is no constraint (P0) are very similar to each other, and the ratio $d_{dtw}(T_3, T_2)/d_{dtw}(T_3, T_1)$ is close to 1 even though T_3 and T_2 have different levels of OLR values while T_3 and T_1 have similar values. P05 gives the highest value of the aforementioned ratio, and from there, the ratio keeps falling. As the slope constraint gets stricter, the algorithm is no longer able to detect significant phase differences between T_3 and T_1 , and hence sees them as vastly different time series. Therefore, the slope constraint should not be kept too high even though it makes the algorithm much faster.

2.6 UMAP

Uniform Manifold Approximation and Projection, or UMAP, is a relatively new technique that can reduce the dimension of a given dataset without losing much information if the data points lie on a low-dimensional manifold in a high-dimensional data space. The UMAP algorithm uses category theory [18] to detect a locally connected low-dimensional manifold on which the data is assumed to be uniformly distributed. To see the theoretical explanation and the complete algorithm of UMAP, refer to McInnes *et al* (2020) [19].

Generally, UMAP is used to visualize a high-dimensional dataset on a 2-D or a 3-D space. But for our project, we use it to reduce the dimension of time-series datasets in order to cluster them as high-dimensional dataset loses variation in pairwise similarities if the number of data points is not large enough compared to the dimension of the data space [20]. The problem with this approach is that we do not know the dimension of the underlying manifold. We tried to build an algorithm to solve this problem but got no satisfactory solution. The algorithm will be discussed later in the section.

To see the workings of UMAP, we begin with an example using a synthetic dataset. The dataset used is shown in Fig 2.8a. The UMAP algorithm is applied to the whole dataset, and the resulting projection is shown in Fig 2.8b. Finally, the UMAP

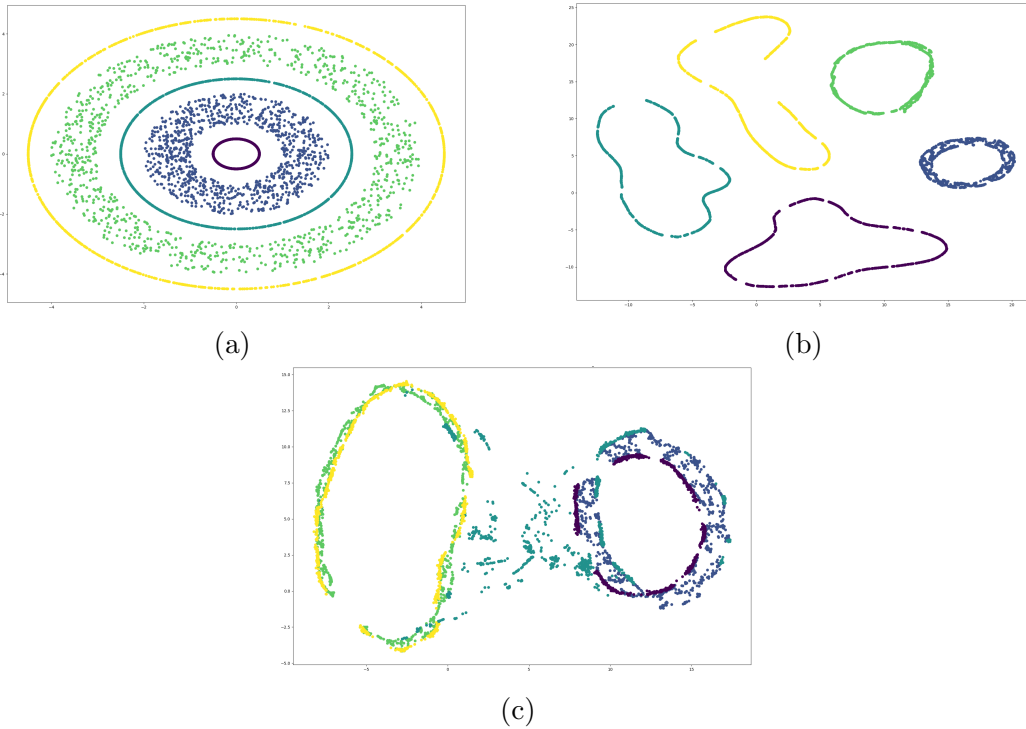
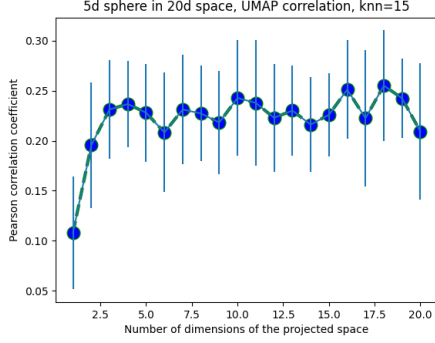


Figure 2.8: Application of UMAP on a synthetic dataset.

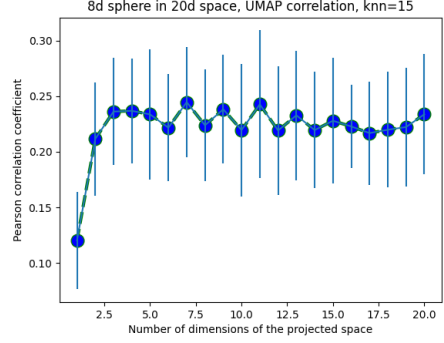
algorithm is applied only to the two concentric annuli in the original dataset, and then the projection function obtained through UMAP is used to plot the rest of the dataset. The result is shown in Fig 2.8c. An important assumption of the UMAP algorithm is that given an underlying manifold, the data is uniformly distributed on it. Therefore, whenever there is a low-density region between two annuli, the algorithm separates them, considering them as two different manifolds. It also builds a projection function that is continuous between the two annuli. Hence the difference between figures 2.8b and 2.8c.

2.6.1 A correlation algorithm

In order to detect the dimension of the underlying manifold in the data space, we develop an algorithm using the following lemma given in McInnes *et al* (2020) [19]:



(a) 5-dimensional sphere in \mathbb{R}^{20}



(b) 8-dimensional sphere in \mathbb{R}^{20}

Figure 2.9: Result of the correlation algorithm on synthetic dataset.

Lemma. *Let (\mathcal{M}, g) be a Riemannian manifold in an ambient \mathbb{R}^n , and let $p \in \mathcal{M}$ be a point. If g is locally constant about p in an open neighborhood U such that g is a constant diagonal matrix in ambient coordinates, then in a ball $B \subseteq U$ centered at p with volume $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$ with respect to g , the geodesic distance from p to any point $q \in B$ is $\frac{1}{r}d_{\mathbb{R}^n}(p, q)$, where r is the radius of the ball in the ambient space and $d_{\mathbb{R}^n}$ is the existing metric on the ambient space.*

Note that r only depends on the manifold and the dimension of the data space, while the locally constant metric g is constructed using Fuzzy simplicial sets. The lemma tells us that the geodesic distance on the manifold is a linear function of the Euclidean distance if the points are close enough. We use this fact to use Pearson’s correlation coefficient to detect the dimension of the manifold. The algorithm we developed is as follows:

1. Choose randomly a certain number of data points from the original dataset.
2. For each chosen data point i , find its Euclidean distance from its $k - th$ nearest neighbor and from its $k_0 - th$ nearest neighbor where $k < k_0$. k_0 is the argument for the hyper-parameter ‘n-neighbor’ that we feed to the UMAP algorithm. UMAP then tries to minimize the distance between each data point and its first $k_0 - th$ nearest neighbor.

3. Divide the two distances to get x_i .
4. Run UMAP to project the dataset on a low dimensional space \mathbb{R}^d for various values of d .
5. On the projected space, find the Euclidean distance from point i to its $k - th$ nearest neighbor on the original data space. Let it be y_i .
6. For each projection, calculate the Pearson's correlation coefficient ρ between the ratio of distances calculated in the original data space and the distances calculated in the projected space. The Pearson's correlation coefficient between two samples $\{x_i\}$ and $\{y_i\}$ is given by

$$\rho = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2}}. \quad (2.6)$$

7. The projection for which the value of ρ is highest should be a good projection.

Unfortunately, this algorithm does not perform on the synthetic datasets for which we tried it. To test it, we generate a dataset in \mathbb{R}^{20} , with the data points lying on a lower-dimensional manifold.

In Fig 2.9a, the data points were generated on a 5-D sphere embedded in \mathbb{R}^{20} , while in Fig 2.9b, the sphere was 8-D. In both plots, the value of Pearson's correlation coefficient becomes stable after the first 3 values with significant error bars. Therefore, the algorithm is not able to detect the embedded manifold that we intended to detect. The choice of Pearson's coefficient may not be suitable for this task, and a better measure of similarity between the original space and the projected space is needed.

Chapter 3

Application and Results

In order to apply spectral clustering on the datasets mentioned in section 1.3, we first apply exploratory data analysis, try to reduce the dimensions of high-dimensional datasets, and try to estimate the number of clusters that can be obtained. We begin with a low-dimensional dataset containing information about earthquakes. Then we try to cluster OLR time-series and maps datasets with mixed results. Finally, we end with some clusters of the rainfall time-series data.

We also show some interpretations of the clusters and eigenvalues that we obtain.

3.1 Earthquake dataset

As described before, this dataset contains latitudes and longitudes of major earthquakes that have occurred this century. It also contains the time of the earthquakes, but in order to cluster a dataset with dimensions of different units, we needed a more dense dataset. So we only clustered it in spatial dimensions. The purpose of clustering these earthquake points is to see if spectral clustering can detect, without any information from Earth Science and simply using the unsupervised algorithm, the major fault lines between Earth's tectonic plates.

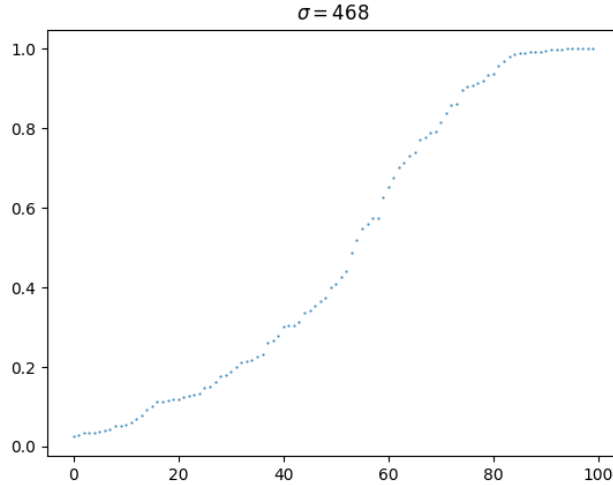


Figure 3.1: Eigenvalues of the Laplacian for the earthquake dataset

We use the Gaussian similarity measure to give weights to the graph's edges. The similarity measure we use is

$$s(i, j) = \exp\left(\frac{-(d_{geo}(i, j))^2}{2\sigma^2}\right), \quad (3.1)$$

where $d_{geo}(i, j)$ is the geodesic distance between i and j , the two locations given by their latitudes and longitudes. Let us assume the latitudes of i and j are θ_i and θ_j respectively, while their longitudes are ϕ_i and ϕ_j respectively. The geodesic distance between them would be:

$$d_{geo}(i, j) = 6378.8 \cos^{-1}(\sin \theta_i \sin \theta_j + \cos \theta_i \cos \theta_j \cos(\phi_i - \phi_j)) \quad (3.2)$$

Using this distance measure and the thumb rule mentioned before to estimate a value of σ , we get $\sigma = 468$. After creating the graph and its Laplacian matrix, we plot the top 100 of its eigenvalues in Figure 3.1.

There are multiple significant eigengaps in the plot, as can be seen in Fig 3.1. We want to keep the number of clusters to a minimum due to a small dataset. There-



Figure 3.2: Earthquake clusters over Asia-Pacific, $\sigma = 468, k = 17$

fore, we choose two eigengaps, the first one between the 17th and the 18th largest eigenvalues, and the second one between the 26th and the 27th largest eigenvalues. We have to run the clustering algorithm twice, once for $k = 17$, and then for $k = 26$, where k is the number of clusters we want.

Increasing the number of clusters does help in separating some fault lines from each other. For example, when we have $k = 17$, the algorithm clusters together the earthquakes along the Eurasian and the Indian plates with those along the Indian and Arabian plates. Both of the clusterings have correctly detected fault lines between the Philippine plate and its neighboring plates.



Figure 3.3: Earthquake clusters over Asia-Pacific, $\sigma = 468, k = 26$

The map with fewer clusters has managed to keep all earthquakes occurring between the Indo-Australian plate and the Pacific plate in one cluster, but the map with 26 clusters has broken it into 3 different clusters. The reason for this could be that we do not have a bigger dataset, as this dataset contains only those earthquakes that were above the magnitude of 6.5.

In both cases, we have managed to separate earthquakes between the Indo-Australian plate and the Eurasian plate from the earthquakes along the border between the Indo-

Australian plate and the Pacific plate. These detections would not have been possible without proper information about the number of clusters required. The number is confirmed by looking at the eigengap, and not by providing any theory from Earth Science. Therefore, it is correct to conclude that Luxborg’s thumb rule [1] is sufficient to detect fault lines for this earthquake dataset.

Recall from the Davis-Kahan theorem that the upper bound on the distance between the eigenspace of the ideal Laplacian and that of the perturbed Laplacian is $\|H\|/\delta$. If the clusters are dense, the intra-cluster similarity increases which makes the clusters very tight. According to the theorem given by Andrew *et al* (2001) [11], if the clusters are well connected, or ‘tight’, then even a small eigengap can be used to detect those clusters using spectral clustering. Therefore, if we had a bigger dataset, the problem that arose with detecting fault lines might not have occurred.

3.2 OLR dataset

This dataset contains hourly values of outgoing longwave radiation intensity at the locations in and around the Indian subcontinent from 10°S to 40°N and from 60°E to 100°E after every 0.25° of latitude and longitude. These values were produced using climate reanalysis. The dataset is used to create two new types of datasets, the time series dataset and the maps dataset, which are discussed below.

3.2.1 Time series dataset

The time-series dataset is obtained by creating a time-series of OLR values at each location given above during the monsoon period in the Indian subcontinent. This period is assumed to be from June 1 to September 30, 2010, although yearly variations exist.

Since the initial time series contains hourly data, the length of each time series will be $24 \times 122 = 2928$. If each time series is considered to be a data point, the dataset will be 2928 dimensional, which is quite high, and we could lose the similarity variations

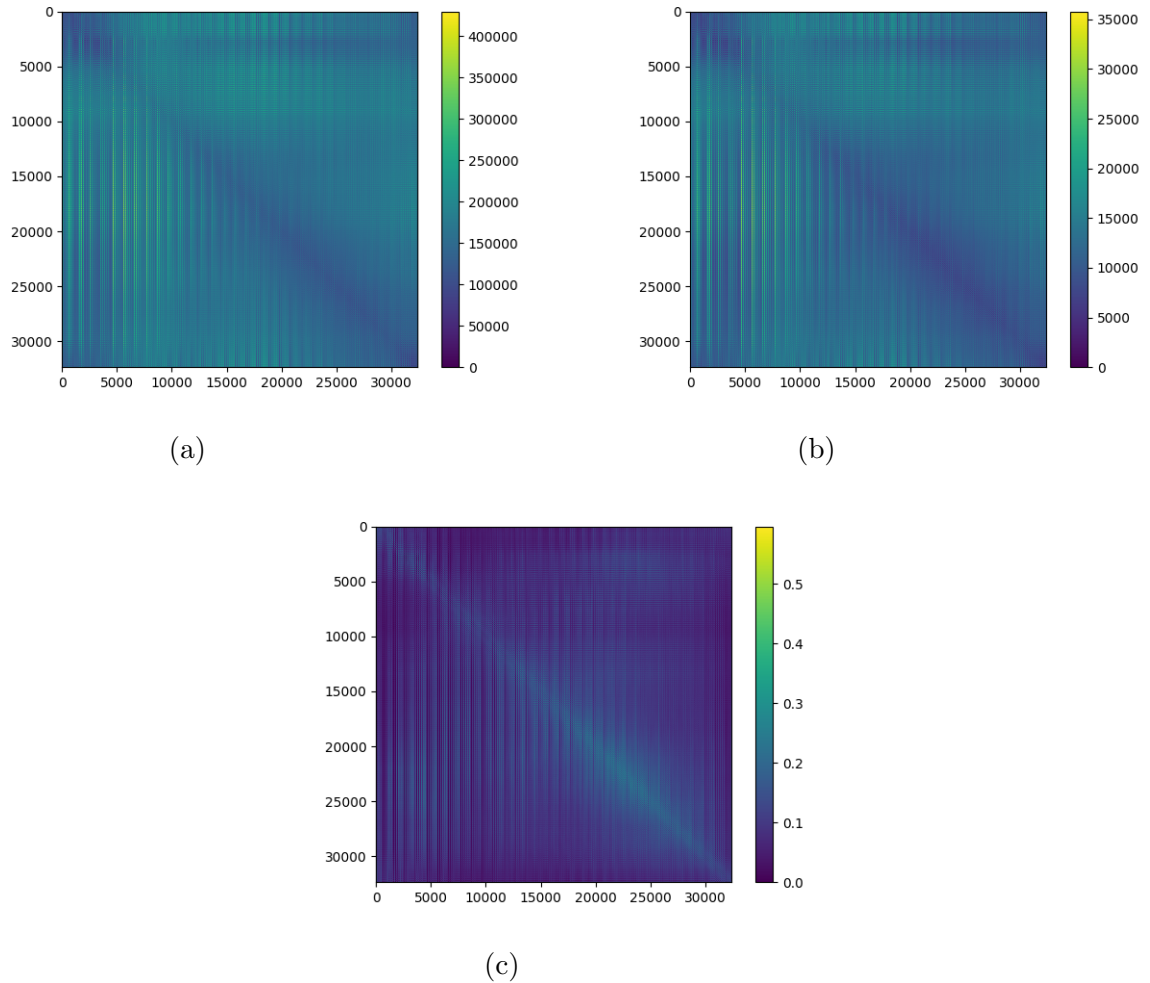


Figure 3.4: L1 Norm distance matrices for (a) Hourly time-series and (b) Time-series averaged over every 12 hours. And (c) The difference between the two matrices divided by the first matrix.

among the pairs of data points. The algorithm also takes much longer to run as calculating the distance between two time series using the L1 norm requires $O(n)$ time while DTW requires $O(n^2)$ time, where n is the length of the time series.

In order to reduce the dimensions of the dataset, the simplest thing to do would be to average the OLR values over 12 hours. If we do that, the new dimension of the

dataset will be $2928/12 = 244$. However, we need to ensure that while reducing the dimensions, the information about the pairwise similarities of the data points is not lost. In order to do that, we plot the distance matrices for the hourly time-series dataset and the averaged time-series dataset over every 12 hours.

The plots are shown in Figure 3.4. Although the matrices look very similar, we plot the matrix resulting from subtracting the above two matrices and then dividing it by the first matrix. We cannot simply subtract the two matrices as the distance matrix for the hourly time-series dataset has bigger distances than the one for 12-hour averaged time-series data due to the bigger length of the hourly time-series. To account for that, we divide the distance matrix for the hourly time-series data by 12. The plot of the resulting matrix is shown in Fig 3.4c. Most elements of the matrix shown in Fig 3.4c are less than 0.2 as can be seen using the colorbar. Therefore, very little information is lost on avergaing the OLR values every 12 hours.

From now on, OLR time-series will mean 12 hours averaged OLR time-series until otherwise stated. We now create a graph where each vertex corresponds to a time series, and the weight of the edges between vertices i and j will be given by the similarity function:

$$s(i, j) = \exp\left(\frac{-\left(\|i - j\|_1\right)^2}{\sigma^2}\right), \quad (3.3)$$

where we use the L1 norm as the distance function. Next, we find an initial estimate of σ with the same method as before. Let us call the initial estimate $\sigma_0 = 1434.58$. We vary σ around σ_0 and build the Laplacian matrix for each of those values of σ . The plots of the top 100 eigenvalues for all the values of σ that we tried are given in Fig 3.5. As we increase the σ , the $k - th$ eigenvalue decreases for all k except $k = 1$. The top eigenvalue always remains 1.

For very small values of σ , a significant eigengap is found only for larger k 's, which means we will obtain too many clusters, and it will be difficult to interpret them. For example, $\sigma = 0.2\sigma_0$ gives an eigengap after the top 60 eigenvalues. On the other hand, for very large values of σ , the eigengap between the largest and the second largest eigenvalues is significantly big, meaning the algorithm will be unable to detect

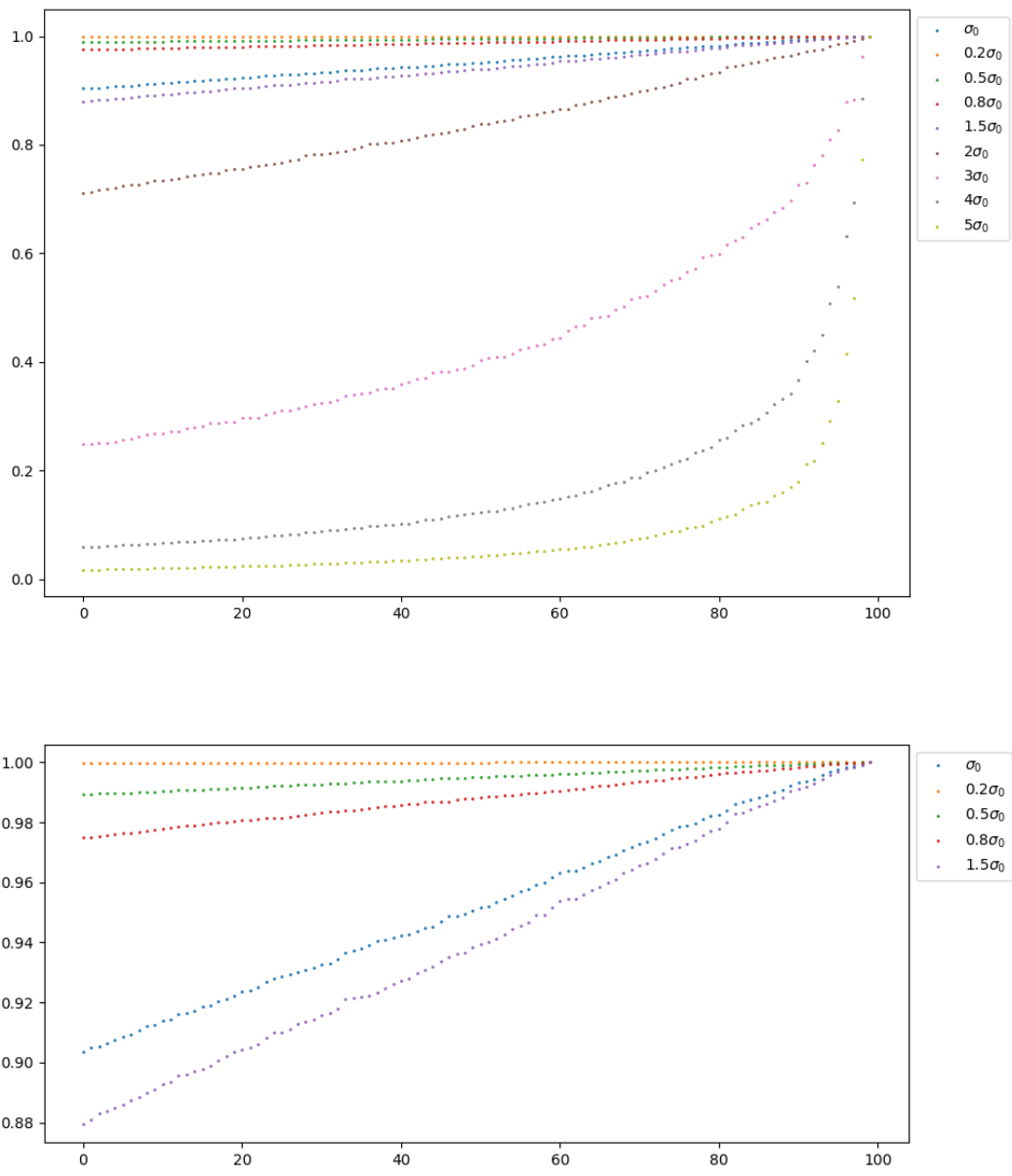


Figure 3.5: Eigenvalues of graph Laplacian, for different values of σ , OLR time-series dataset.

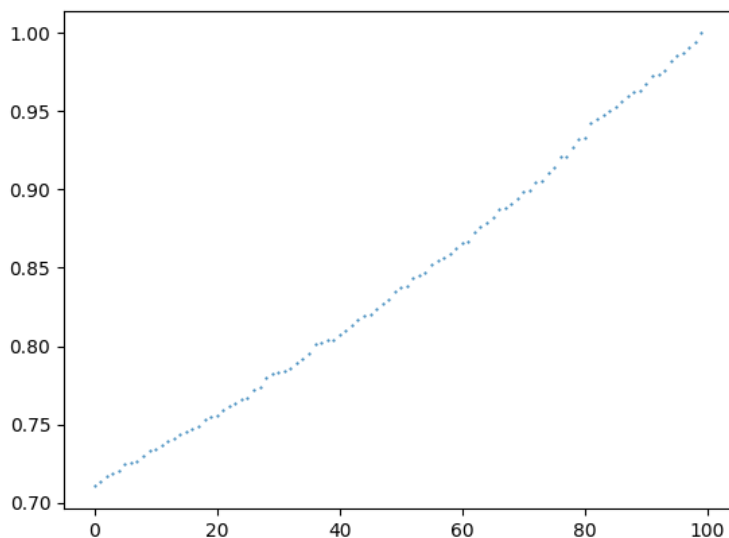


Figure 3.6: Top 100 eigenvalues of the Laplacian matrix for $\sigma = 2\sigma_0$, OLR time-series data.

any clusters at all.

Only for $\sigma = 2\sigma_0$ do we find a significant eigengap after a limited number of eigenvalues so that we neither get too many small clusters nor get only 1-2 clusters. The top 100 largest eigenvalues of the Laplacian matrix for $\sigma = 2\sigma_0$ are shown in Fig 3.6. The eigengap we choose is between the 19th and the 20th largest eigenvalues. Therefore, we ran the algorithm to find 19 clusters in the dataset. The clusters are shown in Fig 3.7.

The algorithm has simply clustered together points that are close together in spatial coordinates. This is because the L1-norm function is incapable of considering any phase difference two time series might have. So, no matter how similar the two time series might be, the L1 norm is going to give a large distance if there is a significant spatial distance between the two corresponding locations. In order to take into account the phase differences, we use dynamic time warping.

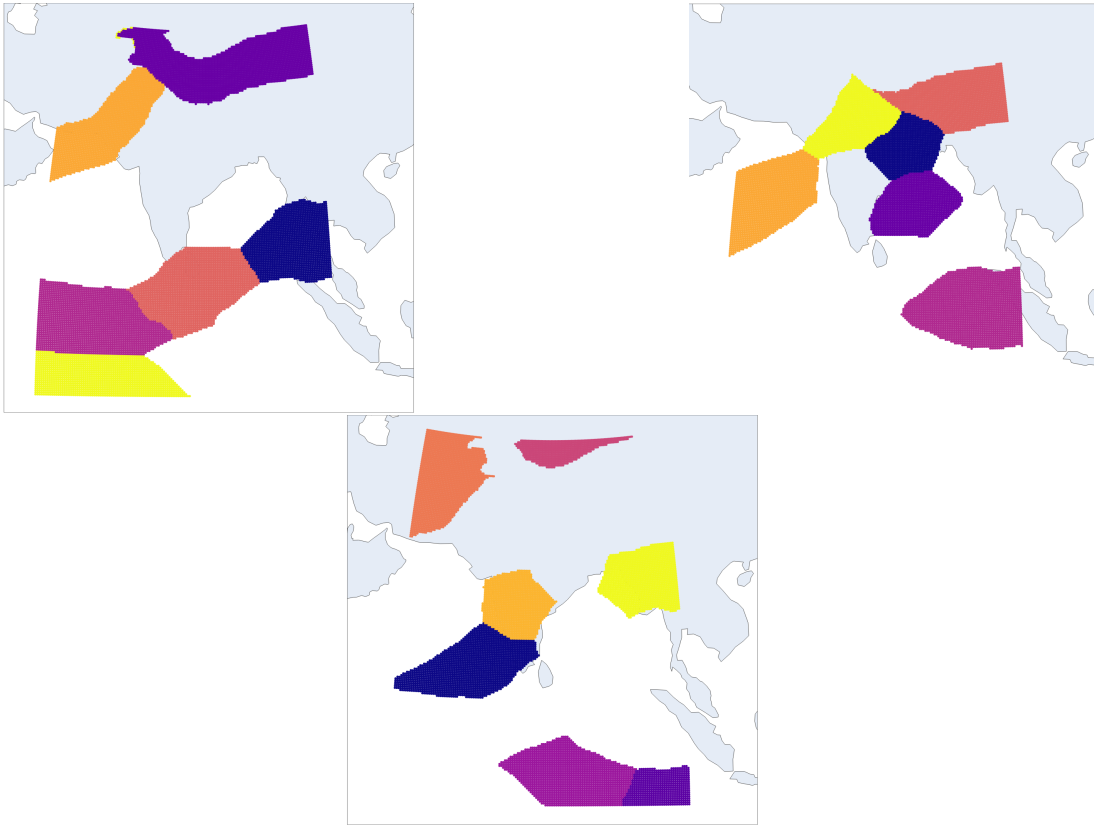


Figure 3.7: The 19 clusters of OLR time-series data, using l1-norm distance function, and $\sigma = 2\sigma_0$, shown in three different different plots for clarity.

Dynamic Time Warping (DTW)

The DTW algorithm is very slow, and it was not possible to try it using various slope constraints. We choose the symmetric P05 constraint because it would account for most phase differences but would not give unrealistic comparisons that P0 gives, as seen in Section 2.5.

The same algorithm as above is followed, except this time, we use the DTW distance function instead of the L1-norm. We try to cluster for different values of σ estimated in the same manner as before, but the algorithm simply clusters almost all of the subcontinent together. Some points in the northern part form small clusters, but

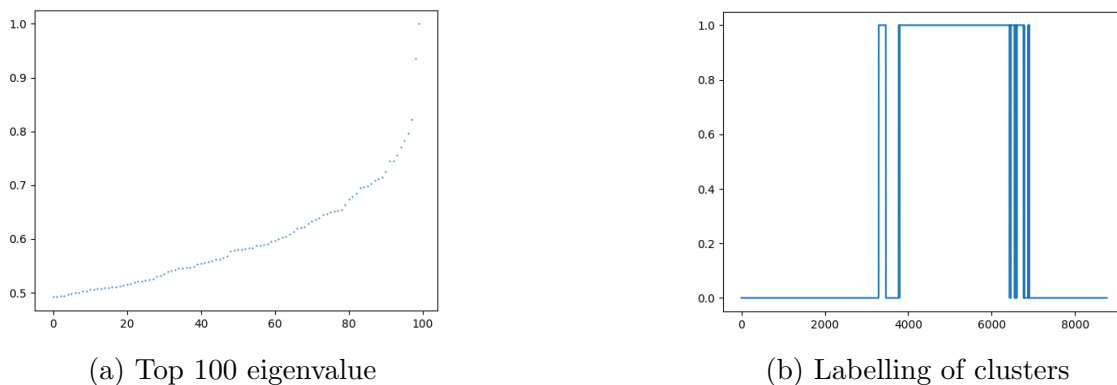


Figure 3.8: Clustering of OLR 2010 maps to detect monsoon period.

they are too small to give us meaningful interpretations. The P05 might be too low of a constraint that makes most of the points similar. More strict constraints like P1 or P2 might be tried, but due to their heavy time-consuming nature, we do not try them.

3.2.2 Maps dataset

This dataset is derived from hourly OLR values in the subcontinent from 2010. Each data point (also called a map) lies in $\mathbb{R}^{32,361}$ and contains the OLR values at 32,361 locations all over the Indian subcontinent at a time instant. In other words, each data point is a snapshot of the OLR values over the entire subcontinent at a particular time. The total number of data points is $365 \times 24 = 8760$. We again begin with using L1 norm for measuring distances between maps.

The main goal of clustering maps is to determine if the monsoon period can be separated from the rest of the year using spectral clustering and appropriate manipulation of the parameter σ . Using earlier methodology, we obtain the initial estimate of σ , $\sigma_0 = 669,255$. Since we already know that we need 2 clusters and have already seen how the eigengap behaves with σ varying, we just need to manipulate the value of σ so that the eigengap is significant after the second largest eigenvalue. We get that eigengap for $\sigma = 0.8\sigma_0$ as shown in Fig 3.8a. Using this σ , and the Gaussian

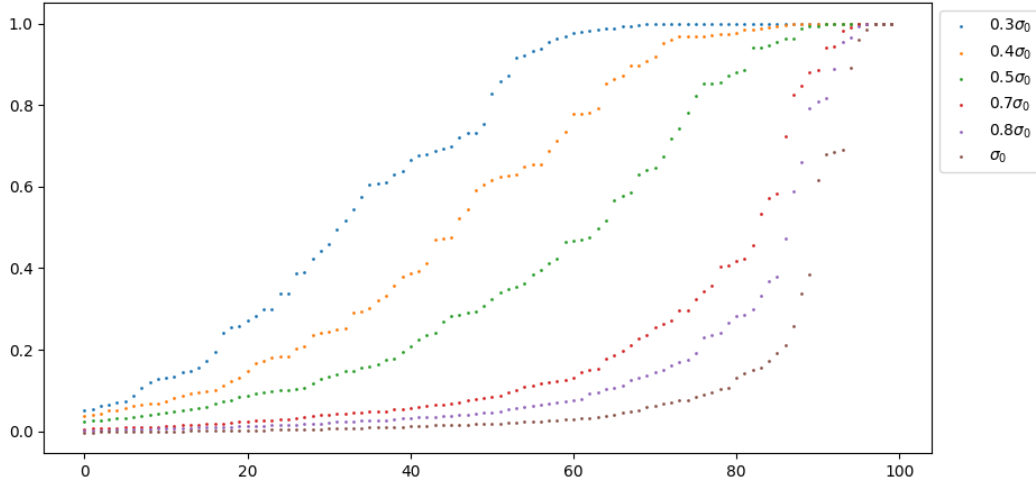


Figure 3.9: Top 100 eigenvalues of the Laplacian for the rainfall time series data

similarity function as given in eq 3.3, we run the spectral clustering algorithm to obtain two clusters.

The two clusters are shown in Fig 3.8b. Label 1 on the y-axis corresponds to the monsoon maps, while label 0 corresponds to non-monsoon maps. It has very accurately separated the monsoon from the rest of the year, as can be seen.

3.3 Rainfall data

The rainfall data contains daily rainfall values over India interpolated into grids of $1^\circ \times 1^\circ$ created by Rajeevan *et al* (2006) [9]. We use the same similarity function and the same method for the initial estimate of σ as for the OLR time series data.

The initial estimate of σ is $\sigma_0 = 807.54$. We vary the value of σ and calculate the eigenvalues of the Laplacian for each value. The plot of the top 100 eigenvalues of the Laplacian for various values of σ is given in Fig 3.9.

For $0.3\sigma_0$, $0.4\sigma_0$, and $0.5\sigma_0$, we find a significant eigengap after the 46^{th} , 29^{th} , and

18th largest eigenvalues, respectively. We run the algorithm using these parameter values. The clusters are shown in Fig 3.10, along with the number of points in each clusters.

The clusters get progressively smaller as we decrease the σ value due to decreasing pairwise similarities. Most points are in one cluster, while smaller clusters correspond to high-rainfall areas. Therefore, the algorithm has separated high-rainfall regions from the rest of India without getting any information about the number of clusters from climate science.

3.4 OLR vs Rainfall datasets

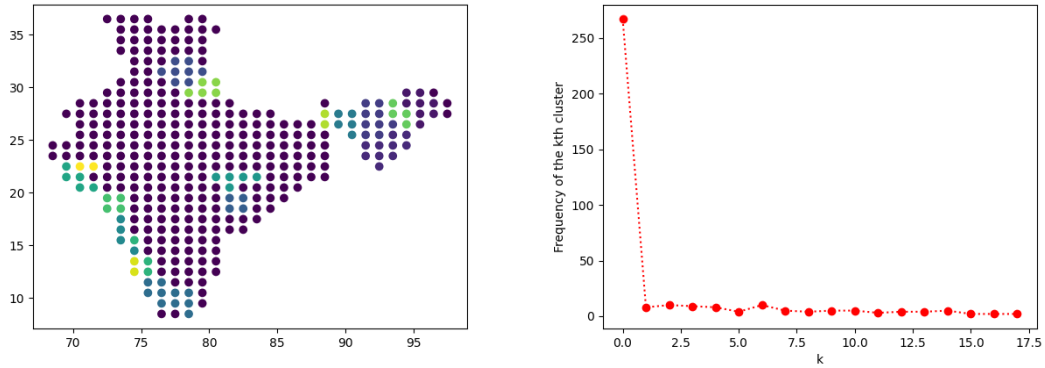
We compare the two time-series datasets to see why rainfall time-series have been clustered but the OLR time-series could not be. To see the difference between the two datasets, we plot histograms of pairwise distances and UMAP projections.

3.4.1 Histogram of pairwise distances

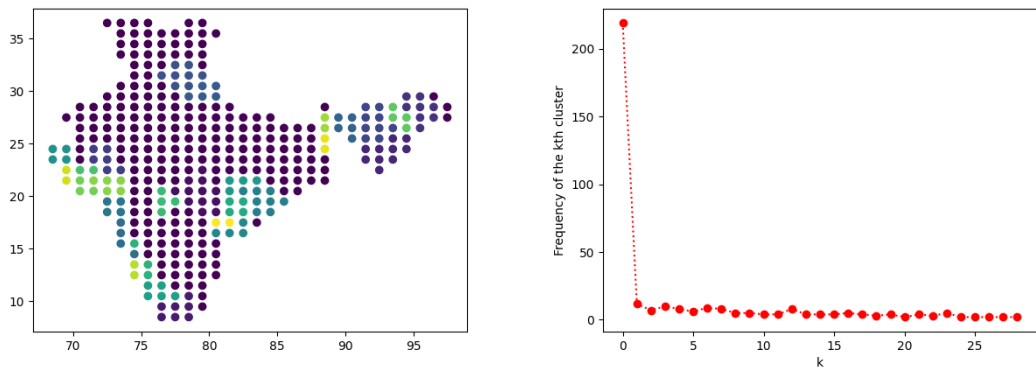
We plot the histogram of pairwise distances of all possible pairs of data points for both the OLR time-series and the rainfall time-series datasets. The plots are shown in Fig 3.11. If the histogram has more than one peak, it implies that there exists at least two clusters in the data with respect to the distance function used [20]. As can be seen in the plot, the OLR histogram has only one peak.

3.4.2 UMAP

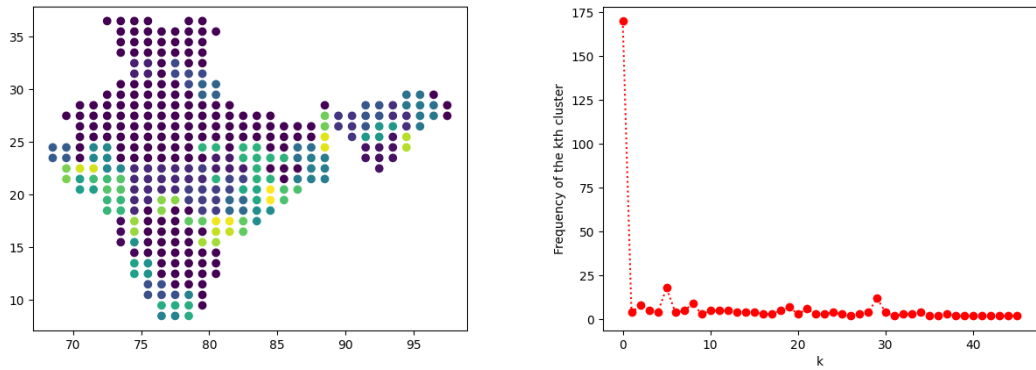
We now plot their UMAP projections on \mathbb{R}^2 to visualize the differences between rainfall and OLR time-series datasets. The projections are shown in Fig 3.12. The rainfall data has far more patterns than the OLR dataset. OLR data points are scattered almost uniformly on the manifold, making it difficult to cluster them. Therefore, we conclude that it is not possible to cluster the OLR time-series dataset that we had.



(a) $\sigma = 0.5\sigma_0, k = 18$

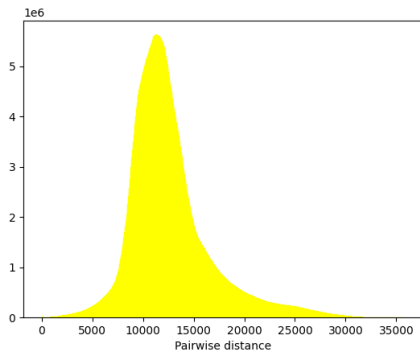


(b) $\sigma = 0.4\sigma_0, k = 29$

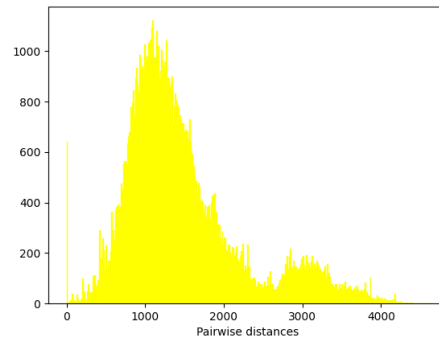


(c) $\sigma = 0.3\sigma_0, k = 46$

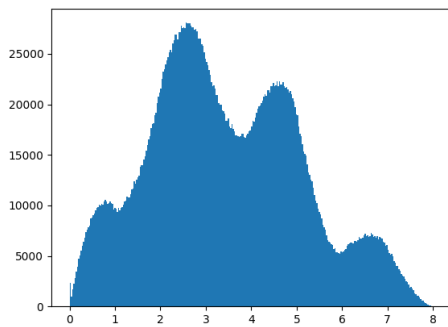
Figure 3.10: Rainfall time series data clusters, using L1 norm. The scatter plot show the number of data points in each cluster.



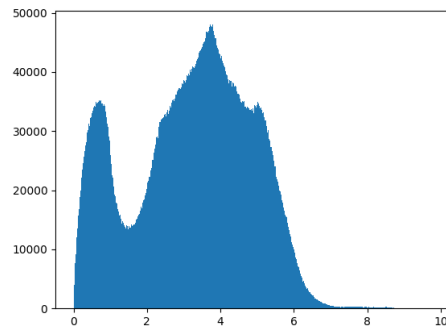
(a)



(b)

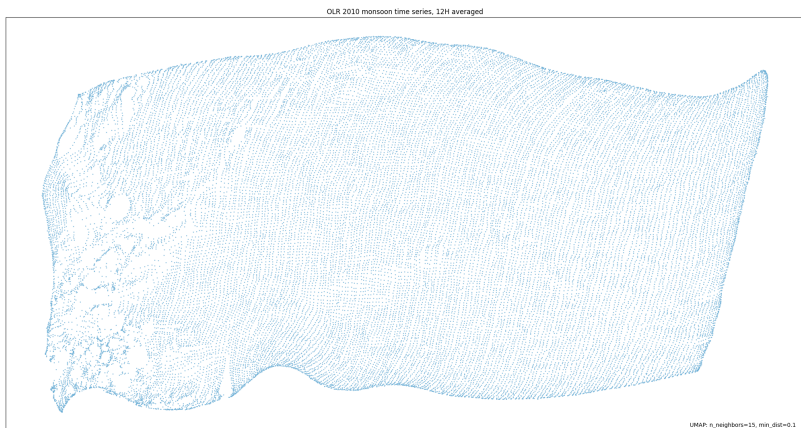


(c)

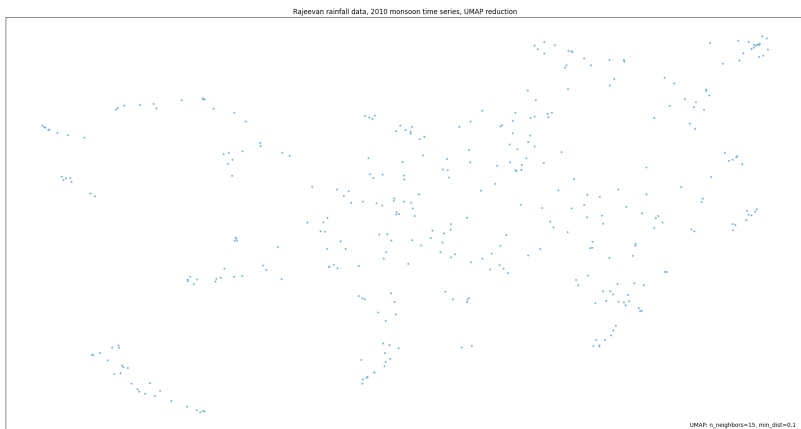


(d)

Figure 3.11: Histograms of pairwise distances for (a) OLR time-series data, L1 norm distance function, (b) Rainfall time-series data, L1 norm distance function, (c) Concentric annuli, Euclidean distance, and (d) Synthetic data containing 5 well-separated balls each containing uniformly distributed points, Euclidean distance.



(a) OLR time-series data.



(b) Rainfall time-series data.

Figure 3.12: UMAP projections on \mathbb{R}^2

Chapter 4

Conclusion

During the project, we have had some success in finding some properties of clustering parameters like σ and k . We have also successfully used these properties to detect geological and climate features without using any theory from Earth science, which would not have been possible using common clustering algorithms such as K-means. We list here all the conclusions we have drawn from the project results.

4.1 Parameter σ

While using the Gaussian similarity function (eq 2.1), the value of σ can be initially estimated using a thumb rule given in Luxborg (2007) [1]. After that, we need to vary σ to vary the location of the eigengap of the Laplacian. We have discovered a relationship between the eigenvalues and the σ parameter. For the k -th eigenvalue of the Laplacian, increasing the σ decreases the eigenvalue. And the magnitude by which this decrement happens appears to be increasing with k for smaller values of k . And since we do not want too many clusters with too few data points, we are only concerned with relatively small values of k .

4.2 Spectral clustering has detected geological and climate features.

Using Luxborg's thumb rule [1] and manipulation of connectivity parameters and Laplacian's eigenvalues, the spectral algorithm has found the following features:

1. The fault lines between Earth's tectonic plates using latitudes and longitudes of recorded earthquakes.
2. The monsoon days separated from the rest of the year using OLR maps dataset.
3. The high rainfall areas separated from the rest of India using rainfall time-series dataset.

4.3 OLR time-series data could not be clustered.

None of the methods worked on OLR time-series data to find any meaningful clusters. Extreme manipulations of the Laplacian's eigenvalues did give a clustering, but it was not significant and mostly arbitrary. On the other hand, rainfall time-series data was successfully clustered.

4.4 Future Work

- More work is needed in order to find the rate of change in eigenvalues of the Laplacian as we change the parameter σ .
- The slope constraints for the DTW algorithm can be varied and the result might have more meaningful clusters as the constraints are eased.
- To find the dimension of the underlying manifold in a high-dimensional dataspaces, we need to find a better coefficient than the Pearson's correlation coefficient to be used with UMAP.

References

- [1] Ulrike von Luxburg. *A Tutorial on Spectral Clustering*. 2007. arXiv: 0711.0189 [cs.DS]. URL: <https://arxiv.org/abs/0711.0189>.
- [2] Dorothea Wagner and Frank Wagner. “Between min cut and graph bisection”. In: *18th International Symposium on Mathematical Foundations of Computer Science*. Springer. 1993, pp. 744–750.
- [3] Russell Merris. “Laplacian matrices of graphs: a survey”. In: *Linear Algebra and its Applications* (1994), pp. 143–176. URL: <https://api.semanticscholar.org/CorpusID:205010266>.
- [4] Francis R. Bach and Michael I. Jordan. “Learning Spectral Clustering, With Application To Speech Separation”. In: *Journal of Machine Learning Research* 7.71 (2006), pp. 1963–2001. URL: <http://jmlr.org/papers/v7/bach06b.html>.
- [5] Danielle Middlebrooks. *Application of the Spectral clustering algorithm*. 2016.
- [6] Zakariyaa Ait El Mouden, Abdeslam Jakimi, and Moha Hajar. “An application of spectral clustering approach to detect communities in data modeled by graphs”. In: *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*. NISS '19. Rabat, Morocco: Association for Computing Machinery, 2019. ISBN: 9781450366458. DOI: 10.1145/3320326.3320330. URL: <https://doi.org/10.1145/3320326.3320330>.
- [7] Mohd Ansari et al. “Spatiotemporal clustering: a review”. In: *Artificial Intelligence Review* 53 (Apr. 2020). DOI: 10.1007/s10462-019-09736-1.

- [8] Earthquake dataset. URL: <https://www.kaggle.com/datasets/warcoder/earthquake-dataset>.
- [9] M. Rajeevan et al. “A High Resolution Daily Gridded Rainfall Data for the Indian Region: Analysis of break and active monsoon spells”. In: *Curr Sci India* 91 (Aug. 2006).
- [10] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
- [11] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. “On spectral clustering: analysis and an algorithm”. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. NIPS’01. Vancouver, British Columbia, Canada: MIT Press, 2001, pp. 849–856.
- [12] Xin Jin and Jiawei Han. “K-Means Clustering”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 563–564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425. URL: https://doi.org/10.1007/978-0-387-30164-8_425.
- [13] G.W. Stewart and J. Sun. *Matrix Perturbation Theory*. Computer Science and Scientific Computing. Elsevier Science, 1990. ISBN: 9780126702309. URL: <https://books.google.co.in/books?id=178PAQAAMAAJ>.
- [14] R. Bellman and R. Kalaba. “On adaptive control processes”. In: *IRE Transactions on Automatic Control* 4.2 (1959), pp. 1–9. DOI: 10.1109/TAC.1959.1104847.
- [15] Pavel Senin. *Dynamic Time Warping Algorithm Review*. Jan. 2009.
- [16] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49. DOI: 10.1109/TASSP.1978.1163055.
- [17] Giorgino Toni. “Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package”. In: *Journal of Statistical Software* 31 (Aug. 2009). DOI: 10.18637/jss.v031.i07.
- [18] Saunders MacLane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics, Vol. 5. New York: Springer-Verlag, 1971, pp. ix+262.

- [19] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML]. URL: <https://arxiv.org/abs/1802.03426>.
- [20] Michael Steinbach, Levent Ertöz, and Vipin Kumar. “The Challenges of Clustering High Dimensional Data”. In: *Univ. Minnesota Supercomp. Inst. Res. Rep.* 213 (Jan. 2003). DOI: 10.1007/978-3-662-08968-2_16.