Spectral Clustering via Diffusion Maps: Algorithm Optimization and Applications in Predictive Analysis of Weather Patterns

Branden Keck, Johns Hopkins University

**Abstract**—Weather forecasting is a very common problem. Given the abundance of complicated, intricate weather forecasting models, it is of interest to determine if a simpler approach could provide results that are still meaningful. One potential approach is the use of a Spectral Clustering algorithm - implemented via Diffusion Maps - that can be used to analyze radar data. This approach is utilized along with naïve predictive analysis algorithms to predict the evolution of a basic data set. The potential for this approach to be extended to more complicated, computationally taxing data sets is then discussed and analyzed.

**Index Terms**—Diffusion Maps, Spectral Clustering, Nystrom Approximation, Stochastic Differential Equations, Weather Forecasting, Predictive Analysis

—————————— ◆ ——————————

# 1 Introduction

Weather analysis and forecasting is a discipline that has been at the forefront of human curiosity since the time of the Babylonians. Due to their wide array of applications and importance in daily life, weather forcasting models are continually improved and tested by researchers around the globe [1]. Many of these models are very complicated given the number of variables necessary to make an accurate prediction. Additionally, these models are difficult to understand given the number of complex meteorological concepts that are often incorporated in them.

Spectral Clustering via Diffusion Maps can provide a less-complicated solution to the problem of predictive analysis. However, given the number of computations required to compute the Diffusion Map for reasonably sized data, such calculations can become computationally taxing. In this paper, small images used to determine the viability of predicting the evolution of two-dimential image data sets. Then, optimations are introduced in an attempt to extend these principals to a larger set of data.

In the remainder of this introductory section, the theory behind various mathematical and computational techniques is described. In the following two sections, specific computational methods and the results of those methods are described. Finally, conclusions are drawn about the overall feasibility of this approach to predictive data analysis.

## 1.1 Diffusion Maps

Diffusion maps are a dimensionality reduction technique which is beneficial in data analysis due to its applicability to nonlinear data. Additionally, Diffusion Maps are commonly used to discover underlying properties of a data set that would not necessarily be revealed by other dimensionality reduction techniques [2]. In terms of data clustering, Diffusion Maps provide the added benefit on introducing parameters that can be used to adjust clustering criteria until optimal data clusters are produced. These benefits are described in more detail in a later section.

The underlying theory of the Diffusion Map algorithm is rooted in stochastic calculus. The primary principal of a Diffusion Map is that the connectivity between data points is determined by the probability of transitioning from one data point to another in a random walk of “t” time steps. As such, the Diffusion Map can be parameterized with respect to time, where an increase in the time, “t”, results in a corresponding increase in the probability of transitioning between any two points in the dataset. Therefore, as time increases, the Diffusion Map becomes less sensitive to differences between the data points [2].

The first step in implementing a Diffusion Map is to develop a kernel. The kernel acts as a measure of similarity between two data points and is often denoted as follows [3]:

The measure of “similarity” between two data points can be defined in a variety of ways. A common similarity measure is Euclidean distance. However, sometimes overall similarity should account for other data attributes such as color of pixels in an image [4]. The previously mentioned probability of transitioning between data points is commonly calculated using the kernel scaled by a normalization constant [2].

After defining a kernel function, the next step in constructing the Diffusion Map is to create an affinity matrix based on this function. Given a set of “n” data points, the affinity matrix will be of size “n by n” with the kernel being calculated for every possible set of points in the dataset. Using the notation proposed by Farbman et. al., the affinity matrix will be denoted as matrix “W” with elements given by [4]:

with xi and xj being data points. The affinity matrix is the most computationally expensive component of this predictive analysis model. As such, approximation techniques were implemented to lessen the burden of computing this matrix. These methods will be described in the next section.

The next data structure required to compute the Diffusion Map is the “all-pairs normalized affinity matrix”, denoted by “M”. While “W” was a good metric of the distance between points, “M” is a stochastic matrix that represents the set of probabilitys of transitioning between any pair of points in the dataset. This matrix is calculated via the following formula [4]:

Here, “D” is the matrix formed by diagonalizing the row sums of matrix “W”. As such, the matrix “D” and subsequently matrix “M” are both “n by n” in size where “n” is the size of the original dataset.

The final Diffusion Map computation is completed by taking the the eigenvectors and eigenvalues of the all-pairs normalized affinity matrix, where each column of the Diffusion Map is given by [2]:

In this formula, each “Y” is a column of the Diffusion Map where “m” columns are selected to create a map with minimal size to achieve the desired information about the underlying data manifold [3]. and represent eigenvalues and eigenvectors of matrix “M”, respectively. The index “i” represent the element index within the given eigenvector. Additionally, the exponent “t” on the eigenvalues in this computation is the time parameter, which was described previously.

## 1.2 K-Means Clustering

K-Means Clustering is a very common data analysis technique. The purpose of this technique is to categorize “unlabeled” data. The variable “k” represents the number of categories into which the data will be partitioned [6]. By performing K-Means Clusting, it is possible to draw conclusions about different subsets of data such as which pixels in an image correspond to objects that are in the “foreground” and which pixels correspond to “background” objects.

The K-Means Clustering algorithm is comprised of several basic steps. An overview of this algorithm is as follows [6]:

The term “Spectral Clustering” refers to a specific type of clustering in which the “spectrum” (eigenvectors) of a graph Laplacean is used instead of the explicit coordinates of the datapoints. There are several benefits to using “spectral” K-Means clustering over “traditional” K-Means clustering. The primary benefit of the spectral method is enhanced performance (i.e. data is more accuarately “clustered”). Additionally, spectral clustering provides an opportunity to envoke computationally efficient linear algebra methods without adding much complexity to the algorithm itself [7].

Diffusion Maps lend themselves very naturally to spectral clusting because the elements of the Diffusion Map are already meaningful “spectral” functions the are constructed based on the coordinates of the orginal datapoints. In this way, the Diffusion Map is very similar to a graph Laplacean and can be used in the same manner. To do so, the “n” rows (where “n” is number of original datapoints) of the Diffusion Map are used as spectral coordinates in the K-Means algorithm.

## 1.3 Nyström Approximation

As previously mentioned, a benefit of spectral clustering over traditional methods is the efficiency that can be achieved by envoking linear algebra concepts. Diffusion Map calculations require an abundant amount of computational resources, especially for large image data sets. It is possible to limit the resource requirements by applying the Nyström approximation to the pairwise affinity matrix, W, instead of explicitly calculating kernel values for each element of the matrix. Several computational pathways for this approximation are presented by Fowlkes et. al. [5], the most basic of these pathways being an estimation of W using smaller matrices in the form:

Given an affinity matrix of size “n by n”, A is an “m by m” matrix of “sample” affinities and B is an “(n-m) by m” matrix of affinities between the remaining data points and those in the sample. W can then be completed by estimating the remaining portion of the matrix, C, via the following formula [5]:

Thus, a very large problem (“n2“ kernal computations) is reducable to something more manageable (“nm” kernel computations).

## 1.4 Predictive Analysis

asdf

# 2 Methods

A predictive analysis model based on these techniques was created using the python programing language. All methods were combined into a single .py file and this code as well as the data sets used to test to model can be found on github via the following link:

github.com/BrandenKeck/JHU\_625.714\_Final\_Project

## 2.1 Data Set

Three primary data sets were used to develop and test this model. All of these data sets were collections of images. The first set consisted of images of varying sizes. This set was used to test the speed of the model as well as some of the initially developed clustering algorithm code. It can be seen as follows in Fig. 1:

The other two data sets used to test the model were series of evolving images. These images were used to determine the accuracy by which the model predicts movements of data clusters. These are shown in Fig. 2:

It can be seen that the first image set in the above figure is

## 2.2 Development of Python Model

A python script was developed to p

The kernel developed for this model is similar to that developed by Farbman et. al. [4] and is based on the RBF kernel as used by the scikit-learn python library:

*)*

where is a scaling parameter, and are data points.

## 2.3 Efficiency Improvement

Also drawn from the Farbman et. al paper [4], is a well-known efficiency improvement that can be implemented via the Nyström approximation.

Additionally, the ‘numpy’ and ‘tensorflow’ python libraries were used to handle all linear algebra computations. Since these libraries have been optimized for these types of operations, it is worthwhile to note that they were levaraged to improve upon the inefficiency of naïve matrix operations.

# 3 Results

Additionally

## 3.1 Parameter Selection

Additionally

## 3.2 Simple Spectral Clustering Test

Additionally

## 3.3 Simple Predictive Analysis Test

Additionally

## 3.4 Complex Predictive Analysis Test

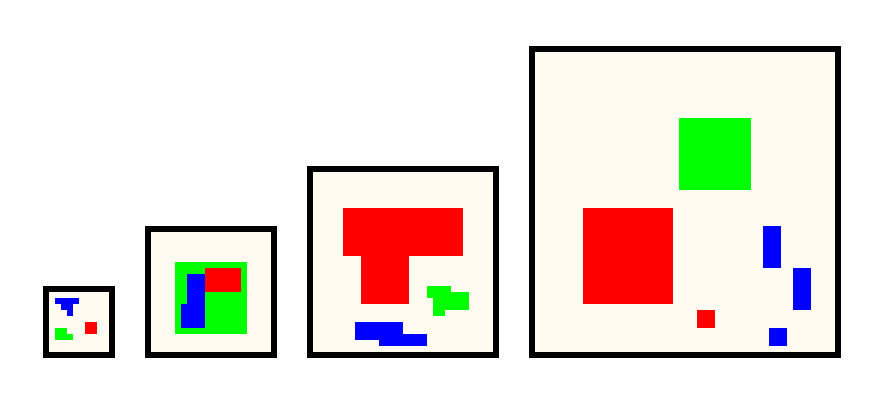
Additionally

# 4 Conclusion

A predictive analysis model that utilizes Diffusion Mapping, a Nystrom approximation method, Spectral Clustering and other computational techniques was presented as a potential method for very basic weather forecasting.

**References**

1. Steve Graham, et al. “Weather Forecasting Through the Ages.” *NASA, Earth Observatory.* 25 February 2002. Web. 18 August 2018. https://earthobservatory.nasa.gov/Features/WxForecasting/wx2.php
2. J. de la Porte, et. al. “An Introduction to Diffusion Maps.” *Techniques.* 2008.
3. Ronald R. Coifman and Stephane Lafon. “Diffusion Maps.” *Applied and Computational Harmonic Analysis.* 2 April 2006.
4. Zeev Farbman, et. al. “Diffusion Maps for Edge-Aware Image Editing.” *ACM Transactions on Graphics.* 2010.
5. Charless Fowlkes, et al. “Spectral Grouping Using the Nyström Method.” *IEEE Transactions on Pattern Analysis and Machine Intelligence.* February 2004.
6. Andrea Trevino. “Introduction to K-means Clustering.” *Oracle Datascience.* 6 December 2016. Web. 21 August 2018. https://www.datascience.com/blog/k-means-clustering
7. Ulrike von Luxburg. “A Tutorial on Spectral Clustering.” *Statistics and Computing.* 2007.
8. Mubaris NK. “K-Means Clustering in Python.” 1 October 2017. Web. 16 August 2018. https://mubaris.com/2017/10/01/kmeans-clustering-in-python/



**Fig. 1**: First data set. Includes small test images of sizes:

a) 10x10 pixels b) 20x20 pixels c) 30x30 pixels d) 50x50 pixels.

(note: images enlarged in this figure):