Problem Set 1: Dimensionality Reduction via PCA and Diffusion Maps

Unsupervised Learning for Big Data
CPSC/AMTH 453 / CBB 555 / CPSC/AMTH 553 / GENE 555 / NSCI 453

Assigned:September 14
Due: October 6, 11:59pm
2021

1 Introduction

Real world data sets - such as those obtained by high-throughput drug screening or gene sequencing - are massive, containing hundreds of thousands of measured variables of tens of thousands of observations[1]. These data sets often require exploratory analysis. Common questions include

1. Which measured variables are important in understanding my data set (e.g. explain the variance or other statistics of observed data points) and which variables are not so important?
2. Do my observed data points form natural clusters and, if so, which observed data points cluster together and which measured variables best describe these clusters?
3. How can I visualize my observed data points to better understand these relationships and develop a deeper intuition for my data set (whether it be a measured phenomenon or controlled experiment)?

Somewhat ironically, as we develop technologies to measure a larger amount of variables (i.e. when p is large and especially when $p \gg n$), analyzing this high dimensional data and answering these questions above become much more complicated, both statistically and algorithmically. This is known as the *curse of dimensionality.*

In class, we've discussed *dimensionality reduction* techniques to reduce the dimension of data sets from p to $d \ll p$ while (approximately) preserving various structural properties of the data set. Such techniques include Principal Component Analysis (PCA), t-SNE, and Diffusion Maps. Aside from their obvious algorithmic differences, each of these techniques seeks to preserve different properties of the data set and has different mathematical motivations and assumptions. Once we develop a dimensionality reduction technique, how do we - as data *scientists* - experimentally validate these methods? Generally speaking, we first test our method on artificially constructed data sets that we understand either geometrically or statistically (e.g. a low-dimensional point cloud, or points drawn from a known distribution) and then see if our method confirms the known structure in our data set. The next step is to test our method on a real data set that we understand biologically and see if our method confirms the biological understanding.

2 Understanding the Data Set

Use the provided function `read_json_files()` to read the two JSON files containing the data, `swiss_roll_points.json` and `swiss_roll_labels.json`. The `swiss_roll_points.json` contains 2000 points, each with 3 features and `swiss_roll_labels.json` contains labels for each of the 2000 points, ranging from 0 to 1. Pre-process the data by centering it (i.e. data points have zero mean). Next, visualize this data set using a scatter plot where the provided features are x, y, and z coordinates and each point is colored by its label.

[1] In this class, we adopt the convention of denoting the number of observed data points by n and denoting the number of measured variables by p. Thus, a data matrix X whose rows are observations and whose columns are variables has size $n \times p$.

Question 2.1. *What does this visualization of the swiss roll data set look like? What properties do you notice about the data set? Are the provided labels meaningful, and if so, in what way? How do you expect a "good" dimensionality reduction technique to look for the swiss roll data set?*

2.1 Visualizing Data with PCA

Now that we have an understanding of the swiss roll data set and how we expect a successful dimensionality reduction to look, it's time to experiment. We'll start with PCA.

1. Run PCA on the swiss roll data set, obtaining the principal components, projections, and singular values
2. Plot the swiss roll in two dimensions, using the principal components. You might wish to try different combinations, since there are few of them (i.e. the first and second components, the first and third components, the second and third components). Color the points with their corresponding labels.
3. Plot the singular values in whatever way you see fit. This could be a plot of the singular values themselves, their cumulative sum, or the "explained variance" $S^2/(n-1)$, where S are the singular values and n is the number of points.

Question 2.2. *As a dimensionality reduction technique, to what extent does PCA retain properties of the swiss roll data set? Can you explain why the visualizations look like this, given how the algorithm works? What can you learn about the intrinsic dimensionality from the singular values?*

3 Implementing Diffusion Maps

In this assignment, you will implement a Diffusion Map and test this dimensionality reduction method on artificial and real data sets. All code will be written in Python. You are expected to submit a detailed write-up of your experiments, as well as your code, to the Canvas site by the due date of **October 6, 11:59pm** . The layout of the assignment is as follows: Section 3 describes the code specifications that you will write for implementing the Diffusion Map technique, as well as coding tips. Section 4 outlines the first experiment on the "Swiss Roll", a popular artificial data set, and Section 6 outlines the second experiment on a single-cell mass cytometry data set for iPSC Reprogramming. Section 7 contains the grading rubric and Section 8 contains the submission instructions.

3.1 Algorithm Review

The 2006 paper "Diffusion Maps" by Ronald Coifman and Stéphane Lafon [1] is the most cited paper with the Diffusion map algorithm, probably because it provides a very extensive analysis. However, the 2005 NeurIPS paper "Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators" is more user-friendly [2] and I recommend reading that paper for more clarification. Let's quickly review the algorithm now. Recall that for data points $X = \{x_1, \dots x_n\}$, kernel matrix W with entries defined by the kernel

$$\kappa(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right)$$

and diffusion parameter t, the Diffusion Maps algorithm creates an embedding of the points $\Psi_t : X \to \mathbb{R}^n$ based on the right eigenvectors of the powered Markov transition matrix M^t where $M = D^{-1}W$ and D is the diagonal matrix of row sums. More specifically, let $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ be eigenvalues of M with right eigenvectors $\psi_1, \psi_2, \dots \psi_n$.[2] Then the diffusion map is given by

$$\Psi_t(x_j) = \left(\lambda_2^t \psi_2(j), \lambda_3^t \psi_3(j) \dots \lambda_n^t \psi_n(j)\right)$$

[2]It shouldn't be immediately obvious that this non-symmetric matrix has positive real eigenvalues, with the largest one being equal to 1. This can be shown in a few ways, but the most popular way is through the Perron–Frobenius theorem. Proving this would be a good exercise in linear algebra.

. Recall that the eigenvector with the eigenvalue 1 is the constant vector; for this reason, we call it the trivial eigenvector and we discard it from our diffusion map since it reveals no information about our data. As it turns out, computing the eigenpairs λ_i, ψ_i can be done by first computing the eigenpairs of the symmetric matrix $M_s = D^{-1/2}WD^{-1/2}$, denoted $\tilde{\lambda}_i$, v_i and then taking $\lambda_i = \tilde{\lambda}_i$ and $\psi_i = D^{-1/2}v_i$. For somewhat technical reasons, this method is more efficient and numerically stable because M_s is a real symmetric matrix. It is a good exercise to show that, indeed, the eigenvalues of M and M_s are the same and that $D^{-1/2}v_i$ is a corresponding eigenvector ψ_i of M, as advertised. We now summarize the Diffusion Maps algorithm below.

Algorithm 1: Diffusion Maps Algorithm

```
  Input: data set $X = \{x_1, \dots x_n\}$, kernel function $\kappa(;)$, diffusion parameter $t$
  Output: diffusion mapping $\Psi_t : X \to \mathbb{R}^n$
1 Construct kernel matrix $W$, $W_{i,j} = \kappa(x_i, x_j)$;
2 Construct symmetric matrix $M_s = D^{-1/2}WD^{-1/2}$;
3 Compute eigenpairs $(\lambda_1, v_1), \dots (\lambda_n, v_n)$ of $M_s$;
4 Compute (normalized) eigenvectors of $M$ as $\psi_i = D^{-1/2}v_i / \|D^{-1/2}v_i\|$;
5 Construct diffusion map $\Psi_t(x_j) = (\lambda_2^t \psi_2(j), \lambda_3^t \psi_3(j) \dots \lambda_n^t \psi_n(j))$;
6 return $\Psi_t$
```

3.2 Implementation Details

This assignment will be written using Python. For array computations, you should use the `numpy` package. You may find the following commands particularly useful (some of these are in the `numpy.linalg` subpackage): `sum`, `std`, `exp`, `abs`, `zeros`, `dot`, `norm`, `power`, `svd`, and `eigh`. See the `numpy` documentation online for detailed function descriptions. For plotting, please use the `matplotlib` package, as it is easy to use and interfaces well with `numpy` arrays.

I have provided you with a skeleton file, `ps1_functions.py`, that contains the following empty functions which you should complete. The purpose of each function is briefly stated in the function header, but you should review the Diffusion Maps method to remind yourself how these functions should fit together. The distance matrix calculation and the eigendecomposition are the most computationally expensive components of the algorithm; we have designed the functions in the skeleton file accordingly. The skeleton file `ps1_functions.py` contains a function `read_json_files()`, which can be used for loading data. You may add more functions to `ps1_functions.py` if you feel that they will help you in your experiment, but these additional functions will not be tested by the grader. You should create new Python files for running your experiments, outlined in Sections 4 and 6. In these experiment files, you may call functions defined in `ps1_functions.py` using the `import` command in Python. All code should be sufficiently commented so that someone who understands diffusion maps [3] can easily understand your code.

In Lectures, we introduced PCA as providing p linear combinations of features that best explain the variance. To obtain the projected data points, we can evaluate the linear combinations of the features of each data point; however, this is not the most efficient computational method. We mentioned that the Singular Value Decomposition (SVD), which decomposes a data matrix $X = USV^T$ is an appropriate way to obtain both the linear combination of features or loadings (which are given in V) *and* the projection of the data points. (which are given as US). When writing your code, you should always center the data and compute its SVD rather than constructing the covariance matrix and computing its eigendecomposition. For computing diffusion maps, it is best to use `eigh`, for reasons described above.

4 Experiment 1: Swiss Roll Dataset

The "swiss roll" data set is a commonly used synthetic data set to test embedding algorithms. The swiss roll is a good test case because it consists of points sampled from a 2-dimensional manifold. Without getting overly technical, a set $X \subset \mathbb{R}^n$ is a *2-dimensional manifold* if at every point $x \in X$, a neighborhood around x looks just like a section of $\mathbb{R}^2$. Geometrically, you can think about this as a 2 dimensional sheet in 3 dimensional space (or higher). Because the swiss roll has this kind of structure, it's easy for us to understand

[3]for instance, the course grader

what a "good" 2 dimensional embedding looks like. A picture would certainly be illuminating here and in fact, this will be your first task.

4.1 Visualizing Data with Diffusion Maps

Now we will look at diffusion maps as a dimensionality reduction technique. Recall that for each point x in our data set, $\Psi_t(x)$ is an n-dimensional vector [4] so when we visualize it, we can only visualize two or three dimensional slices. Your tasks will be to

1. Construct the diffusion map Ψ_t of the swiss roll data set using euclidean distance, Gaussian kernel with width $\sigma = 3.0$, and diffusion parameter $t = 1$.
2. Create two-dimensional scatter plots of the diffusion mapping using different coordinates.
3. Plot the eigenvalues λ_i of the Markov matrix M
4. Repeat for $\sigma = 1.0$ and $\sigma = 6.0$

Question 4.1. *As a dimensionality reduction technique, to what extent does diffusion mapping retain properties of the swiss roll data set? Can you explain why the visualizations look like this, given how the algorithm works? What can you learn about the intrinsic dimensionality of the data set from the eigenvalues of M? How does the choice of Gaussian kernel width σ change the embedding and why?*

5 Understanding the First Eigenvector of the Markov Matrix

Now let's see a curious property of the first left eigenvector of the Markov matrix. Your task will be to

1. Construct the affinity matrix of the swiss roll data set using euclidean distance and Gaussian kernel with width $\sigma = 1.0$.
2. Compute the largest left eigenvector ϕ_1 of $M = D^{-1}W$, which may be obtained as $\phi_1 = v_1 D^{1/2}$.
3. Plot the swiss roll in 3 dimensions using the original coordinates and color the points using the corresponding values in ϕ_1.

Question 5.1. *How do the values in ϕ_1 correspond to the structure of the swiss roll? Can you explain what you are seeing in terms of diffusion?*

5.1 Using an Adaptive Gaussian Kernel

We have seen the advantages of diffusion maps over PCA in the last portion of the experiment; however, the Gaussian kernel width σ was fixed for all data points. In this portion of Experiment 1, we'll see the effect of using an *adaptive* Gaussian kernel. To this end, we will use an adaptive *k-nearest neighbors kernel*, defined in the following way. Let the k-nearest neighbors kernel $\kappa_{k\text{-}nn}$ be given by

$$\kappa_{k\text{-}nn}(x_i, x_j) = \frac{1}{2}\left(\exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_k(x_i)^2}\right) + \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_k(x_j)^2}\right)\right)$$

where $\sigma_k(x)$ is the distance from x to its kth nearest neighbor. Note that $\kappa_{k\text{-}nn}$ satisfies the properties of a kernel. Let $\sigma_{k\text{-}nn} = \{\sigma_1, \ldots, \sigma_n\}$ denote these choices of adaptively chosen kernel parameters, for a fixed k. Now that we have defined this new adaptive kernel, let's see how it performs differently that a fixed-width kernel. Your tasks for this portion of Experiment 1 are

1. Construct the diffusion map Ψ_t of the swiss roll data set using euclidean distance, adaptive k-nearest neighbor Gaussian kernel with $k = 5$, and diffusion parameter $t = 1$.
2. Create two-dimensional scatter plots of the diffusion mapping using different coordinates.

[4] although in practice, it is common to only compute the first k dimensions. This is easily done with standard eigensolvers.

3. Plot the eigenvalues λ_i of the Markov matrix M

4. Repeat for $k = 10$

Question 5.2. *What are the differences between fixed and adaptive choices of kernel parameters σ? Explain this difference. What can you learn about the intrinsic dimensionality of the data set from the eigenvalues of M? How does the choice of nearest neighbor parameter k change the diffusion map? Which kernel method would you recommend using for the swiss roll data set and why?*

5.2 Changing the Diffusion Parameter t

So far, we have kept the diffusion parameter set to $t = 1$. In this portion of Experiment 1, we will observe the effects of changing t. Note that for observing different t, you can re-use the affinity matrix and eigendecomposition computations. To this end, your tasks are

1. Construct the diffusion map Ψ_t of the swiss roll data set using euclidean distance, adaptive k-nearest neighbor Gaussian kernel with $k = 10$, and diffusion parameter $t = 1$.

2. Create two-dimensional scatter plots of the diffusion mapping using different coordinates.

3. Plot the eigenvalues λ_i of the Markov matrix M

4. Repeat for $t = 10, 20, 50$

Question 5.3. *In terms of diffusion processes, what is the interpretation of increasing t? How do the diffusion embeddings visually change as t increases? How do the eigenvalues change and how does this help explain what you see in the embeddings? What can you learn about the intrinsic dimensionality of the data set from the eigenvalues of M as t increases? Is there a specific value of t that you find most informative for the Swiss roll data set?*

5.3 Final Thoughts

That's all of the computation we need for understanding these methods on synthetic data. Fortunately, you should be able to use all the code you've written for this experiment for the next experiment with real data. Before doing that, let's reflect on the data *science* aspect of our tests.

Question 5.4. *Suppose your boss is very delighted by your implementation of diffusion mappings and the initial tests on the swiss roll dataset; however, she would like to see how the method works on other artificial "control" data sets. Describe two other data sets that you might generate and what you would expect to see from these tests. You do not need to construct these two data sets.*

6 Experiment 2: iPSC Reprogramming Dataset

This is a mass cytometry dataset taken from [3] that shows cellular reprogramming from mouse embryonic fibroblasts (MEFs) to induced pluripotent stem cells (iPSCs) via the introduction of a transcription factor known as Oct4. This data set measures 33 proteins at the single-cell resolution. The protein markers measure

1. pluripotency or stem-ness, e.g. sox4 oct4

2. differentiation: e.g. nanog and lin28 which are embryonic stem cell (ESC) markers

3. cell-cycle: e.g. pRB which is a cell proliferation marker

4. signaling status: e.g. pakt which is a growth inducing signal and p53 which signals DNA damage and could induce cell cycle arrest. The "p" preceding some of these markers indicates that the measurement is for the phosphorylated (usually active) conformation of the protein

5. apoptosis: e.g. ccasp3 which is a caspase marker involved in programmed cell death

Additionally a "timepoint" marker shows the day after stimulation with reprogramming factors that the measurement was taken from. Here you have a mix of timepoints. These are low-granularity time points (on the order of days), while the actual transition is much finer grained. These protein markers enable the measurement of the reprogramming status. For instance, a subset of cells shows evidence of being early reprogramming intermediates with the "correct" set of reprogramming factors Sox2+Oct4+Klf4+Nanog+, another subset seems successfully reprogrammed with ESC-like lineages expressing markers such as Nanog, Oct4, Lin28 and Ssea1, and Epcam that are associated with transition to pluripotency.

As before, use the provided `read_json_files()` for reading the data set `ipsc_data_set.json` and the channel names `channel_names.json`. The original data set contained observed 220,450 time points; however, for computational purposes, we have subsampled the data set to contain 2,000 time points. The *timepoint* of data point is simply its row number. So the first timepoint is row one, the second timepoint is row two, and so on. A variety of other pre-processing steps have already been done for you.

6.1 Visualizing Data with PCA

Let's first explore how PCA can help us understand the processed data set. Your tasks for implementing PCA here are

1. Run PCA on the processed iPSC data set, obtaining the principal components, projections, and singular values.
2. Plot the iPSC data set in two dimensions, using the principal components. You might wish to try a few combinations of the top $k = 5$ principal components. When plotting the projections, color them using the time steps.
3. Plot the singular values in whatever way you see fit.

Question 6.1. *How do the PCA visualizations look? Are they capturing the time progression of the data set? Do you see any clusters forming? What is the intrinsic dimensionality given by PCA? What are the top few channels in the first and second principal directions?*

6.2 Visualizing Data with Diffusion Maps

Now let's do a similar analysis using Diffusion Maps and see what changes. Here, your task will be to

1. Construct the affinity matrix of the iPSC data set using euclidean distance and adaptive k-nearest neighbors Gaussian kernel with $k = 2$ and diffusion parameter $t = 1$.
2. Create two-dimensional scatter plots of the diffusion mapping using different coordinates.
3. Create three-dimensional scatter plot of the diffusion mapping using the first three coordinates for plotting. Color the points with their corresponding timepoint values.
4. Plot the associated eigenvalues of the Markov matrix.
5. Compute the top 5 channels that have the highest absolute correlation with the first, second, and third diffusion components.

Question 6.2. *How do the Diffusion Map visualizations look? Are they capturing the time progression of the data set? Do you see any clusters forming? What is the intrinsic dimensionality given by Diffusion Maps? Which channels are most highly correlated with diffusion components? Any guesses to the biological interpretation? What are the main differences between Diffusion Maps and PCA methods on the iPSC data set?*

6.3 Changing Parameters

In this section, we'll examine how changing parameters affects the iPSC data set. Your tasks will be

1. Try two different values of diffusion parameter t and visualize the embeddings and the eigenvalues.
2. Try two different values of k for the adaptive kernel and visualize the embeddings and the eigenvalues.
3. For every choice of parameters above, compute the top 5 channels that have the highest absolute correlation with the first, second, and third diffusion components.

Question 6.3. *Did the choice of t affect the embeddings and, if so, how? Do the corresponding eigenvalues support your findings? How did the choice of k affect the embeddings and, if so, how? Did previously observed trends (i.e. clusters, time progression) in diffusion dimensions change dramatically? Did the correlated channels change with σ? How would you interpret this in terms of the data?*

7 Grading Rubric

7.1 Implementation – 20 points

Your implementation will be graded by running your submitted `ps1_functions.py` on several test cases. It is **very important** that you do not change the function input / outputs provided in the skeleton file. If you do, then your functions will not properly handle the test cases and you may receive no credit for the implementation portion of the assignment. Please adequately comment your code so that partial credit may be assigned in the event that your code does not pass all test cases.

7.2 Report – 80 points

You must write a detailed report about the experiments that you have run for this homework, including all plots and visualizations. Your report should address the questions raised here, but does not need to follow any particular structure or outline. Feel free to add any interesting insights that you had or extra experiments / validations you ran.

7.3 Cheating Policy

Plagiarism of any kind will not be tolerated in this course. Students are encouraged to discuss general ideas with each other, but must write code and carry out experiments individually. However, the course staff (professor, TA, etc.) is happy to provide assistance in all aspects of the assignment, from coding difficulties to interpretation of experiments. Any plagiarism - whether copying code or sharing experimental analyses - will not be tolerated.

8 Submission Instructions

Each student should submit a zip file titled `[last name]_[first name]_ps1.zip` to Canvas by **October 6, 11:59pm** , containing

1. A detailed write-up in pdf form, titled `[last name]_[first name]_ps1_report.pdf`
2. A subdirectory titled `code` containing all code used in the assignment
3. A subdirectory titled `figures` containing all figures used in the write-up

References

[1] Ronald R Coifman and Stéphane Lafon. “Diffusion maps”. In: *Applied and computational harmonic analysis* 21.1 (2006), pp. 5–30.

[2] Boaz Nadler et al. “Diffusion maps, spectral clustering and eigenfunctions of Fokker-Planck operators”. In: *Advances in neural information processing systems*. 2006, pp. 955–962.

[3] Eli R Zunder et al. “A continuous molecular roadmap to iPSC reprogramming through progression analysis of single-cell mass cytometry”. In: *Cell Stem Cell* 16.3 (2015), pp. 323–337.