Comp790-166: Computational Biology

Lecture 10

February 13, 2022

Good Morning Question

What do you think are some practical limitations for using PHATE on a large single-cell dataset?

Today

- Finish data augmentation strategy for sparse regions of the cellular landscape.
- Intro to GSP (graph signal processing)
- Identifying condition-specific prototypical cells

Welcome SUGAR

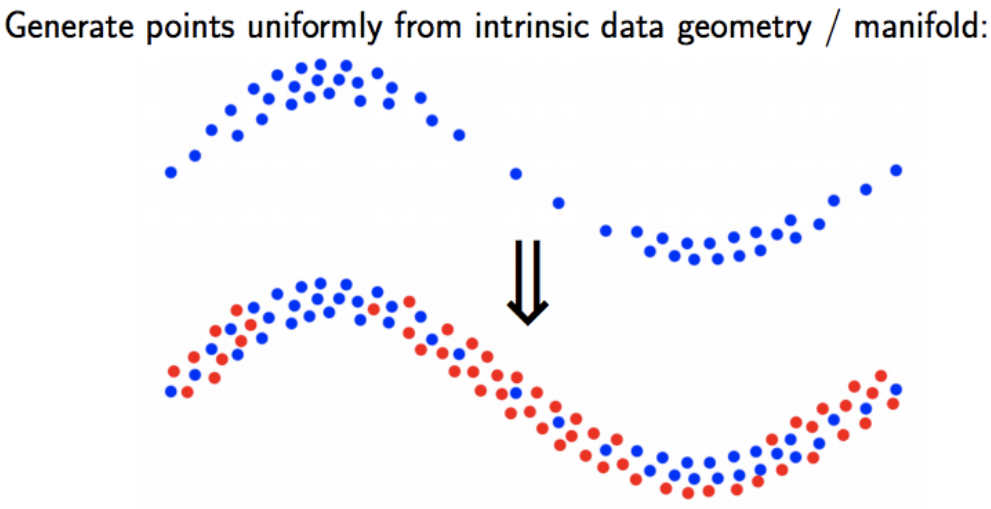


Figure: from Lindenbaum *et al.* NeurIPS. 2018

- Most generative modeling approaches seek to learn and replicate the data density
- SUGAR is a data-generation approach that uses the underlying geometry of the data (e.g. a random walk based approach)

General Problems with Downstream Tasks from Sparse Data

- Imbalanced classes can affect classification accuracy
- In clustering, imbalanced ‘ground-truth‘ clusters can cause distortion or mis-representation of the clusters in the data
- Too few samples/observations can cause mis-representation of the dependencies or correlations between features.

Problem Formulation for Data Generation Problem

- Let $\mathcal{M}$ be a d-dimensional manifold such that $\mathcal{M} \in \mathbb{R}^d$
- Let $\mathbf{X} \subset \mathcal{M}$ be a subset of points samples from $\mathcal{M}$ with $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N\} \in \mathbf{X}$
- Assuming that instances, $\mathbf{X}$ are unevenly sampled from $\mathcal{M}$, we seek a set of new points, $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_M\}$ that also lie of $\mathcal{M}$ and that $\mathbf{X} \cup \mathbf{Y}$ is uniform.

Synthesis Using Geometrically Aligned Random Walks

Algorithm 1 SUGAR: Synthesis Using Geometrically Aligned Random-walks

Input: Dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}, \boldsymbol{x}_i \in \mathbb{R}^D$.

Output: Generated set of points $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M\}, \boldsymbol{y}_i \in \mathbb{R}^D$.

1: Compute the diffusion geometry operators $\boldsymbol{K}$, $\boldsymbol{P}$, and degrees $\hat{d}(i)$, $i = 1, ..., N$ (see Sec. 3)
2: Define a sparsity measure $\hat{s}(i), i = 1, ..., N$ (Eq. 2).
3: Estimate a local covariance $\boldsymbol{\Sigma}_i$, $i = 1, ..., N$, using k nearest neighbors around each $\boldsymbol{x}_i$.
4: For each point $i = 1, ..., N$ draw $\hat{\ell}(i)$ vectors (see Sec. 4.3) from a Gaussian distribution $\mathcal{N}(\boldsymbol{x}_i, \boldsymbol{\Sigma}_i)$. Let $\hat{\boldsymbol{Y}}_0$ be a matrix with these $M = \sum_{i=1}^{N} \hat{\ell}(i)$ generated vectors as its rows.

Figure: from Lindenbaum *et al.* NeurIPS. 2018. Let's walk through steps 1-4 for now.

Synthesis Using Geometrically Aligned Random Walks

- **Specify Kernel:** Initialized by forming a Gaussian Kernel over the input data, $\mathbf{X}$, $\mathbf{G_x}$.
  - Use $\mathbf{G_x}$ to estimate the degree of each node i, $d(i)$.
  - The sparsity of each point, $s(i)$ is defined as the inverse degree of node i or $s(i) = 1/d(i)$.
- **Sample According to Each Point** Next, sample $\ell(i)$ points, $\mathbf{h}_j \in \mathbf{H}_i$ for $j = 1 \ldots \ell_i$ around each $\mathbf{x}_i \in \mathbf{X}$ from a localized gaussian distribution (e.g. k-nearest points around i).
  - $G_i = \mathcal{N}(\mathbf{x}_i, \Sigma_i)$

Practical : Sampling from MV Gaussians

numpy.random.multivariate_normal

random.multivariate_normal(*mean, cov, size=None, check_valid='warn', tol=1e-8*)

Draw random samples from a multivariate normal distribution.

The multivariate normal, multinormal or Gaussian distribution is a generalization of the one-dimensional normal distribution to higher dimensions. Such a distribution is specified by its mean and covariance matrix. These parameters are analogous to the mean (average or "center") and variance (standard deviation, or "width," squared) of the one-dimensional normal distribution.

Back to SUGAR

- Let $\mathbf{Y}_0$ be the set of all new $M = \sum_i \ell(i)$ points generated around each i, with $\mathbf{Y}_0 = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M\}$
- **MGC Kernel:** Now use affinities between points in $\mathbf{X}$ and $\mathbf{Y}_0$. Here, points in $\mathbf{X}$ are used as reference.

$$\hat{\mathcal{K}}(\mathbf{y}_i, \mathbf{y}_j) = \sum_r \mathcal{K}(\mathbf{y}_i, \mathbf{x}_r)\mathcal{K}(\mathbf{x}_r, \mathbf{y}_j)s(r) \tag{1}$$

Pulling $\mathbf{Y}_0$ towards sparser regions of $\mathcal{M}$

Pasted psuedo code for the second part of SUGAR

5: Compute the sparsity based diffusion operator $\hat{\boldsymbol{P}}$ (see Sec 4.2).
6: Apply the operator $\hat{\boldsymbol{P}}$ at time instant t to the new generated points in $\hat{\boldsymbol{Y}}_0$ to get diffused points as rows of $\boldsymbol{Y}_t = \hat{\boldsymbol{P}}^t \cdot \boldsymbol{Y}_0$.
7: Rescale $\boldsymbol{Y}_t$ to get the output $\boldsymbol{Y}[\cdot, j] = \boldsymbol{Y}_t[\cdot, j] \cdot \frac{\text{percentile}(\boldsymbol{X}_{[\cdot,j]},.99)}{\max \boldsymbol{Y}_{t[\cdot,j]}}$, $j = 1, \ldots, D$, in order to fit the original range of feature values in the data.

Figure: from Lindenbaum *et al.* NeurIPS. 2018.

Diffusion Operator Again

- Take affinities from $\hat{\mathcal{K}}$ and convert them to $\mathbf{P}$, the row-normalized Markov matrix.
- This will allow us to correct points in $\mathbf{Y}_0$ according to neighborhood regions

As you will all recognize, powering $\mathbf{P}$ as $\mathbf{P}^t$ estimates the probability of successfully traveling between nodes with t steps. The transformed matrix, $\mathbf{Y}_t$ is computed as

$$\mathbf{Y}_t = \mathbf{P}^t \times \mathbf{Y}_o \tag{2}$$

Same Story Regarding t

Remember in PHATE, t was chosen according to the knee point of the Von-Neumann entropy of the normalized eigenvalues of $\mathbf{P}^t$.

Experiments : Synthetic and Biological

Stanford Bunny

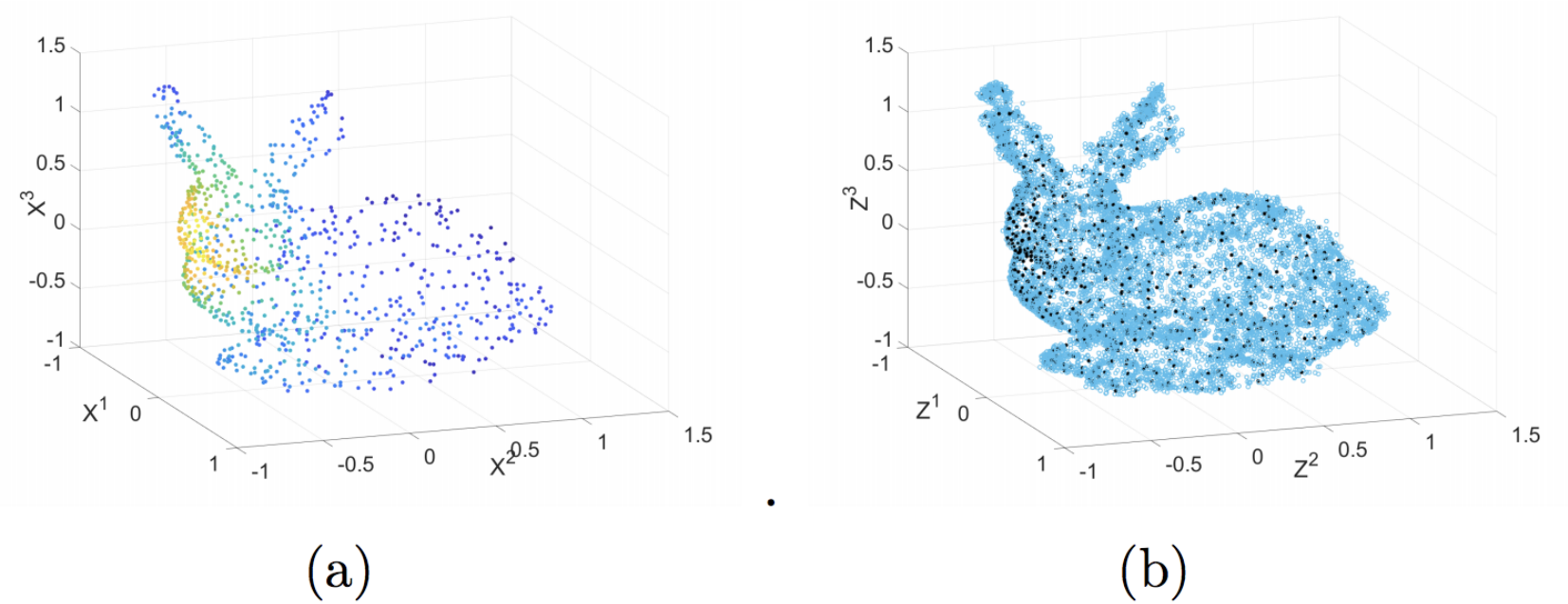


Figure: from Lindenbaum *et al.* NeurIPS. 2018. (**a**). The original set, $\mathbf{X}$ of points, colored by node degree. (**b**) $\mathbf{Y}_0$ are generated points (blue) and original points, $\mathbf{X}$ (black).

Stanford Bunny Part II

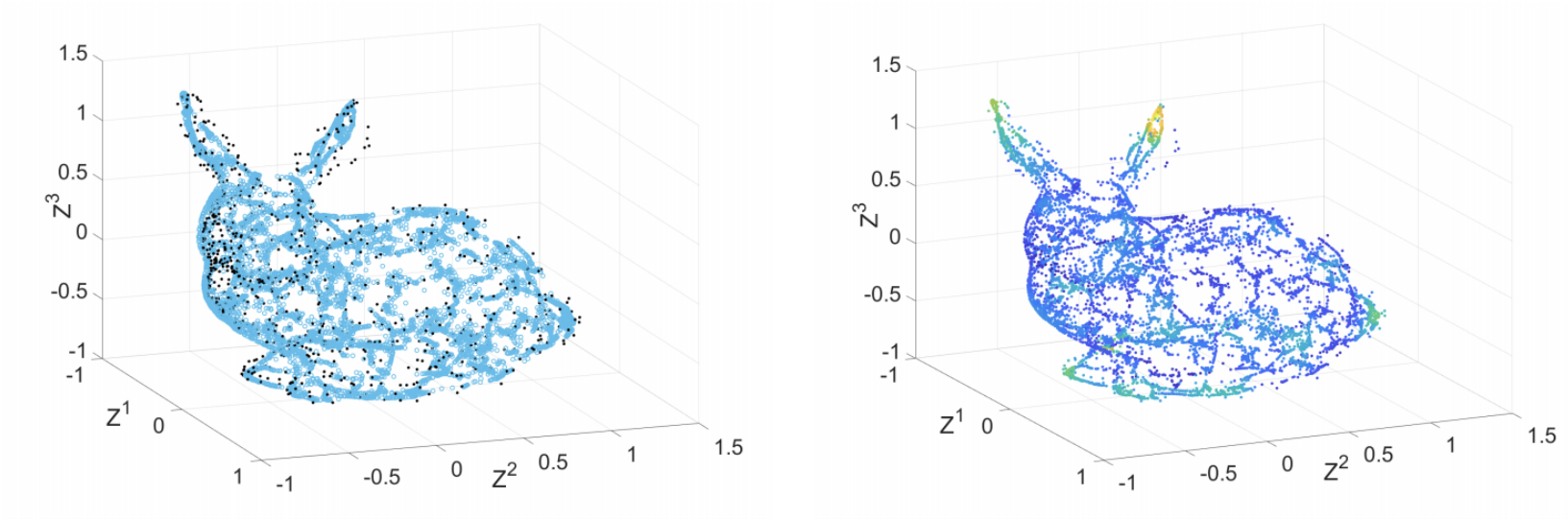


Figure: from Lindenbaum *et al.* NeurIPS. 2018. (**c**). Original and generated points, before MGC diffusion. (**d**) Generated points (**Y**) after MGC diffusion. Points are colored by degree.

Application : Augmented Clustering

SUGAR was used to generate additional data points to improve the quality of clusters identified with k-means.

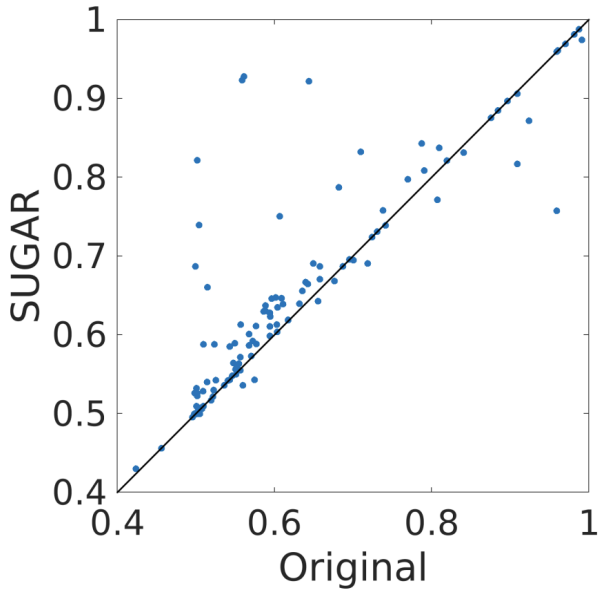


Figure: from Lindenbaum *et al.* NeurIPS. 2018. In 119 datasets, the data were augmented with additional datapoints using sugar. Adjusted Rand Index was computed for the original data vs data + SUGAR.

SUGAR on Single-Cell

SUGAR was using to augment cells in a single-cell RNAseq dataset.

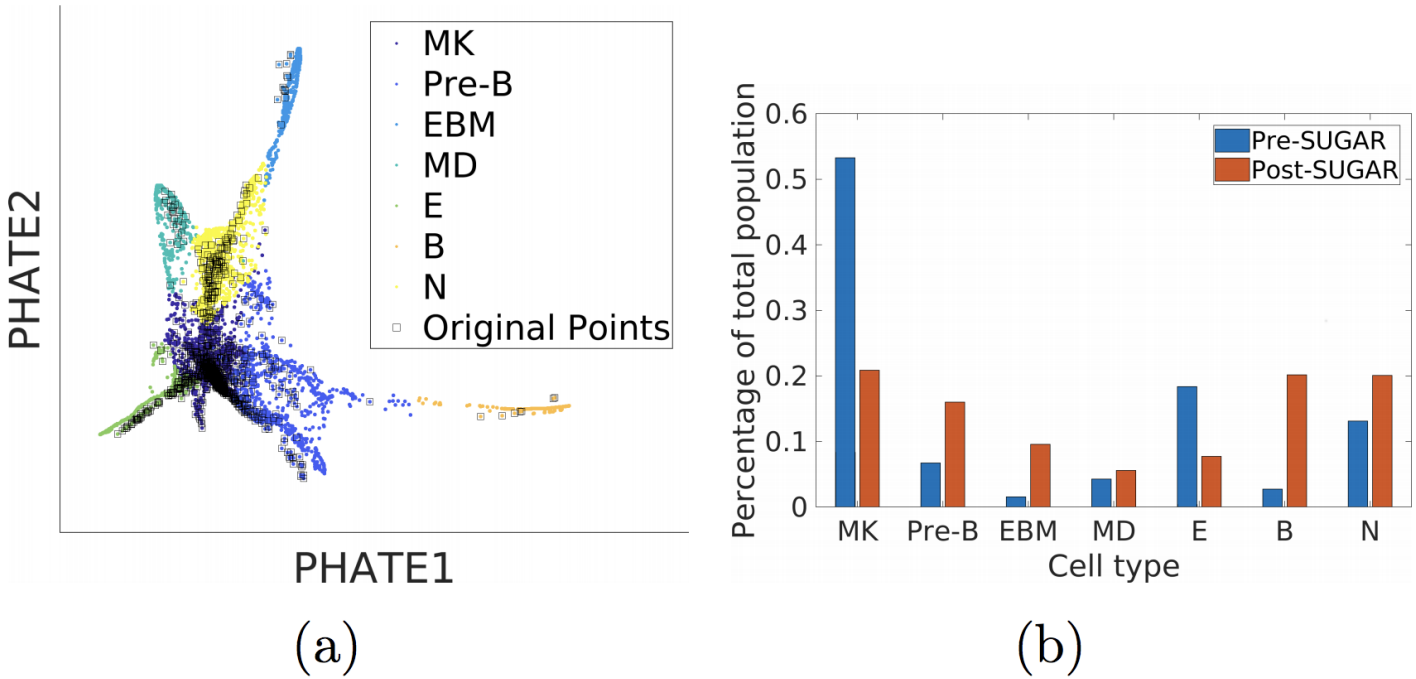


Figure: from Lindenbaum *et al.* NeurIPS. 2018

Maintaining Intra-Module Marker Co-Expression

Among cells assigned to the same module or cluster, after SUGAR lead to higher intra-module between-marker correlation.

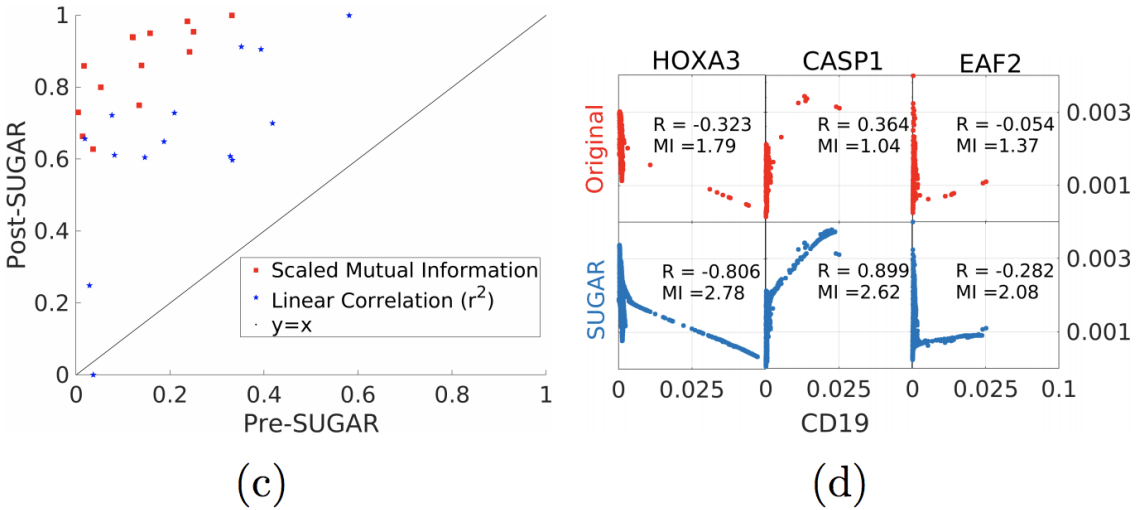


Figure: from Lindenbaum *et al.* NeurIPS. 2018

Graph Signal Processing

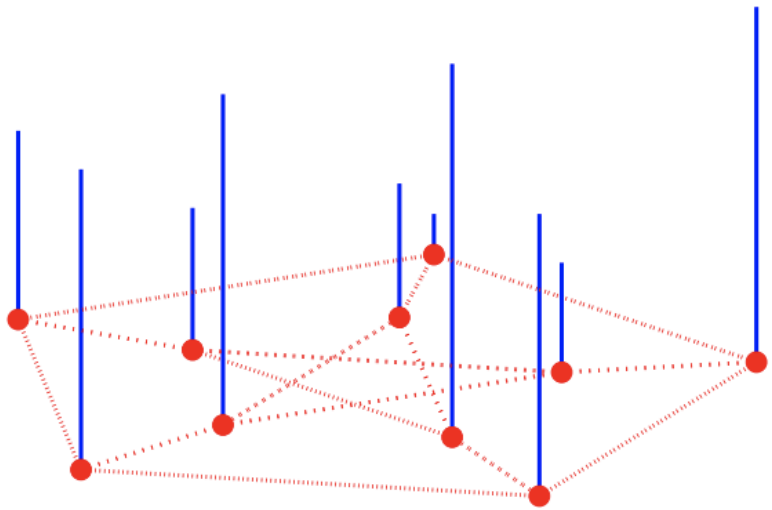
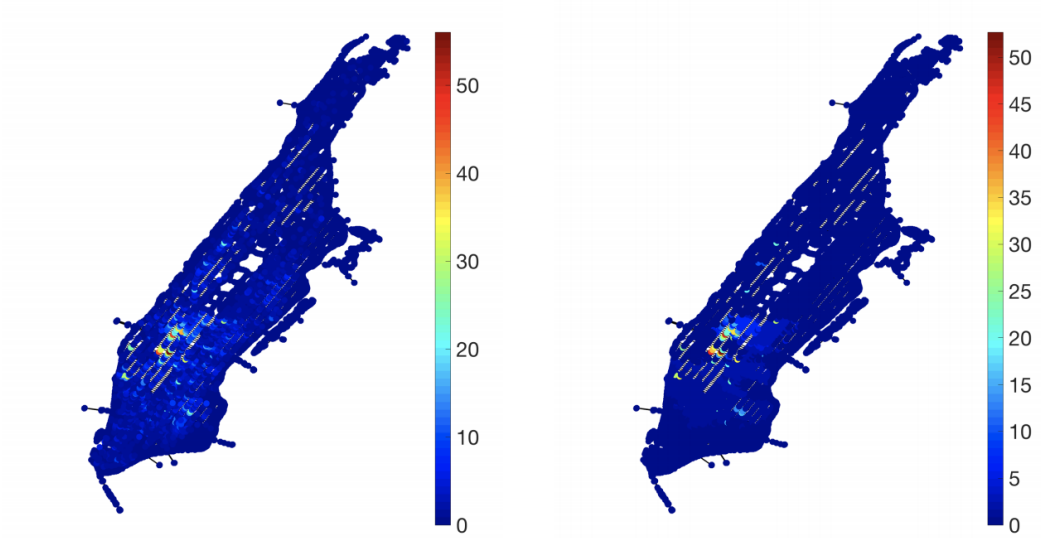


Figure: from Shuman *et al.* ArXiv. The purpose is to study the interplay between some signal and graph connectivity. Often we want to clean up or smooth out a particular signal, given the graph structure.

Smooth Signal Example

Translation: Nodes that are close (in terms of geodesic distance) on the graph should have similar signals. You can approximate the signal of a node, based on its neighbors.



(a) Actual data (b) Approximation

Figure: from `https://arxiv.org/abs/1712.00468`. Here the graph is street intersections in Manhattan and the signal is taxi pickups.

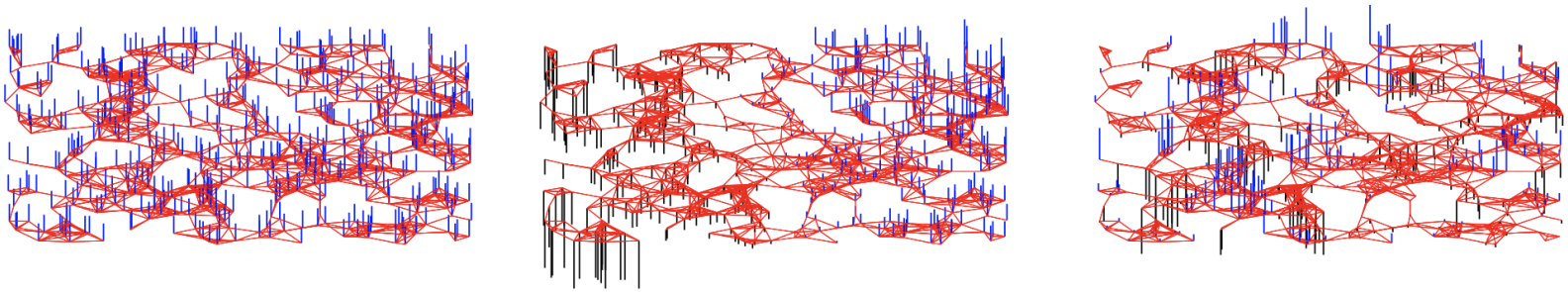
How Localized is the Signal?

Remember, our friend Graph Laplacian ($\mathbf{L} = \mathbf{D} - \mathbf{A}$),

- Some very nice theory falls out about the eigenvalues of the Laplacian matrix in terms of how 'localized' a graph signal, $\mathbf{f}$, is. For example $\mathbf{f}$ could be an expression of some protein.
  - First re-write $\mathbf{f}$ in terms of eigenvectors of the Laplacian
  - The eigenvectors corresponding to the first few eigenvalues of $\mathbf{L}$ are considered **low frequency**, and hence entries of the eigenvector entries corresponding to nodes that are connected should be similar
  - For higher **high frequencies** corresponding to 'later' eigenvalues, the values of the eigenvectors of adjacent nodes will be more different.

Signal Specificity

Here we visualize eigenvector entries at nodes ($\mathbf{u}_0$,$\mathbf{u}_1$,$\mathbf{u}_{50}$)



$\mathbf{u}_0$ $\mathbf{u}_1$ $\mathbf{u}_{50}$

Figure: from GSP Review https://arxiv.org/abs/1211.0053

Similarly

Zero crossings mean that eigenvector entries are neighboring nodes will be different.

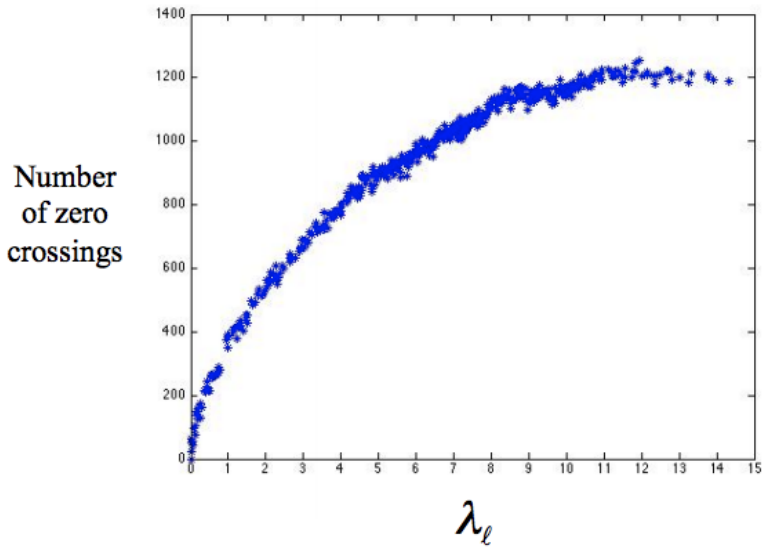


Figure: from GSP Review https://arxiv.org/abs/1211.0053

What is Graph Fourier Transform (on a high level?)

- Explain frequency content of the graph signal (e.g. experimental measurements/labels/etc) as a weighted sum of the eigenvectors of the Graph Laplacian
- The eigenvectors of the Graph Laplacian comprise the **Graph Fourier Basis and can help to decouple high and low frequency signals**

Local Variation of a Signal

The local variation of a signal or the sum of differences around a node can be written as,

$$(\mathcal{L}\mathbf{f})(i) = ([\mathbf{D} - \mathbf{A}]\mathbf{f})(i) \tag{3}$$

$$= d(i)\mathbf{f}(i) - \sum_j A_{ij}\mathbf{f}(j) \tag{4}$$

$$= \sum_j A_{ij}(\mathbf{f}(i) - \mathbf{f}(j)) \tag{5}$$

Local Variation Leads to Total Variation

The total variation of a signal on a graph is defined as follows and is also known as the Laplacian Quadratic Form

$$TV(\mathbf{f}) = \sum_{i,j} A_{ij}(\mathbf{f}(i) - \mathbf{f}(j))^2 \tag{6}$$

$$= \mathbf{f}^T \mathcal{L} \mathbf{f} \tag{7}$$

- Note here I have been assuming that we have an unweighted graph, but you could certainly substitute A_{ij} with a weighted version, W_{ij}

Getting to Graph Fourier Basis

- We can look at eigenvectors, $\mathbf{\Psi} = [\psi_1, \psi_2, \ldots, \psi_N]$ of $\mathcal{L}$
- and eigenvalues, $\mathbf{\Lambda} = [0 = \lambda_1 \leq \cdots \leq \lambda_N]$ of $\mathcal{L}$

The Graph Fourier Transform of a Signal

The Graph Fourier Transform ($\hat{\mathbf{f}}$) of a signal, $\mathbf{f}$ can be written as,

$$\hat{f}(\lambda_\ell) = \sum_i f(i)\psi_\ell^T(i) =< \mathbf{f}, \psi_\ell > \tag{8}$$

Said otherwise in matrix form as,

$$\hat{\mathbf{f}} = \boldsymbol{\Psi}^T \mathbf{f} \tag{9}$$

GFT Will Be Used to Filter

- A filter on the graph will take in a signal and attenuate it according to a frequency response function.
- **Low-Pass Filter:** We filter or preserve only frequencies corresponding to eigenvalues below some threshold, λ_k. So, consider frequencies λ_b, with $\lambda_b < \lambda_k$
- **High-Pass Filters**: Preserve only frequencies corresponding to eigenvalues above some threshold, λ_k. So, consider frequencies λ_b, with $\lambda_b \geq \lambda_{k+1}$

A Simple Low-Pass Filter

Define some filter h as,

$$h : [0, \max(\mathbf{\Lambda})] \rightarrow [0, 1] \tag{10}$$

Assuming the cutoff is λ_k,
$h(x) > 0$, for $x < \lambda_k$ and $h(x) = 0$, otherwise

Defining Notation

Define $h(\mathbf{\Lambda})$ as a diagonal matrix of eigenvalues with the filter applied.

Filtering a Signal Based on GFT

Based on what we computed with GFT, the filtered signal, $\hat{f_{filt}}$ can be computed as,

$$\hat{\mathbf{f}}_{filt} = h(\mathbf{\Lambda})\hat{\mathbf{f}} \tag{11}$$

Example in PyGSP

- Access PyGSP here, `https://pygsp.readthedocs.io/en/stable/tutorials/intro.html`

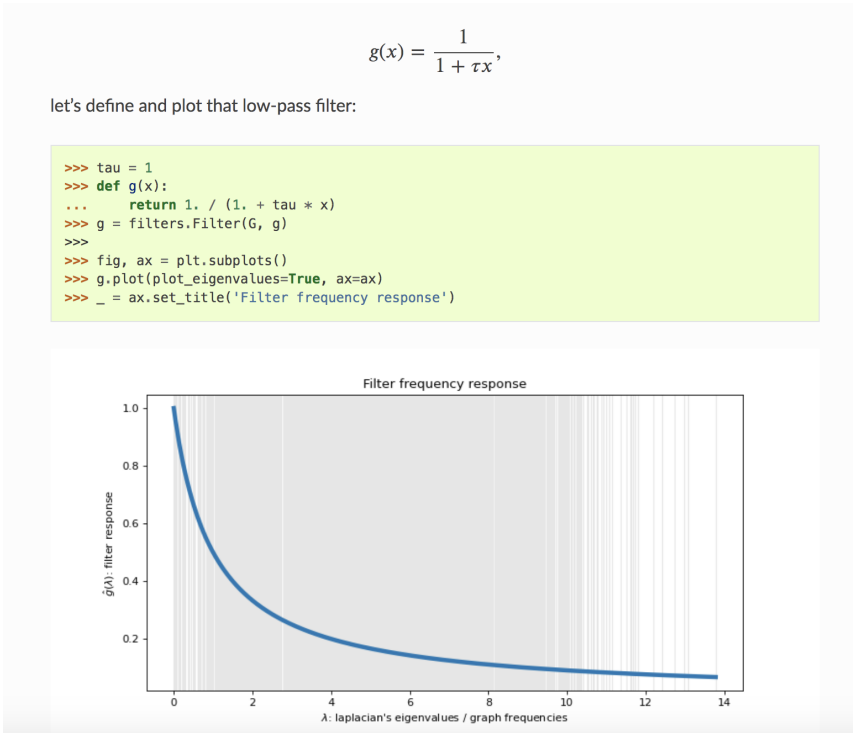
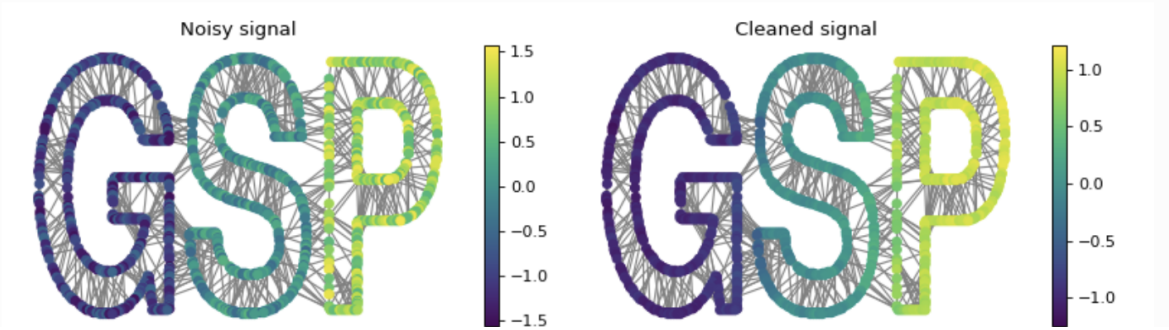


Figure: A simple filter for eigenvalues of **L**

Low-Pass Filtering a Noisy SIgnal

```
>>> s2 = g.filter(s)
>>>
>>> fig, axes = plt.subplots(1, 2, figsize=(10, 3))
>>> G.plot_signal(s, vertex_size=30, ax=axes[0])
>>> _ = axes[0].set_title('Noisy signal')
>>> axes[0].set_axis_off()
>>> G.plot_signal(s2, vertex_size=30, ax=axes[1])
>>> _ = axes[1].set_title('Cleaned signal')
>>> axes[1].set_axis_off()
>>> fig.tight_layout()
```



Linking Single Cell Data to External Information

MELD (an application of GSP!)

The idea of MELD is to model how experimental perturbation alters cell states.

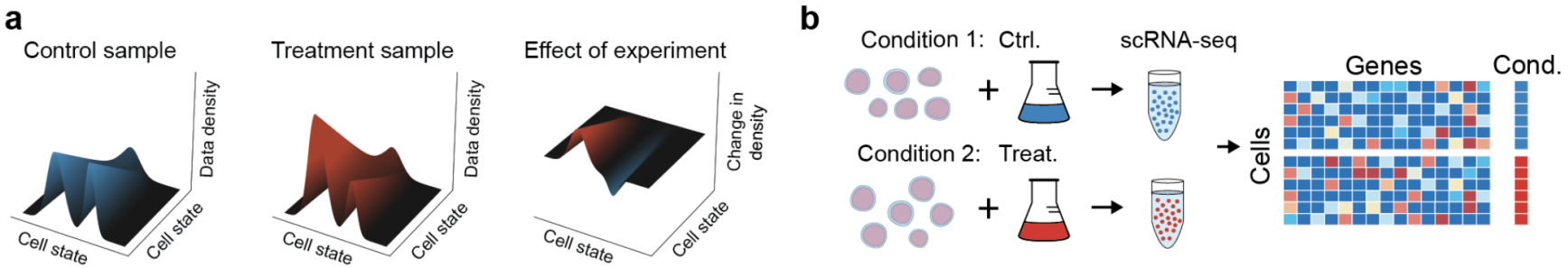


Figure: Burkhardt *et al.*, Nature Biotechnology. 2021. How more or less likely are we to observe a cell in a particular state in a control vs treatment sample?

Treatment as a Signal on a Graph

After creating a graph of cells, an indicator of treatment or control can be viewed as the signal on the graph. Interpretations of 'signal' in relation to graph structure should help to inform treatment associated relative likelihood.

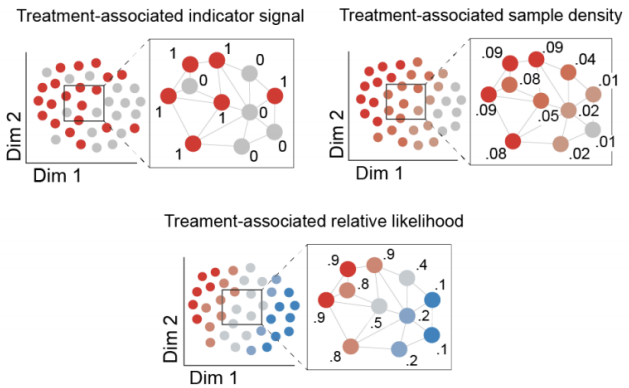


Figure: Burkhardt *et al.*, Nature Biotechnology. 2021.

MELD Overview

Ultimately, the goal is to estimate the density of signals over a graph, where the nodes are cells and edges represent affinity between cells.

Algorithm 1: The MELD algorithm

Input: Dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^m$; Condition labels $\mathbf{y}$ s.t. $\mathbf{y}_i$ indicates the condition in which observation $\mathbf{x}_i$ was sampled.

Output: Sample-associated relative likelihood $\tilde{\mathbf{Y}}_{norm} \in \mathbb{R}^{n \times d}$ where d is the number of unique conditions in $\mathbf{y}$

1. Build graph $\mathbf{G} = \{V, E\}$ by applying anisotropic or other kernel function on $\mathbf{X}$;
2. Instantiate One-Hot Indicator $\mathbf{Y}$, with one column for each unique condition in $\mathbf{y}$;
3. Column-wise L1-normalize $\mathbf{Y}$ to yield $\mathbf{Y}_{norm}$;
4. Apply manifold heat filter over $(\mathbf{G}, \mathbf{Y}_{norm})$ to calculate $\tilde{\mathbf{Y}}$, the kernel density estimate of the data in each condition, also referred to as the **sample-associated density estimates**;
5. Row-wise L1 normalize $\tilde{\mathbf{Y}}$ to yield $\tilde{\mathbf{Y}}_{norm}$ also referred to as the **sample-associated relative likelihoods**;

Figure: Burkhardt *et al.*, Nature Biotechnology. 2021.