

Comp790-166: Computational Biology

Lecture 8

February 6, 2023

Today

- Data Augmentation for Single-Cell Data : Filling in the gaps across the cellular landscape.
- Start graph signal processing (GSP) - an important tool for the upcoming attraction that is differential abundance analysis.

Do you Remember Questions

- What is the intuition for how MAGIC fixes noise and / or dropout? If given a cell, how would I adjust its features, given the rest of the data?
- What do the potential distances in PHATE do?

Rare + Sparse Cell Populations

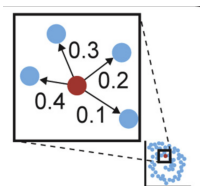


Figure: from MAGIC paper

- 'Dense' cell-populations vs 'Sparse' cell-populations.
- Low-frequency cell-populations and data artifacts can fail to be under-represented and not accurately reflect the underlying biology.
- Are members of a particular cell-type not there or are they just very infrequent?

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.

Welcome SUGAR

Generate points uniformly from intrinsic data geometry / manifold:

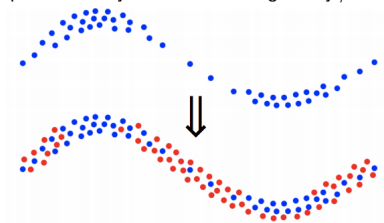


Figure: from Lindenbaum *et al.* NeurIPS. 2018

- 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)

Common Data Generation Approaches

- Older Techniques
 - **Parametric Models** : Specify a model and optimize the parameters through maximum likelihood optimization.
 - Use the learned model to generate new data
 - Use a histogram or kernel to estimate generating distributions
- Newer techniques to generate additional points from complicated data distributions
 - GAN (generative adversarial network) →
<https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf>
 - VAE (variational autoencoders) →
<https://arxiv.org/abs/1606.05908>

Remembering our Good Friend Gaussian Kernel

For a pair of nodes, i and j , the strength of their connection can be computed as $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$, or

$$K_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \quad (1)$$

We also remember our row-stochastic markov matrix, \mathbf{P} computed from \mathbf{K} .

Measure Based Gaussian Correlation (MGC)

- Given your data, \mathbf{X} , define a set of reference points, \mathbf{r} with $r \in \mathbf{X}$.
- Define $\mu(\mathbf{r})$ as some measure over the reference points

Then define a new kernel, $\hat{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j)$ as,

$$\hat{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\mathbf{r} \in \mathbf{X}} \mathcal{K}(\mathbf{x}_i, \mathbf{r}) \mathcal{K}(\mathbf{x}_j, \mathbf{r}) \mu(\mathbf{r}) \quad (2)$$

For our purposes, $\mu(\mathbf{r})$ will be some value that relates to sparsity and is the inverse of node r 's degree.

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 $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \in \mathbf{X}$
- Assuming that instances, \mathbf{X} are unevenly sampled from \mathcal{M} , we seek a set of new points, $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$ 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 $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}, \mathbf{x}_i \in \mathbb{R}^D$.

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

- 1: Compute the diffusion geometry operators \mathbf{K} , \mathbf{P} , and degrees $\hat{d}(i), i = 1, \dots, N$ (see Sec. 3)
- 2: Define a sparsity measure $\hat{s}(i), i = 1, \dots, N$ (Eq. 2).
- 3: Estimate a local covariance $\Sigma_i, i = 1, \dots, N$, using k nearest neighbors around each \mathbf{x}_i .
- 4: For each point $i = 1, \dots, N$ draw $\hat{\ell}(i)$ vectors (see Sec. 4.3) from a Gaussian distribution $\mathcal{N}(\mathbf{x}_i, \Sigma_i)$. Let $\hat{\mathbf{Y}}_0$ 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 $j = 1 \dots \ell_i$ 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 $\mathbf{Y}_0 = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$
- **MGC Kernel:** Now use affinities between points in \mathbf{X} and \mathbf{Y}_0 . Here, points in \mathbf{X} are used as reference.

$$\hat{\mathcal{K}}(\mathbf{y}_i, \mathbf{y}_j) = \sum_r \mathcal{K}(\mathbf{y}_i, \mathbf{x}_r) \mathcal{K}(\mathbf{x}_r, \mathbf{y}_j) s(r) \quad (3)$$

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 $\hat{\mathbf{P}}$ (see Sec 4.2).
- 6: Apply the operator $\hat{\mathbf{P}}$ at time instant t to the new generated points in $\hat{\mathbf{Y}}_0$ to get diffused points as rows of $\mathbf{Y}_t = \hat{\mathbf{P}}^t \cdot \mathbf{Y}_0$.
- 7: Rescale \mathbf{Y}_t to get the output $\mathbf{Y}[\cdot, j] = \mathbf{Y}_t[\cdot, j] \cdot \frac{\text{percentile}(\mathbf{X}[\cdot, j], 99)}{\max \mathbf{Y}_t[\cdot, j]}$, $j = 1, \dots, D$, 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

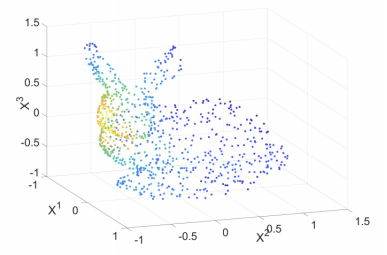
$$\mathbf{Y}_t = \mathbf{P}^t \times \mathbf{Y}_o \quad (4)$$

Same Story Regarding t

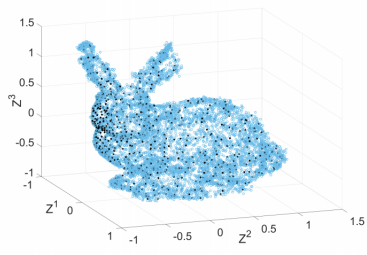
Remember in PHATE, t was chosen according to the knee point of the Von-Neumann entropy of the normalized eigenvalues of \mathbf{P}^t .

Experiments : Synthetic and Biological

Stanford Bunny



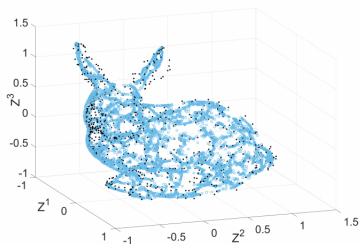
(a)



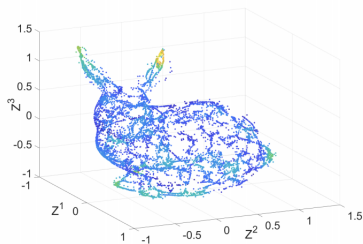
(b)

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



(c)



(d)

Figure: from Lindenbaum *et al.* NeurIPS. 2018. (c). Original and generated points, before MGC diffusion. (d) Generated points (\mathbf{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.

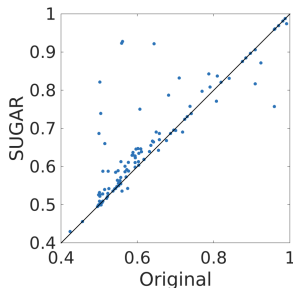


Figure: from Lindenbaum *et al.* NeurIPS. 2018. In 119 datasets, the data were augmented with additional datapoints using sugar. Adjusted Rand Index was computed for the original data vs data + SUGAR.

SUGAR on Single-Cell

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

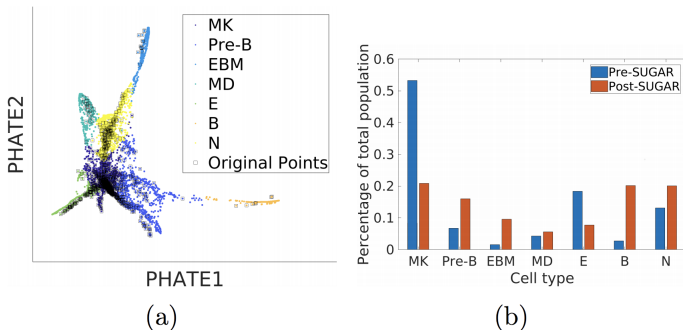


Figure: from Lindenbaum *et al.* NeurIPS. 2018

Maintaining Intra-Module Marker Co-Expression

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

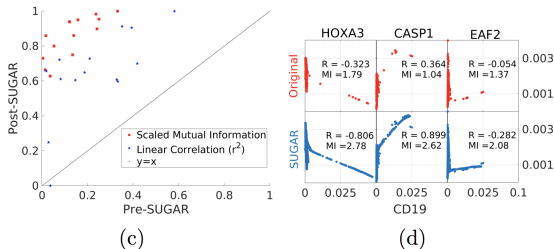


Figure: from Lindenbaum *et al.* NeurIPS. 2018

Graph Signal Processing

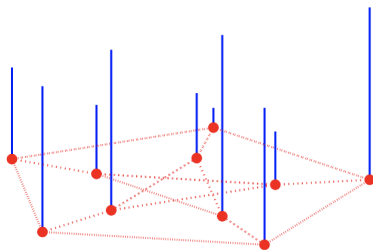


Figure: from Shuman *et al.* ArXiv. The purpose is to study the interplay between some signal and graph connectivity.

Piecewise Smooth Assumption

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.

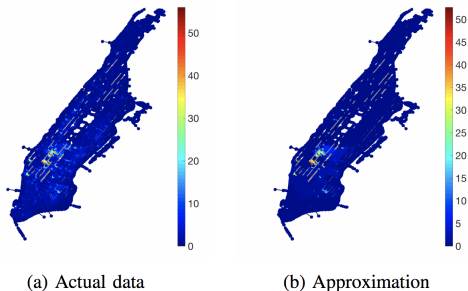


Figure: from <https://arxiv.org/abs/1712.00468>. Here the graph is street intersections in Manhattan and the signal is taxi pickups.

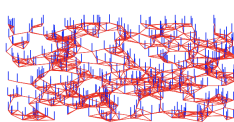
How Localized is the Signal?

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

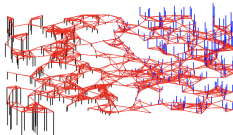
- Some very nice theory falls out about based on the spectra of the Laplacian matrix, relating to how 'localized' a graph signal, \mathbf{f} , is. For example \mathbf{f} could be an expression of some protein.
 - First re-write \mathbf{f} in terms of eigenvectors of the Laplacian
 - The eigenvectors corresponding to the first *smallest* 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 larger eigenvalues, the values of the eigenvectors of adjacent nodes will be more different.

Signal Specificity

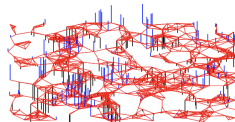
Here we visualize eigenvector entries at nodes ($\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_{50}$)



\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.

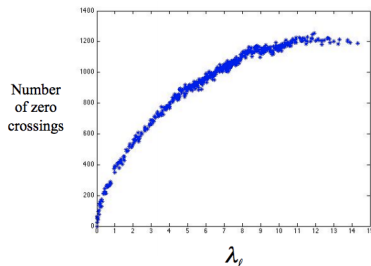


Figure: from GSP Review <https://arxiv.org/abs/1211.0053>

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,

$$(\mathcal{L}\mathbf{f})(i) = ([\mathbf{D} - \mathbf{A}]\mathbf{f})(i) \quad (5)$$

$$= d(i)\mathbf{f}(i) - \sum_j A_{ij}\mathbf{f}(j) \quad (6)$$

$$= \sum_j A_{ij}(\mathbf{f}(i) - \mathbf{f}(j)) \quad (7)$$

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

$$TV(\mathbf{f}) = \sum_{i,j} A_{ij}(\mathbf{f}(i) - \mathbf{f}(j))^2 \quad (8)$$

$$= \mathbf{f}^T \mathcal{L} \mathbf{f} \quad (9)$$

- 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

- Start with the eigendecomposition of \mathbf{L} as $\mathbf{L} = \mathbf{\Psi} \mathbf{\Lambda} \mathbf{\Psi}^T$
- We can look at eigenvectors, $\mathbf{\Psi} = [\psi_1, \psi_2, \dots, \psi_N]$ of \mathcal{L}
- and eigenvalues, $\mathbf{\Lambda} = [0 = \lambda_1 \leq \dots \leq \lambda_N]$ of \mathcal{L}

The Graph Fourier Transform of a Signal

The i th frequency component of a signal, \mathbf{f} is the inner product between ψ_i and \mathbf{f} and can be written as,

$$\hat{f}_i = \psi_i^T \mathbf{f} \quad (10)$$

The Graph Fourier Transform (GFT) is written as,

$$\hat{\mathbf{f}} = \mathbf{\Psi}^T \mathbf{f} \quad (11)$$

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,

$$h : [0, \max(\mathbf{\Lambda})] \rightarrow [0, 1] \quad (12)$$

Assuming the cutoff is λ_k ,

$h(x) > 0$, for $x < \lambda_k$ and $h(x) = 0$, otherwise

Defining Notation and Applying Filter to GFT

Define $h(\mathbf{\Lambda})$ as a diagonal matrix of eigenvalues with the filter applied. Based on what we computed with GFT, the filtered signal, $\hat{\mathbf{f}}_{filt}$ can be computed as,

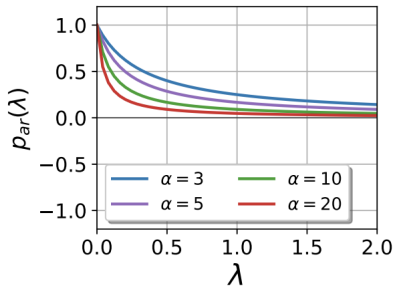
$$\hat{\mathbf{f}}_{filt} = h(\mathbf{\Lambda})\hat{\mathbf{f}} \quad (13)$$

Applying a Filter in General

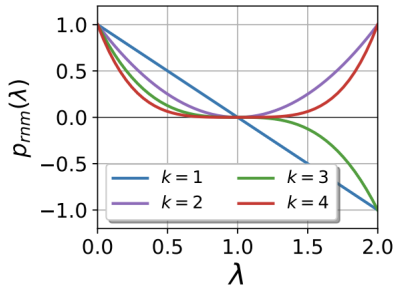
In general, you can filter an original signal, \mathbf{f} in general as,

$$\underbrace{\Psi(\mathbf{I} + \alpha\mathbf{\Lambda})^{-1}\Psi^T}_{\text{Filtered Graph Laplacian}} \mathbf{f}. \quad (14)$$

Example Filters



(a) $p_{ar}(\lambda) = (1 + \alpha\lambda)^{-1}$



(b) $p_{rm}(\lambda) = (1 - \lambda)^k$

Figure: from https://openaccess.thecvf.com/content_CVPR_2019/papers/Li_Label_Efficient_Semi-Supervised_Learning_via_Graph_Filtering_CVPR_2019_paper.pdf