Identifying Localized Signals On Graphs

Sam Leone

May 1, 2022

Introduction

In this problem, we are presented with N cells and their expression of each of M genes (or their expression with dropout). So given a gene g, we would like to come up with a way of answering the question: how localized is g on our set of cells? Two problems are: 1. creating spatial embeddings for the cells based off of their gene expression and 2. using these embeddings, quantify the 'spread' of the expression of g over a gene graph $\mathcal{G}$. Two main approaches include: 1. decomposing signals s into a sparse linear combination of diffusion wavelets; by weighing the coefficients of the large scales more heavily, we quantify a sort of "average scale" of the signal. A second approach uses a spatial embedding of the cells and uses the average distance between two points generated from s viewed as a distribution.

1 — Frequency Approach

One possible way to approach is this problem is to decompose the signal s into a linear combination of diffusion wavelets at various scales. Then the more components the signal has in larger scales, the more spread the signal is.

In particular, we define a family of diffusion wavelets like so: suppose we have a diffusion operator M on our graph, with *columns* giving probability distributions from points. For example, if we have spatial data with affinity matrix A and degree matrix D, $M = D^{-1}A$. We define:

$$P = \frac{1}{2}(I + M) \text{ a lazy random walk operator}$$

As is typical, we will define $\Psi_j = P^{2^{j-1}} - P^{2^j}$. Note that for large values of j, $\Psi_j(\delta_i) \approx \Psi_j)(\delta_k)$ if vertex i is near k. Thus, we can reduce the number of columns in Ψ_j by taking some subset of the columns to form a matrix $\tilde{\Psi}_j$ such that the total error in projecting Ψ_j onto $\tilde{\Psi}_j$ is less than some ϵ fraction of the norm of the whole Ψ_j. That is,

$$\frac{1}{\|\Psi_j\|}\|^2 \|\Psi_j - \tilde{\Psi}_j({\tilde{\Psi}_j}^T \tilde{\Psi}_j)^{-1} \tilde{\Psi_j}^T \Psi_j\|^2 \leq \epsilon \tag{1}$$

Think of $\tilde{\Psi}_j$ as a selection of wavelets at scale j that span the graph. In general, $|\tilde{\Psi}_j|$ decreases in j and increases in ϵ. Here's an algorithm for obtaining such a $\tilde{\Psi}_j$. For example,

Algorithm 1 Obtaining $\tilde{\Psi_j}$

Input: A set of diffusion wavelets Ψ_j and a tolerance $\epsilon \geq 0$
Ouput: $\tilde{\Psi_j}$, whose columns are a subset of Ψ_j such that they nearly span Ψ_j
Let $\tau \leftarrow \epsilon \|\Psi_j\|^2$
$Q, R, p \leftarrow Pivotal - QR(\Psi_j)$
$R_{norm} \leftarrow \|\Psi_j\|^2$
while for i in $1...n$ **do**
  $R_{norm} \leftarrow R_{norm} - \sum_{j=i}^{n} R_{i,j}^2$
  if $R_{norm} \leq \tau$ **then**
    return $\Psi_j(p(1), p(2)...p(i))$
  end if
end while

This algorithm is proven to be correct in the Appendix. If n is very large, we could also input a randomly sampled set of columns into the algorithm. Think of Ψ_j as being some spanning set of wavelets. Of course, for larger and larger scales, we might imagine that we can take relatively few of these, as they can cancel a great deal of low frequency signals. Here is such an example:

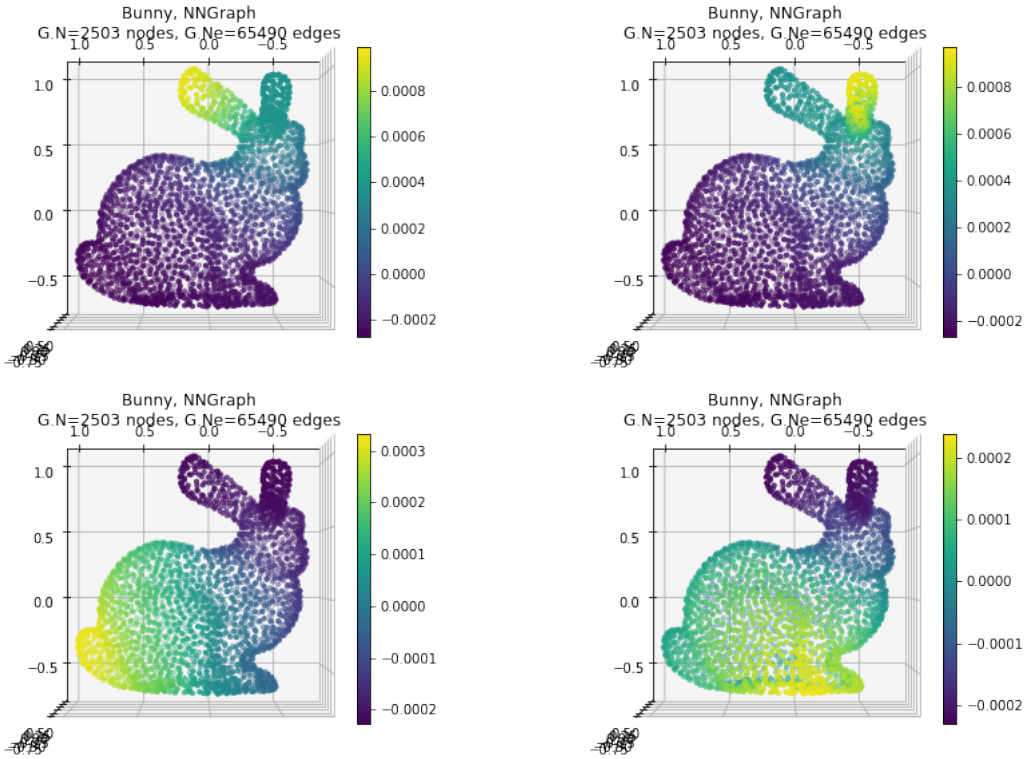


Figure 1: Example of $\tilde{\Psi_j}$. Notice that the original matrix Ψ_j is 2503×2503. In contrast, $\tilde{\Psi_j}$ is 2503×23. Four such "spanning" columns as outputted by Algorithm 1 using tolerance $\epsilon = 10^{-3}$ are shown above.

Detecting Local Signals

With this out of the way, we can present a possible algorithm for detecting "localized" signals. We first create a matrix $\Phi = \begin{bmatrix}\Psi_0 & \Psi_1..\Psi_J\end{bmatrix}$. That is, it is the horizontal concatenation of all of our wavelets. We then define a weight vector $\omega = \begin{bmatrix}1...1 & 2...2 & .... & 2^J...2^J\end{bmatrix}$. For a more efficient implementation, we could also set Φ using the $\tilde{\Phi}_j$'s instead; this would also help avoid overfitting. Regardless, normalize the columns of Φ for consistency. Now, given a signal s (assume it is normalized), we can estimate a sparse linear combination x of columns of Φ that approximate s. For our purposes, let's employ the matching pursuit algorithm. Recall that this algorithm attempts to solve, given some ϵ,

$$\min_x \|x_0\| \text{ such that } \|\Phi x_0 - s\| < \epsilon$$

So suppose we have an algorithm $MP(\Phi, s, \epsilon)$ that outputs the desired x. We can then set a measure of locality ℓ of s like so:

$$\ell(s; x) = \omega \cdot |x| = \sum_i |x_i|\omega_i = \sum_j 2^j \sum_{k:k \text{ corresponds to scale j}} |x_k|$$

Which motivates the following algorithm for finding localized signals:

Algorithm 2 Calculating Localization

Input: A diffusion operator M and signal s defined on a set $\mathcal{X}$. SPARSE, a boolean variable encoding whether to use the full Ψ_j or $\tilde{\Psi}_j$, an ϵ_1 to dictate $\tilde{\Psi}_j$. OMP a routine for matching pursuit. And associated tolerance ϵ_2 for this algorithm.
Output: $\ell(s)$, a measure of localization of s on the set $\mathcal{X}$.

$P \leftarrow \frac{1}{2}(I + M)$
$J \leftarrow \log(|\mathcal{X}|)$ (unless otherwise specified)
$\Psi_0 \leftarrow I$
for $j \in [J]$ **do**
  $\Psi_j \leftarrow P^{2^{j-1}} - P^{2^j}$
  if SPARSE **then**
    Calculate $\tilde{\Psi}_j$ given Ψ_j, ϵ_1 per algorithm 1
  end if
end for
$\omega \leftarrow [1, 1...2, 2...2^J, 2^J]$ with as many 2^j's as columns in Ψ_j or $\tilde{\Psi}_j$
$\Phi \leftarrow \begin{bmatrix}\Psi_0 & \Psi_1.. & \Psi_J\end{bmatrix}$ or $\Phi = \begin{bmatrix}\tilde{\Psi}_0 & ... & \tilde{\Psi}_J\end{bmatrix}$ as appropriate.
Normalize columns of Φ
$x \leftarrow OMP(\Phi, \frac{s}{\hat{s}}, \epsilon_2)$ (we normalize s in case we'd like to compare different signals)
$\ell(s) \leftarrow \langle w, |x| \rangle$, where $|x|$ is the elementwise absolute value of x
return $\ell(s)$.

An alternative algorithm, for the reduced set, is to use $\tilde{\Psi}_j$ to calculae $\tilde{\Psi_{j+1}}$. This would go like so:

Algorithm 3 Calculating Locality (Alternative)

Input: A diffusion operator M and signal s defined on a set $\mathcal{X}$. SPARSE, a boolean variable encoding whether to use the full Ψ_j or $\tilde{\Psi}_j$, an ϵ to dictate $\tilde{\Psi}_j$. OMP a routine for matching pursuit.
Output: $\ell(s)$, a measure of localization of s on the set $\mathcal{X}$.

$P \leftarrow \frac{1}{2}(I + M)$
$J \leftarrow \log(|\mathcal{X}|)$ (unless otherwise specified)
$\Psi_0 \leftarrow I$
for $j \in [J]$ **do**
  $\Psi_j \leftarrow (I - P^{2^{j-1}})\tilde{\Psi_{j-1}}$
  Set $\tilde{\Psi}_j$ per our algorithm.
end for
$\omega \leftarrow [1, 1...2, 2...2^J, 2^J]$ with as many 2^j's as columns in $\tilde{\Psi}_j$
$\Phi \leftarrow \begin{bmatrix}\tilde{\Psi}_0 & ... & \tilde{\Psi}_J\end{bmatrix}$ as appropriate.
Normalize columns of Φ
$x \leftarrow OMP(\Phi, \frac{s}{\tilde{s}})$ (we normalize s in case we'd like to compare different signals)
$\ell(s) \leftarrow \langle w, |x| \rangle$, where $|x|$ is the elementwise absolute value of x
return $\ell(s)$.

A benefit to this version is that Ψ_j has as many rows as $\tilde{\Psi_{j-1}}$, so our work keeps decreasing, which becomes highly beneficial at larger and larger scales. Note it is more efficient to caalculate P via repeated squaring. This way, we calculate $P^{2^j} \leftarrow (P^{2^{j-1}})^2$ in time $\mathcal{O}(n^2)$. So for J scales, runtime is at worst $\mathcal{O}(n^2 \log(n))$. Compare this to an eigendecomposition, which could take time up to $\mathcal{O}(n^3)$. So for large n and relatively few scales, this is more effective.

Example on the Bunny Graph

As a start, let's use the redundant ϕ, which contains all wavelets at all scales. As a sanity check, we can consider graph signals of the form $s = M^t\delta_i$ and verify that, as we increase t, ℓ increases.

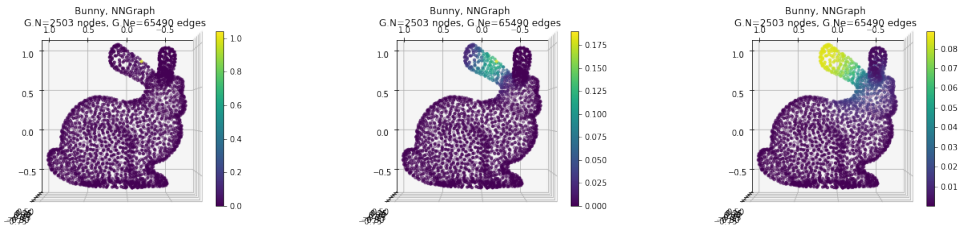


Figure 2: Here we have visualized signals of the form $M^t\delta_i$ with $t = 0, 2^3, 2^6$ and $i = 1400$. We see that as we increase t, the signal should intuitively be less "local." Accordingly, the respective measures of locality are approximately $50, 157$, and 242

We could also repeat this using our QR-reduced dictionaries. In this case, the signals are the same, but the estimates of locality are $24, 357$, and 492. So wee see that in the simplest case,

our algorithm does what it's supposed to.

Moving Signals Across the Sphere

Another thing we might imagine doing is fixing one signal on a sphere and moving another to see how the relative localization changes. This is visualized in the below diagram:

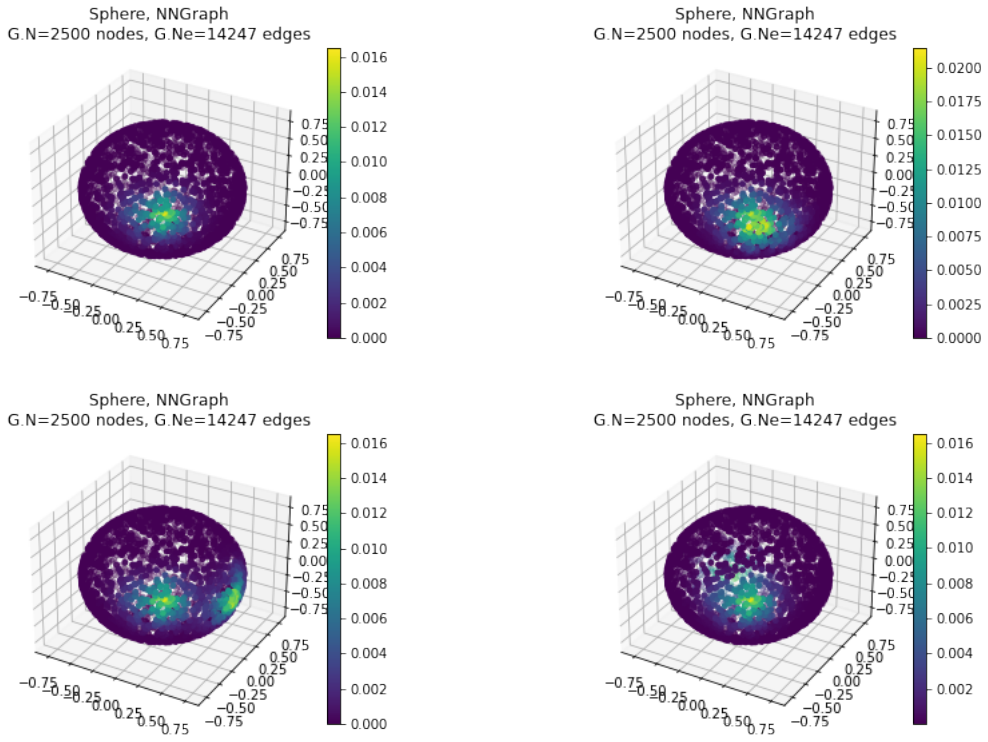


Figure 3: We can see that as we continue this process, we would hope that the signal is identified as less and less localized. But once the signal becomes more and more bimodal, the level of localization stabilizes. What we see is that the corresponding levels of localizations $684, 728, 741$, and 421. Here we use the sparser wavelets with $\epsilon = 0.001$

This example demonstrates that, at the bare minimum, the most centered signal indeed receives the lowest score. Then, the large jump in value from the first to the second signals that "wider" signals are weighed more heavily than bimodal ones. Whether this is desirable is questionable; maybe it is, maybe it isn't. If it is not, it may be worth revisiting the weight vector ω.

Non-Frequency Approach

Another possible approach is to avoid scanning the signal using frequencies altogether and use a quantity which captures the "spread" of a signal. Suppose we have our set $\mathcal{X}$ and, rather than a diffusion operator, some distance $d(\cdot,\cdot)$ defined on $\mathcal{X}\times\mathcal{X}$. In this case, if we have a nonnegative signal $s\in\mathbb{R}_+^n$, we can normalize s by taking $\hat{s}=\frac{s}{\sum_{x\in X}s(x)}$, and view $\hat{s}$ as a probability distribution. Now we can examine the quantity:

$$\mathcal{S}(s)=\sum_{x,y}\hat{s}(x)\hat{s}(y)d(x,y)$$

Notice that in the simplest case, if s is binary on $\mathcal{X}$ (ie $s\in\{0,1\}^{|\mathcal{X}|}$), we couls say $s=1$ over a set $A\subset X$ and observe

$$\mathcal{S}(s)=\frac{1}{|A|^2}\sum_{x,y\in A^2}d(x,y)$$

Which can be viewed as average distance between vertices where s is "on." So then $\mathcal{S}(s)$ for general s serves as a continuous extension of this. Similarly, if we think of $\hat{s}$ as a distribution, then, the joint distribution of (x,y) is simply $\hat{s}(x)\hat{s}(y)$. So then we have the interpretation,

$$\mathcal{S}(s)=\sum_{x,y\in\mathcal{X}\times\mathcal{X}}\mathbb{P}\{x,y\}d(x,y)=\mathbb{E}_{x,y\sim\hat{s}}[d(x,y)]$$

Note that if $\mathcal{X}$ is finite, then we can store $d(\cdot,\cdot)$ in a matrix D. So then,

$$\mathcal{S}(s)=\frac{s^TDs}{s^T\mathbf{1}\mathbf{1}^Ts}$$

A possible normalization factor, for the sake of comparison, could be

$$C(s):=(\sum_x s(x))^2-\sum_x s(x)^2=\hat{s}^T(J-I)\hat{s}=\mathbb{E}_{x,y\sim\hat{s}}(1-\delta(x,y))$$

Which is the expected distance over the complete graph with weight $\omega:E\to\mathbb{R}$ with $\omega(x,y)=\delta(x,y)$. For diracs s, $C=0$, this sends the normalized measure to ∞, which could be a nice way to weed out trivially localized signals.

So in practice, we could choose $d(\cdot,\cdot)$ in any number of ways. If $\mathcal{X}\in\mathbb{R}^n$, we could simply use the Euclidean distance between data points and store them in a matrix D. If X is not spatial, d could be the length of the shortest path. We could also let $d(x,y)$ be, for example, the diffusion distance between x and y, given by $\|P(x,\cdot)-P(y,\cdot)\|^2_{L^2(X,dP/d\pi)}$ where $P:\mathcal{X}\times\mathcal{X}$ is a probability kernel with stable distribution π. As shown by Coifman & Lafon, this would be equivalent to $\|\Psi(x,\cdot)-\Psi(y,\cdot)\|^2$ where Ψ denotes the matrix whose columns are right eigenvectors of P. Another possible choice could be the distance used by PHATE, given by $\sqrt{\|\log P^t(x,\cdot)-\log P^t(y,\cdot)\|}$ (t is by default the Von Neumann entropy).

Thus, we present the following algorithm:

Algorithm 4 Algorithm For Computing Locality 2

Input: a graph G with associated affinity or adjacency matrix A. Set of signals $S = \{s_1...s_m\}$ on the graph

Output: A set of localities $\{l_1...l_m\}$ for each signal.

Calculate the diffusion operator $P = AD^{-1}$, where D is the matrix of row sums of A.

Calculate the Von Neumann entropy t of P.

Calculate $\log(P^t)$. Optionally, run PCA on this matrix to expedite the next step.

Form a matrix D such that $D_{i,j} = \sqrt{\| \log P^t(i,\cdot) - P^t(j,\cdot)\|}$

for $i \in 1..m$ **do**

$s_i \leftarrow \frac{1}{\sum_x s(x)} s^t D s$

$c_i \leftarrow \frac{1}{\sum_x s(x)} s^t (J - I) s$

$l_i = \frac{s_i}{c_i}$

end for

return $l_1...l_m$.

Remark: For large graphs G, D is highly expensive to compute, requiring $\mathcal{O}(n^3)$ arithemetic operations. Instead, you could generate the matrix $\log(P)$ in time $\mathcal{O}(n^2)$, reduce to dimensionality d via PCA (like $d = 100$) and then calculate D in time $\mathcal{O}(n^2 d)$.

Results

Bunny Graph: On the same examples for the Bunny (Figure 2) as before, the corresponding measures of locality are 30, 89, and 160.

Moving Signals on the Sphere: A similar example to one on the sphere yielded 196, 197, 292, and 443 as we spread the signal to the other side. Thus, whereas the frequency based algorithm just cared about decomposing signals into sparse wavelets, this new meaasure *does* penalize multimodal distributions.

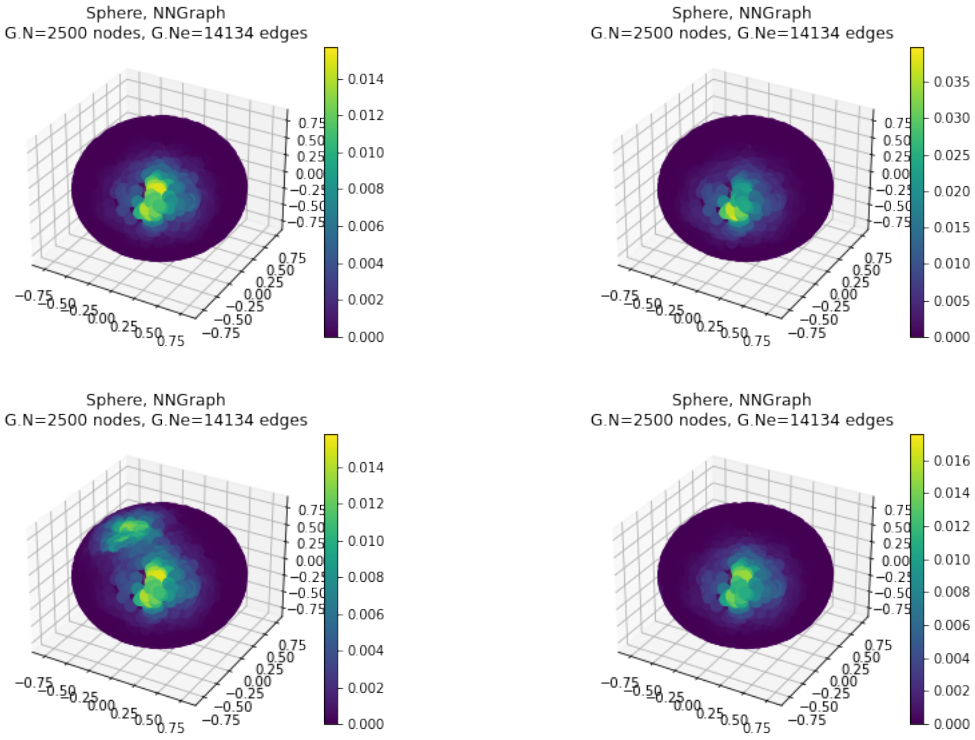


Figure 4: Note that in the final figure, there is a second mode on the other side of the sphere which is not visible. Again, the corresponding locality values were 197, 196, 292, and 443, which is intuitively what we might hope for! While signal 1 is given by heat diffusion centered at just 1 point and signal 2 is the sum of two diffusions (so the intent was admittedly for 1 to be more local), just from visual inspection, signal 2 is the more localized one.

On the Single Cell Data Now, let's see what the resulting output is on cell-cell data. Here, we look at a signal given by expression of 1991 genes. We also take an order 32 diffusion of the data to counteract dropout. Furthermore, we rank the *normalized* localities to obtain a list of the least to most "local" genes. For reference, here is the distribution of the data in general:

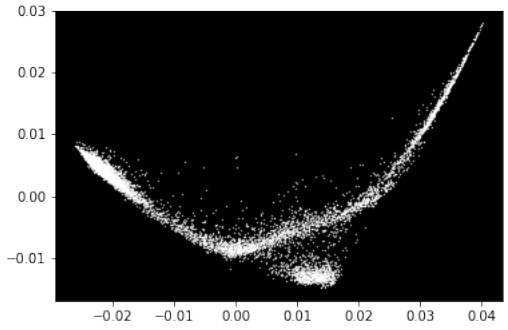


Figure 5: Here each point represents a data cell. Coordinates are given by a PHATE embedding using default settings. The diffusion operator is generated from Euclidean distance in a PCA-reduced feature space.

And here are genes 0, 500, 1000, and 1500 (numbered in increasing order of l):

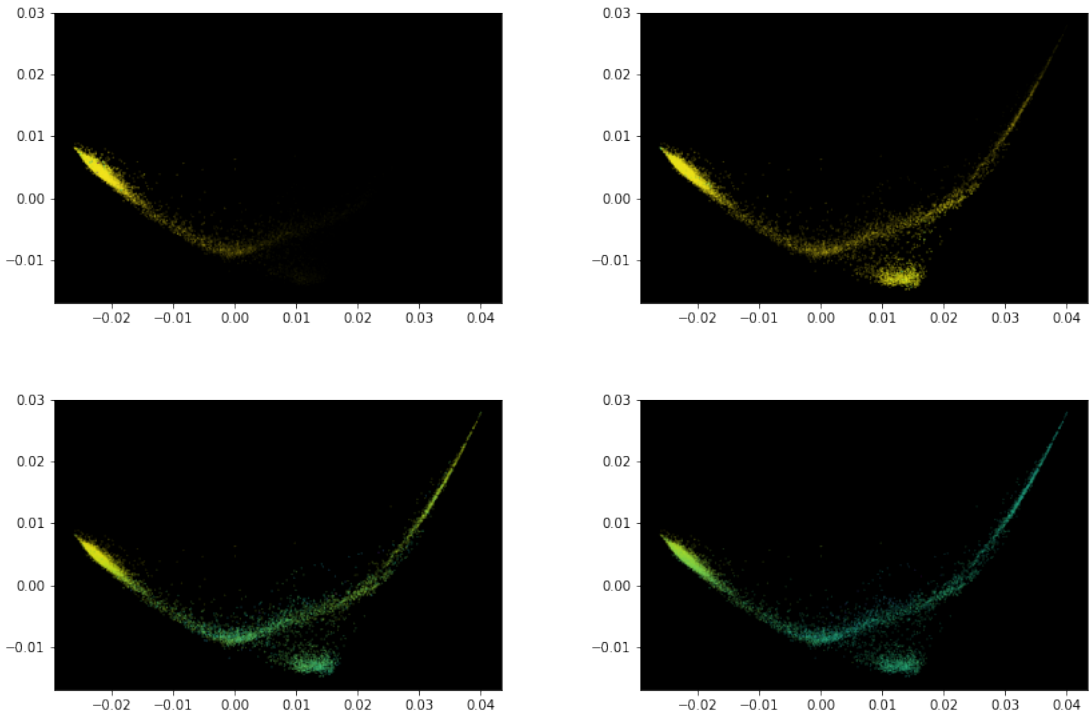


Figure 6: Here we have drawn plots where we use color and size to indicate gene intensity. The visualization is a 2-dimensional phate embedding (default settings). The corresponding locality measures are 110, 250, 285, and 306. Fortunately, these signals to intuitively seem to be more and more spread out.

Robustness to Dropout?: Another natural question is: how necessary was it to diffuse the signal to combat dropout? Certainly, if we're working with frequencies, this is necessary. But here our appraoch isn't so sensitive. Let's go ahead and re-rank the signals, but using their *raw* values over the genes. The result will be two vectors ℓ_r and ℓ_d, where $\ell_r(i)$ contains the localization of the raw signal of gene i and $\ell_d(i)$ is the diffused localization. In our case,

$$\ell_r = (306.0, 286.3, 179.9, 207.3...179.2, 287.5, 295.0)$$

$$\ell_d = (302.3, 283.4, 208.8, ...187.2, 287.4, 234.9)$$

By initial inspection, these values roughly aline, but we can also go ahead and compare how they compare on a *relative* basis. That is, let r_r be the indices of ℓ_r in increasing order and r_d likewise. We can then look at the Kendall-Tau distance between ℓ_r and ℓ_d. Unfortunately, here we have $\tau = 0.00672$ and a p-value of nearly 0.65. Even after comparing this to a more moderate scale of diffusion, we get weak results. This doesn't mean they're not correlated, just that perhaps that sensitivity to small perturbations affected the ranking too much to observe a

significant value here.

While it would have been nice to be able to say the rankings are nearly equivalent, we can also look at a correlation test. Here, we have 90% correlation with a p-value of 0 (up to machine precision). So there is consistency regardless of if we account for dropout.

Tikhonov-Regularized Witness Functions

Another possible idea is to extend notions of Earth Mover's Distance to the graph domain. To do this, let us recall our definition of Earth Mover's Distance:

Definition: *Let P, Q be probability measures on a metric space. Let $\Pi(P, Q)$ denote the set of couplings of the distributions P and Q. That is, for each $\gamma \in \Pi$, γ is a joint distribution whose marginals are P and Q, respectively. Then the Earth Mover's Distance between P and Q is defined to be:*

$$\text{EMD}(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}\gamma d(x, y)$$

Kantorovich-Rubinstein Duality

One of the most famous theorems related to optimal transport is the Kantorovich Rubinstein duality. **Theorem (Kantorovich-Rubinstein Duality:)**

$$\text{EMD}(P, Q) = \max_{\|f\|_L \leq 1} \mathbb{E}_P(f) - \mathbb{E}_Q(f)$$

While the Kantorovich-Rubinstein Duality is well known in L_2 Lebesgue space with the usual Euclidean distance, we can go ahead and try to prove it for a more discrete setting. In this case, the setup is as follows. We are given an inner product space $(\mathcal{X}, \langle \cdot, \cdot \rangle)$ which induces a distance $d(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Also, say we are given a finite sample V of $\mathcal{X}$ and two probability distributions over V. Finally, let S denote the space of functions f on V such that $|f(u) - f(v)| \leq d(u, v)$. This is analagous to a 1-Lipschitz condition on our abstract distance. I claim that:

$$\text{EMD}(P, Q) = \max_{f \in S} \mathbb{E}_P(f) - \mathbb{E}_Q(f)$$

Proof:

We can appeal to duality to write the optimization. Recall that, by definition,

$$\text{EMD}(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}\gamma d(x, y)$$

Furthermore, suppose without loss of generality that V is the numbers $1...n$. We could turn γ into a $n^2 \times 1$ vector γ' such that $\gamma(x, y) = \gamma'(x(y-1)+y)$. That is, γ' is the concatenation of $\gamma(1, \cdot), \gamma(2, \cdot)$.... Furthermore, $\sum_y \gamma(x, y) = P(x), \sum_y \gamma(x, y) = P(x)$ are also linear constraints on γ'. In particular, we can represent this as the condition $A\gamma' \geq (P, Q), (-A)\gamma' \geq (-P, -Q)$ for some matrix A. If $n = 2$, this might look like:

$$\gamma' = (\gamma(1,1), \gamma(1,2), \gamma(2,1), \gamma(2,2)) \qquad A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

So that,

$$A\gamma' = \begin{bmatrix} I(X) \\ I(Y) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma(1,1) \\ \gamma(1,2) \\ \gamma(2,1) \\ \gamma(2,2) \end{bmatrix} = \begin{bmatrix} \gamma(1,1) + \gamma(1,2) \\ \gamma(2,1) + \gamma(2,2) \\ \gamma(1,1) + \gamma(2,1) \\ \gamma(1,2) + \gamma(2,2) \end{bmatrix} = \begin{bmatrix} P(1) \\ P(2) \\ Q(1) \\ Q(2) \end{bmatrix} = \begin{bmatrix} P \\ Q \end{bmatrix}$$

That is, we can break down into A into indicators for fixed x over each y (given by $I(x)$ and then fixed y over each x ($I(y)$). Thus, we can enforce $A\gamma'$ obey the marginals by forcing:

$$\begin{bmatrix} A \\ -A \end{bmatrix} \gamma' \geq \begin{bmatrix} P \\ Q \\ -P \\ -Q \end{bmatrix}$$

We can do the same concatenation to d to get d', too. Finally, we can express γ as the solution to the following linear program:

$$\text{EMD}(P, Q) = \min_{\gamma' : (A, -A)\gamma \geq (P, Q, -P, -Q), \gamma' \geq 0} \gamma'^T d'$$

Now, we can appeal to the dual linear program. Therefore, this is also equal to:

$$\max_{x : (A, -A)^T x \geq d'} (P, Q, -P, -Q)^T x$$

x will be a vector of length $4n$, say it breaks down in to $x = (x_1, x_2, x_3, x_4)$. Note that $(P, Q, -P, -Q)^T x = P^T x_1 + Q^T x_2 - P^T x_3 - Q^T x_4 = \mathbb{E}_P(x_1 - x_3) + \mathbb{E}_Q(x_2 - x_4)$. For simplicity, let $f = x_1 - x_3, g = x_2 - x_4$. We can also show that $(A, -A)^T x \geq d'$ enforces a Lipschitz type condition. First, notice this can be written as:

$$\begin{bmatrix} A^T & -A^T \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \geq d' \qquad A^T \begin{bmatrix} x_1 - x_3 \\ x_2 - x_4 \end{bmatrix} \geq d'$$

Recalling our definition of f and g as well as our decomposition of A:

$$A^T \begin{bmatrix} f \\ g \end{bmatrix} \geq d' \qquad \begin{bmatrix} I(X)^T & I(Y)^T \end{bmatrix} \begin{bmatrix} f \\ g \end{bmatrix} \geq d'$$

Or finally:

$$I(X)^T f + I(Y)^T g \geq d'$$

What does it mean, for some i, that $(I(X)^T f)(i) + (I(Y)^T g)(i) \geq d'(i)$? Notice, that in the $n = 2$ case, we'd have:

$$I(X)^T = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad I(Y)^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Thus, $(I(X)^T f)(i) + (I(Y)^T g)(i)$ corresponds to $f(x) + g(y)$ for some x and some y. So then our constraint means that for all x, y, $f(x) + g(y) \geq d(x, y)$. This is equivalent to a Lipschitz - type condition on f and g. Thus,

$$\text{EMD}(P, Q) = \max_{f,g: f(x)+g(y) \geq d(x,y)} \mathbb{E}_P(f) + \mathbb{E}_Q(y)$$

Which is looking more similar to the result of Kantorovich-Rubinstein, but not quite. To finish it off, we can show that we could have $f = -g = h$, where h is some "Lipschitz" function. By Lipschitz, I mean $|h(x) - h(y)| \leq d(x, y)$ for all x, y. This would imply that $f(x) + g(y) \geq d(x, y)$, in turn. Thus, this is a subclass of candidate f, g's, so:

$$\text{EMD}(P, Q) \geq \max_{h: |h(x)-h(y)| \leq d(x,y)} \mathbb{E}_P(h) - \mathbb{E}_Q(h)$$

It remains to be shown that we can construct an h which achieves the EMD. In particular, say the maximum is achieved by f and g. It then follows that for all x, y, $f(x) + g(y) \leq d(x, y)$, or $f(x) \leq d(x, y) - g(y)$. In particular, we can let $h(x) = \min_y(d(x, y) - g(y))$, from which we obtain $f(x) \leq h(x)$. But we also have that $h(x) = \min_y(d(x, y) - g(y)) \leq d(x, x) - g(x) = -g(x)$. Interestingly, this is the only point so far where we need d is a distance. Thus, $f(x) \leq h(x) \leq -g(x)$, so $f(x) + g(x) \leq h(x)$, meaning both $\mathbb{E}_P(f) \leq \mathbb{E}_Q(g)$ and $\mathbb{E}_P(g) \leq -\mathbb{E}_P(h)$. And thus,

$$\mathbb{E}_P(f) + \mathbb{E}_Q(g) \leq \mathbb{E}_P(h) - \mathbb{E}_Q(h)$$

It remains to be checked that h is "Lipschitz." For any x_1, x_2, we have:

$$\begin{aligned} h(x_1) - h(x_2) &= \min_y d(x_1, y) + g(y) - \min_y d(x_2, y) + g(y) \\ &\leq \min_y \left[d(x_1, y) - d(x_2, y) + g(y) - g(y) \right] \end{aligned}$$

Because d is a distance:

$$\leq \min_y \left[d(x_1, x_2 \right]$$

So $h(x_1) - h(x_2) \leq d(x_1, x_2)$. Reversing x_1, x_2, we have $h(x_2) - h(x_1) \leq d(x_1, x_2)$, which implies the desired condition. Thus, we conclude that the Kantorovich - Rubinstein equality is true for abstract distances, at least over finite sets. $\square$

Graph Setting and Relaxations

Now, assume that we are given a weighted, undirected graph $G = (V, E, \omega)$ with n vertices. Also, assume that there is a metric space on $\mathcal{X} \supseteq V$ with distance measure d, such that $\omega = d^{-1/2}$. Then $(f(x) - f(y))^2 \leq d(x, y)^2$ becomes $\omega(x, y)(f(x) - f(y))^2 \leq 1$. We could of course, then seek the solution to:

$$\max_{f:\forall(x,y),\omega(x,y)(f(x)-f(y))^2\leq 1} \mathbb{E}_P(f) - \mathbb{E}_Q(f)$$

But the search space for such a function is enormous and complicated. Let us instead impose a slight weakening of $\omega(x,y)(f(x)-f(y))^2 \leq 1$ for all x, y. Rather than having this hold for all x, y, we can just have it be true on average: $\frac{1}{T}\sum_{x,y}\omega(x,y)(f(x)-f(y))^2 \leq 1$ where $T = \sum_{x,y}\omega(x,y)$. Perhaps it's clear where this is going. Of course, we can let A be adjacency matrix, such that $A(x,y) = \omega(x,y)$. We can of course then define a diagonal degree matrix D with $D(x,x) = \sum_y A(x,y)$. And of course, the combinatorial Laplacian as $L = D - A$. Then, we define the resulting optimization:

$$D(P,Q) = \max_f \mathbb{E}_P(f) - \mathbb{E}_Q(f) \text{ subject to } f^T L f \leq T$$

We can actually solve this relatively easily! Note that P and Q have associated vectors in $\mathbb{R}^n$ on V given by p and q , respectively. Likewise, f takes values in $\mathbb{R}^n$. So we can then obtain:

$$\max_f \langle f, p\rangle - \langle f, q\rangle = \max_f \langle f, p-q\rangle \text{ subject to } f^T L f \leq T$$

From here, there are two cases:

- $(p-q)$ is in null space of L. Then $p-q$ is constant, the null space of L corresponding to a fully connected graph is the span of $\mathbf{1}$. But since p and q are both strictly positive and sum to 1, $0 = 1-1 = \sum_i p(i) - \sum_i q(i) = \sum_i (p-q)(i)$. If $\sum_i (p-q)(i) = 0$ and $p-q$ is constant, it follows that $p-q=0$, so $p=q$. Intuitively, this tells us the EMD should be 0. This is suitable with the above formulation too, because no matter the f, $\langle p-q, f\rangle = 0$.
- $(p-q)$ is not in the null space of L. Then, because the constraints are convex and we have a linear objective, it follows that the optimal solution f will fall on the $n-1$ dimensional ellipsoid given by $f^T L f = T$.

Assume case (ii), since case (i) can be solved trivially. We can set up an objective function featuring Legrange multipliers:

$$\mathcal{L}(f,\lambda) = \langle f, p-q\rangle + \lambda(f^T L f - T)$$

Taking the natural set of derivatives, we find that we seek to solve:

$$\frac{\partial \mathcal{L}}{\partial f} = p - q + 2\lambda L f = 0 \tag{2}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = f^T L f - T = 0 \tag{3}$$

Equation (ii) tells us that $2\lambda L f = (q-p)$, or that $Lf = \frac{1}{2\lambda}(q-p)$. In particular, since $q-p$ is in the column space of L, we know f is therefore given by $\frac{1}{2\lambda}L^-(q-p)$, where L^- is the Moore-Penrose Generalized Inverse of L. On the other hand, we require f satisfy equation (3), so:

$$\frac{1}{4\lambda^2}(p-q)^T(L^-)^T LL^-(p-q) = T$$

Since L is symmetric, so is L^-. Thus, the above becomes:

$$\frac{1}{4\lambda^2}(p-q)^T L^-(p-q) = T$$

Or alternatively:

$$\lambda = \frac{1}{2}\sqrt{\frac{1}{T}(p-q)^T L^-(p-q)}$$

Plugging this in for λ:

$$f = \frac{\sqrt{T}}{\sqrt{(p-q)^T L^-(p-q)}} L^-(p-q)$$

Thus, we have found our optimal f, call it $f^\star$. We may as well substitute this into the expression we seek to maximize:

$$\begin{aligned}\langle p-q, f^\star\rangle &= \frac{\sqrt{T}}{\sqrt{(p-q)^T L^-(p-q)}}(p-q)^T L^-(p-q) \\ &= \sqrt{T(p-q)^T L^-(p-q)} = \sqrt{T}\|L^{-\frac{1}{2}}(p-q)\|\end{aligned}$$

Which can be finally written as:

$$D(P,Q) = \|(\frac{1}{T}L)^{-\frac{1}{2}}(p-q)\|$$

We could also compute an eigendecomposition of L into $\Psi\Lambda\Psi^T$. Recall that $T = \sum_{x,y}\omega(x,y) = \sum_x D(x,x) = tr(D)$. But $tr(A) = 0$, so $tr(L) = tr(\Psi\Lambda\Psi^T) = tr(\Lambda\Psi^T\Psi) = tr(\Lambda) = \sum_i \lambda_i$. Thus, yet a third representation of D would be:

$$\begin{aligned}D(P,Q) &= \|(\frac{1}{\lambda_1+\lambda_2...+\lambda_n}\Lambda)^{-\frac{1}{2}}\Psi^T(p-q)\| \\ &= \sqrt{\sum_i \lambda_i}\cdot\sqrt{\sum_{i>0}\lambda_i^{-1}\langle\psi_i, p-q\rangle^2} \\ &= \sqrt{\sum_i \lambda_i}\sqrt{\sum_{i>0}\lambda_i^{-1}\widehat{(p-q)}(i)^2}\end{aligned}$$

And by linearity of the Fourier transform,

$$D(P,Q) = \sqrt{\sum_i \lambda_i}\cdot\sqrt{\sum_{i>0}\lambda_i^{-1}(\hat{p}(i)-\hat{q}(i))^2}$$

More broadly, we could have imposed $f^T Lf \leq \mu^2$ for some μ, in which case the solution would be (if you track the same steps):

$$D_\mu(P,Q) = \mu\sqrt{\sum_{i>0} \lambda_i^{-1}(\hat{p}(i) - \hat{q}(i))^2}$$

Thus, the exact value of μ doesn't matter, since the distances are monotonic.

Claim: $D(\cdot,\cdot)$ *is a distance on functions.*

Proof:

First observe that if $D(\cdot,\cdot)$ is a distance for any μ, it is a distance for all μ. So let $\mu = 1$. Then:

$$D(P,Q) = \sqrt{\sum_{i>0} \lambda_i^{-1}(\hat{p}(i) - \hat{q}(i))^2}$$

Note that this can be interpreted as the L_2 distance between $\hat{p}$ and $\hat{q}$ according to a measure ν which assigns i mass λ_i^{-1}. That is, $D(P,Q) = \int (\hat{p} - \hat{q})^2 d\nu$. Thus, D can at least be regarded as a Distance in the Fourier domain. Consequently, D inherits some nice properties:

- $D(P,Q) = 0 \Leftrightarrow \Psi^T(p-q) = 0 \Leftrightarrow \int \widehat{p-q} d\nu = 0$.
- $D(P,Q) = \int (\widehat{p-q})^2 d\nu = \int (\widehat{q-p})^2 d\nu = D(Q,P)$
- $D(P,Q) + D(P,R) = \int (\widehat{p-q})^2 d\nu + \int (\widehat{p-r})^2 d\nu \leq \int (\widehat{q-r})^2 d\nu = D(Q,R)$ since the $L2$ norm is a norm. Here, we let r be the vector representation of some arbitrary third measure on V called R.

Thus, $D_1(\cdot,\cdot)$ is a valid norm. By multiplying the above statements by μ, it follows that D_μ is a valid distance as well. $\square$

Remark: While it was the guiding intuition, the fact that our affinities be related to distances. It's only necessary that our affinities are positive and symmetric to construct a positive semidefinite Laplacian matrix.

Approximation

If the full Generalized Inverse L^- is too expensive to calculate, we could also just approximate this as:

$$D_\mu(P,Q) \approx D_\mu^k(P,Q) = \mu\sqrt{\sum_{0<i\leq k} \lambda_i^{-1}\langle p-q, \psi_i\rangle}$$

Where $\lambda_1^{-1} \geq \lambda_2^{-1} ... \geq \lambda_3^{-1}$. Although this requires computing the smallest k eigenvalues and eigenvectors, which can be a bit laborious.

Comparing to the Uniform

So now let Q be the uniform distribution over the vertices. Since q is thus the constant vector, $\hat{q} = \delta_0$, so our distance D becomes:

$$D_\mu(P,Q) = \mu\sqrt{\sum_{i>0} \lambda_i^{-1}(\hat{p}(i) - \delta_0)^2} = \mu\sqrt{\sum_{i>0} \lambda_i^{-1}\hat{p}(i)^2} = \mu\sqrt{pL^{-}p}$$

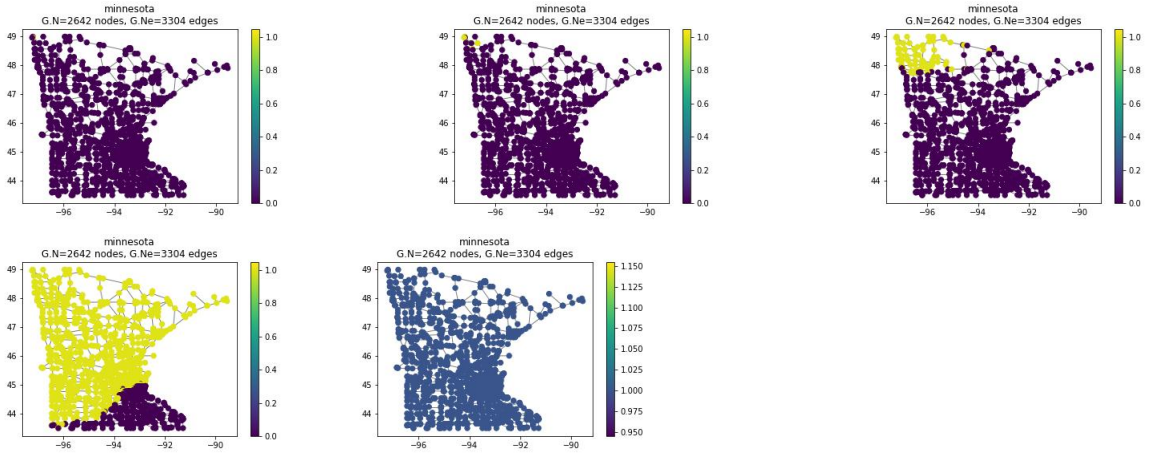
On the Bunny Graph

If we run this on, for example, the Bunny graph, we find that as we diffuse a signal from a dirac out into the graph, the resulting distanes to the uniform are given by 0.11598975, 0.09482865, 0.08496759, 0.08078381.... 0.01305662 using a scale 2^{10}. Every time we diffuse the signal, the distance to the uniform decreases, which is perfectly sensible!

On the Minnesota Graph

Another thing we might try is considering signals of the form $\sigma(A^{2^i})\delta_a$, where $\sigma(x) = \mathbf{1}\{x > 0\}$. Since $A^{2^i}(a,b)$ counts how many paths of length 2^k from nodes a to b, $\sigma(A^{2^i})\delta_a(b) = \mathbf{1}\{\text{b accessible from } a \text{ in} \leq 2^k \text{ steps}\}$. If the graph, is fully connected, we have $\delta_a \leq A\delta_a \leq \sigma(A^2)\delta_a ... \uparrow 1$. Thus, we might hope that as we increase k, our signal becomes nearer and nearer to the uniform distribution, according to our measure. Accordingly, using $k = 0, 2, 4, 6, 8$, the resulting distances to the uniform are given by $208.446, 179.184, 125.049, 27.839$, and $9.60596706e-08$. In other words, this is exactly what we might hope for.

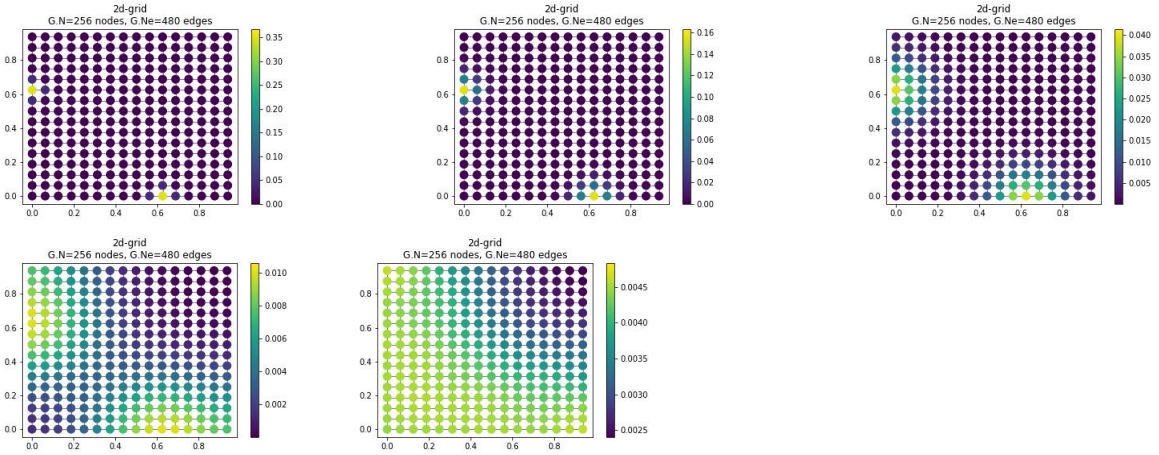
Figure 7: Neighbors of Neighbors on the Minnesota Graph



Spreading Bimodal Signals on the Grid Graph

Another thing we might try to do is fix two center vertices and diffuse the signal spreading out from those. We'd imagine the level of locality will decrease:

Figure 8: Spreading out a Bimodal Signal

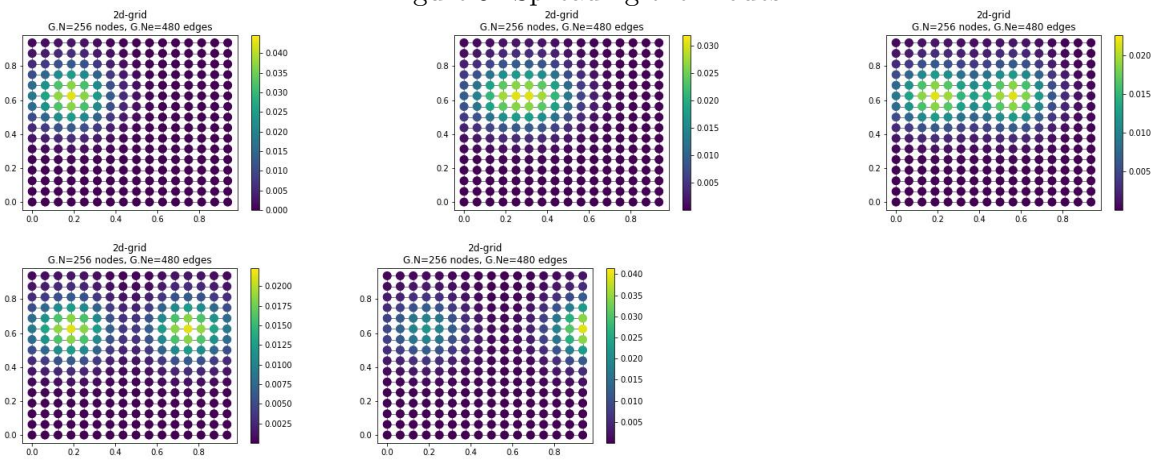


Accordingly, we find that the measures of locality for these signals are: 17.019, 13.807, 9.775, 5.417, and 1.614. Which is exactly what we might hope for.

Moving Signals Away

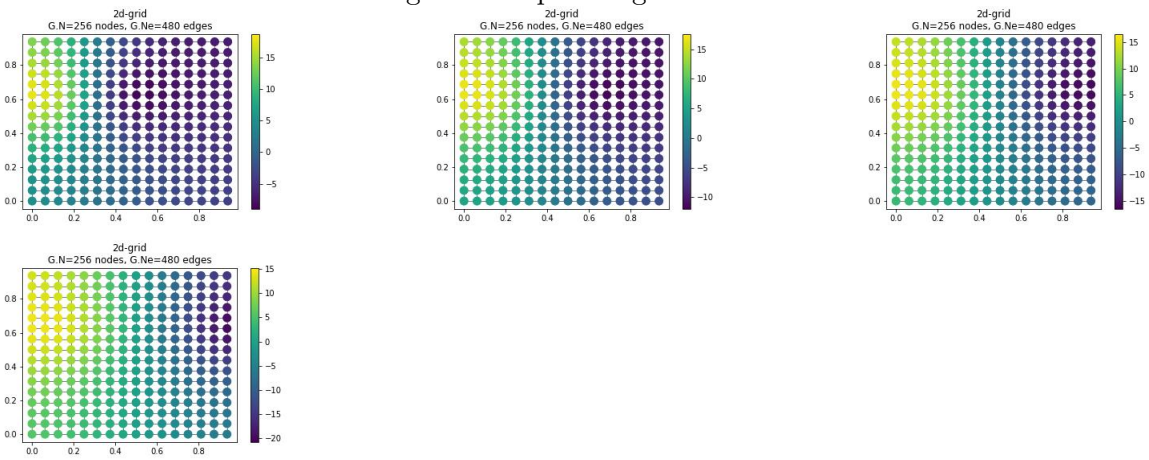
Alternatively, we might try moving a composite of two signals farther and farther away from each other to see how that changes. By moving two modes farther apart until they pinch apart, we'd imagine the signal becomes gradually more local:

Figure 9: Spreading the Modes



And in this example, we see that the corresponding measures of locality are given by 13.743, 11.044, 8.33, 7.418, and 8.654. For fun, we could also try viewing the modes as two different signals, and calculating he "distance" between them. Here are the corresponding witness functions for the last four (since in the first example, the signals are identical, any witness function would do). Then the witness functions are:

Figure 10: Spreading the Modes

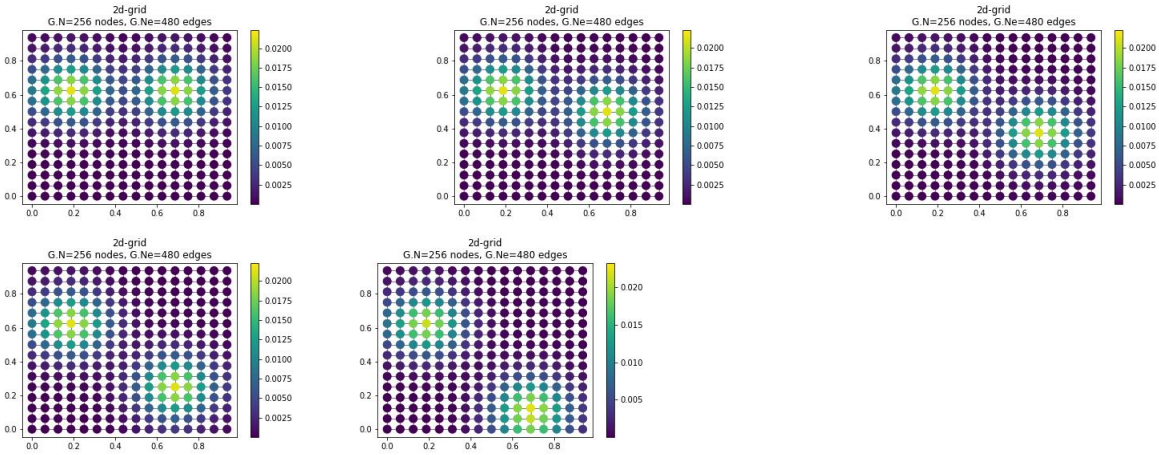


Likewise, we find the calculatd distance between them becomes, for the above progression: 0, 10.198, 17.731, 23.137, and 26.897. Since the signals are moving away from each other, that's the expected behavior.

Shift Invariance

Another thing we might imagine is some notion of shift invariance: that if we move a signal around, the measure of locality wouldn't change much, so long as the signals are behaviorally the same.

Figure 11: Translation of Modes



And again, we find that the values are 7.409, 6.337, 5.7545, 6.0168, and 7.237. So they're not exactly the same, but they are all roughly equal, which is quite nice! Again, we can view the modes as separate signals, and take the distance between them. If we do this, we get that the distances, for the above signals, are 21.476, 21.544, 22.599, 24.373, and 26.662. Since we're physically moving the signals farther and farther apart, this is what we'd hope for.

1 Hypothesis Testing

Hypothesis Testing in Fourier Domain

Another thing we might try is to test for some level of significance in difference from the uniform. Here is a fairly naive attempt. First, we can devise a fairly crude model of signal generation. We will assume that graph signals are generated according to following model:

$$s = \sum_i \mathcal{N}(\mu_i, \sigma^2)\psi_i$$

Where ψ_i is the corresponding graph Laplacian. Notice that in the case of the uniform distribution, we'd simply set $\mu_i = \sqrt{n}\delta_0$ and $\sigma = 0$. So we consider the hypothesis that a signal is uniformly distributed: $H_0 : \mu_i = \delta_0(i)$. Alternatively, we have H_1, in which μ_i are able to vary freely. Recall that we can use as a test statistic the generalized log-likelihood ratio between the models:

$$-2\log\Lambda \quad \text{such that} \quad \Lambda = \frac{\max_{\theta_0 \in H_0} f(s|\theta_0)}{\max_{\theta \in H_1} f(s|\theta)}$$

Let's go ahead and compute the maximum likelihood estimates for each model. First, what is $\max_{\theta_0 \in H_0} f(s|\theta_0)$? The likelihood of our signal is the same as the likelihood as its Fourier transform $\hat{s}$, which is as follows:

$$\ell(s|\theta) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mu_i - \langle s, \psi_i \rangle)^2}{2\sigma^2}}$$

In $H_0, \mu_i = \delta_0$, so this simply becomes:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(1 - \langle s, \psi_0 \rangle)^2}{2\sigma^2}} \cdot \prod_{i>0} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\langle s, \psi_i \rangle)^2}{2\sigma^2}}$$

Taking the logarithm:

$$= -n/2\log(2\pi) - n\log(\sigma) - \frac{(1 - \langle s, \psi_0 \rangle)^2}{2\sigma^2} - \sum_{i \geq 1} \frac{\langle s, \psi_i \rangle^2}{2\sigma^2}$$

Setting derivatives equal to 0, we obtain:

$$\frac{\partial}{\partial \sigma} \log(l(s|\theta_0)) = -\frac{n}{\sigma} + \frac{(1 - \langle s, \psi_i \rangle)^2}{\sigma^3} + \sum_{i \geq 1} \frac{\langle s, \psi_i \rangle^2}{\sigma^3} = 0$$

Which is solved for

$$\sigma^2 = \frac{1}{n}((1 - \langle s, \psi_1 \rangle)^2 + \sum_{i \geq 1} \langle s, \psi_i \rangle^2)$$

We can be a little clever and note that $\sum_{i \geq 1} \langle s, \psi_i \rangle^2 = \|s\|^2 - \langle s, \psi_0 \rangle^2$, since $\{\psi_i\}_i$ is an orthonormal basis. Thus, we obtain:

$$\sigma^2 = \frac{1}{n}((1 - \langle s, \psi_0 \rangle)^2 + \|s\|^2 - \langle s, \psi_0 \rangle^2) = \frac{1}{n}(1 - 2\langle s, \psi_0 \rangle + \|s\|^2) = \frac{1}{n}(1 - 2\bar{s} + \|s\|^2)$$

Which is a fairly reasonable estimate. Thus, we set $\theta_0^* = (\sigma, \mu_0 = 1, \mu_2 = 0...)$. What about for the larger model? We have total freedom in the latter model: we can just set $\sigma = 0$ and the μ's to our observed Fourier transform. And we get a likelihood of 1. Finally, observe that H_0 is 1 dimensional while H_1 is $n + 1$ dimensional. Thus, for large n,

$$-2 \log \Lambda = -2 \log(l(s|\theta_0^*)) \sim \chi_n^2$$

Then testing to see if $-2 \log \Lambda \geq \chi_n^2(\alpha)$, where $\chi_n^2(\alpha)$ is the $1 - \alpha$ quantile of the χ_n^2 distribution, could provide a reasonable test of H_0. Note we could make this more interesting if we had "multiple observations" from of the same signal distribution, so that we would get a nontrivial MLE for the larger model. Note that, after sufficient cancellation, graph structure is not taken into account at all. This could be modified by instead supposing the signal is generated as a linear combination of some subset of the spectrum and developing an analagous test.

Hypothesis Testing using our Locality Measure

One thing we might suppose, more simply, is that our signal is generated with $s(i) = 1/n + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$ and all the errors are i.i.d. Then we can look at the distribution of $D(S, U)$, where S is the distribution induced by our signal and U is the uniform distribution over the vertices. First, we estimate $\hat{\sigma}$ as the variance of our signal s. For the purposes of constructing confidence intervals, it'd be more prudent to consider $D(S, U)^2$, which is:

$$s^T L^- s = (\frac{1}{n}\mathbf{1} + \vec{\epsilon})^T L^- (\frac{1}{n}\vec{1} + \vec{\epsilon}) = \frac{1}{n^2}\mathbf{1}^T L^- \mathbf{1} + \frac{2}{n}\mathbf{1} L^- \vec{\epsilon} + \vec{\epsilon}^T L^- \vec{\epsilon}$$

Of course, since 1 is in the null space of L^-, and L^- is symmetric:

$$= \vec{\epsilon}^T L \vec{\epsilon} = \sum_{i>0} \lambda_i^{-1} \langle \epsilon, \psi_i \rangle^2 = \sum_{i>0} \lambda_i^{-1} \langle \epsilon, \psi_i \rangle^2$$

One final thing to note is that since $\Psi^T \epsilon$ is just a rotation of ϵ, it will also be $\mathcal{N}(0, \sigma^2 I)$, meaning the below can also be represented as:

$$\sum_{i>0} \lambda_i^{-1} \sigma^2 Z_i^2$$

Where $Z = (Z_1 ... Z_n) = \Psi^T \epsilon$ is the Fourier transform of ϵ. We would then like to figure out a distribution function for the above. Of course,

$$\mathbb{E}[\sum_{i>0} \lambda_i^{-1} \sigma^2 Z_i^2] = \sum_{i>0} \sigma^2 \mathbb{E}[Z_i^2] = \sigma^2 \sum_{i>0} \lambda^{-1}$$

Likewise,

$$\text{Var}(\sum_{i>0} \lambda_i^{-1} \sigma^2 Z_i^2) = \sum_{i>0} \lambda_i^{-2} \sigma^4 \text{Var}(Z_i^2)$$

Finally, note that since the variance of a χ^2 distribution is 2, the total variance is $2\sigma^4 \sum_{i>0} \lambda_i^{-2}$. Now, we would like to apply Chebyshev's Inquality to obtain a reasonable cutoff. Note if we want:

$$\mathbb{P}\{\sum_{i>0} \lambda_i^{-1}\sigma^2 Z_i^2 > c\} \leq \alpha$$

This means

$$\mathbb{P}\{\sum_{i>0} \lambda_i^{-1}\sigma^2 Z_i^2 > c\} \leq 1/(1/\sqrt{\alpha})^2$$

Is true of $c = \sqrt{2\sigma^4 \sum_{i>0} \lambda_i^{-2}} \cdot 1/\sqrt{\alpha} = \sigma^2\sqrt{2/\alpha \sum_{i>0} \lambda_i^{-2}}$. Thus, a reaasonable hypothesis is: reject H_0 if $|p^T L^- p - \hat{\sigma^2} \sum_{i>0} \lambda^{-1}| \geq \hat{\sigma}^2\sqrt{2/\alpha \sum_{i>0} \lambda_i^{-2}}$. While the exact cutoff could be improved, namely because this would work just as well for a two-sided test, and Chbyshev's only gave us a naive bound, this is a nice start.

As a quick sanity check, I ran the above method of hypothesis testing on the Minnesota graph. The first thing to do is simulate from the null hypothesis. What I find is that the test is conservative. That is, in 10000 simulations, only 9 of them rejected the null hypothesis. On the other hand, for the earlier signals on the Minnesota graph, we rejected the null hypothesis for all of them, which is pretty good news. Of course, one fast workaround to the issues with this test would be to just generate a bootstrap estimate of the necessary one-sided cutoff.

Bootstrapping

We can try to estimate the distribution of $\sqrt{Tz^T L^- z}$ for $z \sim \mathcal{N}(1/n, \hat{\sigma(s)}^2)$, where $\hat{\sigma}(s)$ is the sample standard deviation of our sample. Using the example from "Spreading Bimodal Signals on the Grid Graph," and $B = 1000$ simulations, we find that our estimated p-values are all significant at the $\alpha = 0.05$ level, which is what we might hope.

More Exact Calculation

Another thing we could try to do is directly calculate the distribution function for a linear combination of χ^2 random variables. Let $X = \sum_i a_i X_i$, where the X_i's are independent χ^2 random variables, and a_i is a fixed coefficient. We know the moment generating function of this distribution will be:

$$M_X(t) = \mathbb{E}[e^{itX}] = \mathbb{E}(e^{it\sum_i a_i X_i}) = \prod_i \mathbb{E}[e^{ita_i X_i}]$$

Substituting he usual formula for the χ^2 characteristic functions:

$$= \prod_i (1 - 2ita_i)^{-\frac{1}{2}}$$

Ideally, we'd be able to relate the moment generating function to the CDF. Suppose that a random variable X with a continuously differentiable PDF f has a Fourier Transform given by $c(t) = \mathbb{E}(e^{itX})$. We know that,

$$\mathbb{P}\{X \leq a\} = \int_{-\infty}^{a} f(x)dx = \frac{1}{2\pi}\int_{-\infty}^{a}\int_{-\infty}^{\infty} e^{itX}\overline{c(t)}dtdx$$

We could take a kth order Taylor expansion of $c(t)$ about 0:

$$c(t) = \sum_{j=0}^{k-1}(\frac{t^j}{j!})\frac{\partial^j}{\partial t^j}c(t)|_{t=0} + \frac{\partial^k}{\partial t^k}c(\xi)$$

Also, note that for each j,

$$\frac{\partial^j}{\partial t^j}c(t) = \mathbb{E}[\frac{\partial^j}{\partial t^j}e^{itX}] = \mathbb{E}[(iX)^j e^{itX}]$$

Thus,

$$\frac{\partial^j}{\partial t^j}c(t) = i^j\mathbb{E}[X^j]$$

So our Taylor Expansion becomes:

$$c(t) = \sum_{j=0}^{k-1}\frac{1}{j!}(-it)^j\mathbb{E}[X^j] + \frac{\partial^k}{\partial t^k}c(\xi(t))$$

Substituting this in, we find:

$$\mathbb{P}\{X \leq a\} \approx \frac{1}{2\pi}\int_{-\infty}^{a}\int_{-\infty}^{\infty} e^{itx}\sum_{j=0}^{k-1}\overline{i^j}\cdot t^j\cdot\left[\frac{1}{j!}\mathbb{E}[X^j]\right]dtdx$$

Or exchanging integrals with sums:

$$= \int_{-\infty}^{a}\sum_{j=0}^{k-1}\left[\frac{1}{j!}\mathbb{E}[X^j]\right]\overline{i^j}\left\{\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{itx}\cdot t^j dt\right\}dx$$

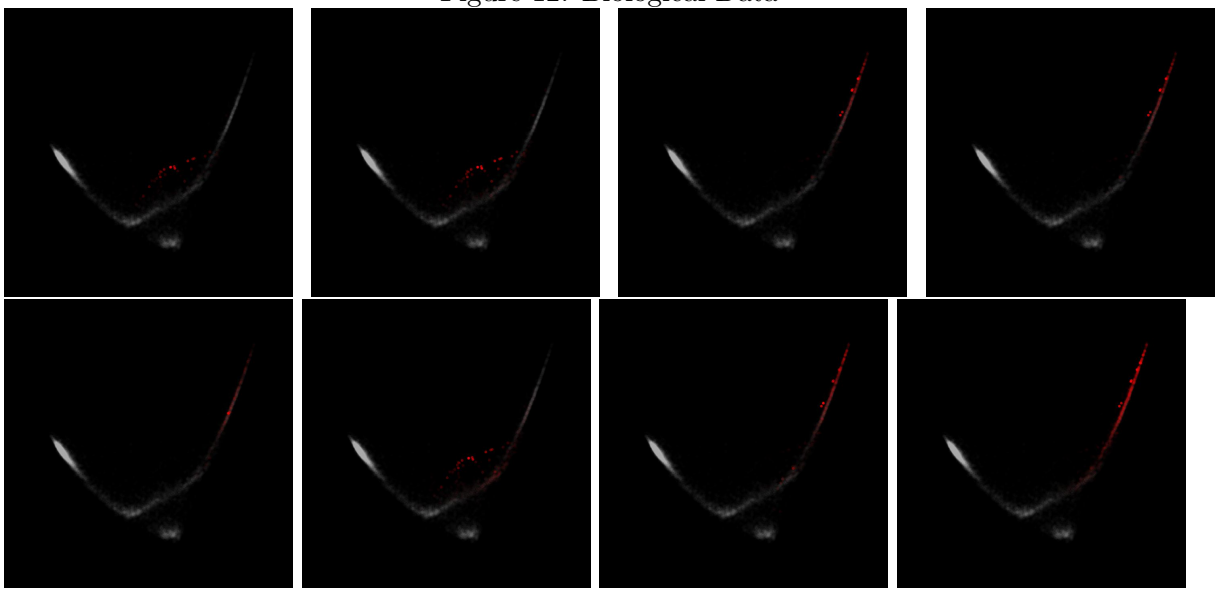
To be continued...

Testing on Biomedical Data

Retrieving Localities

We can go ahead and look at our single cell data and see which genes are the most "local." Here, we will use slightly diffused versions of the signals (via an order 32 chebyshev approximation). Here are the 8 most "local" gene signals according to our metric:

Figure 12: Biological Data



Comparing Signals

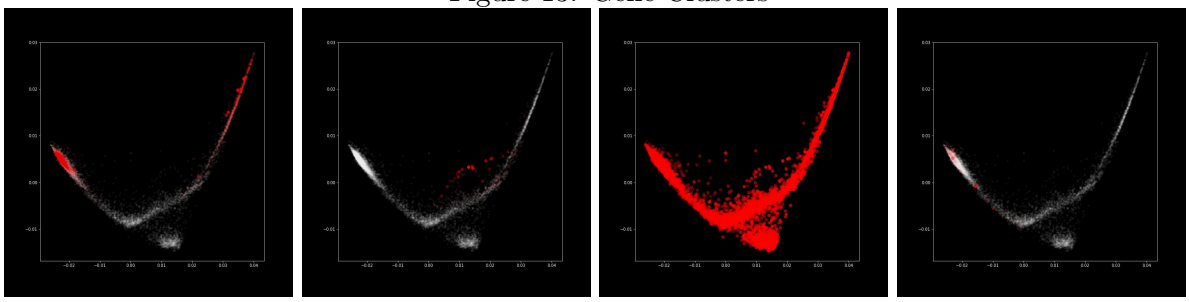
Recall that our distance between two signals P and Q (viewed as distributions) is $\sqrt{T(P-Q)^T L^-(P-Q)}$. Suppose we want to compute this efficiently for all P, Q. Note that we have:

$$(P-Q)^T L^-(P-Q) = P^T L^- P - 2P^T L^- Q + Q^T L^- Q$$

What we can first do is calculate $L^- P$ for each signal P. This will take time at most $\mathcal{O}(n^2 d)$, if we have d signals. Using these, we can calculate $P^T L^- P$ for all P (and thus Q) in time $\mathcal{O}(nd)$. Then we have to calculate the cross terms in time $\mathcal{O}(nd^2)$. So total computation is $\mathcal{O}(n^2 d + d^2 n + nd) = \mathcal{O}(nd(d+n))$. Note that this is an improvement over the naive $\mathcal{O}(n^2 d^2)$ approach if compute everything at each step. We can easily take the transformations necessary to get $D(P, Q)$ for each P, Q from there.

Using this metric, we can create a distance matrix D and run hierarchical clustering. Here, we ran it using average linkage. Using a tolerance of 1.1547002, we got about 30 clusters, but of very inconsistent sizes. One cluster had 1879 genes, the next had 68, then 8, then 6. I did not consider clusters with 3 or fewer genes. Here are the average signal of each of those clusters, by descending order of the size of the cluster:

Figure 13: Gene Clusters



For later: biological interpetation of the clusters.

Appendix

Selecting $\tilde{\Psi}_j$ — Algorithm Correctness

Proposition: $\tilde{\Psi}_j$ as outputted by Algorithm 1 satisfies equation (1). *Proof:* This is equivalent to proving:

$$\|\Psi_j - \tilde{\Psi}_j(\tilde{\Psi}_j{}^T\tilde{\Psi}_j)^{-1}\tilde{\Psi}_j^T\Psi_j\|^2 \leq \epsilon\|\Psi_j\|^2 = Q \cdot R(p(1)...p(i))$$

Recall that our algorithm first takes a pivotal QR decomposition of Ψ_j. Our algorithm first finds i such that $\|\Psi_j\| - \sum_{k=0}^{i}\sum_{j=k}^{n} R_{k,j}^2 \leq \epsilon\|\Psi_j\|$. Or equivalently, so that $\|\sum_{k=0}^{i}\sum_{j=k}^{n} R_{k,j}^2\| \geq (1-\epsilon)\|\Psi_j\|$. First, observe that $\|\Psi_j\|^2 = \|QR\|^2 = \|R\|^2$ because Q is orthonormal.

Observe that for all $k \leq i$ that $R_{k,k} > 0$ per how the pivotal QR decomposition algorithm works. If any columns are linearly dependent, those will be attributed to indices higher than i. I also claim that. Thus, $C(\tilde{\Psi}_j) = C(q_1...q_i)$, where $q_1...q_i$ are the first i columns of Q. Thus, if we assemble these columns into a matrix Q_i, we find that the projection matrix onto $C(Q_i)$ is the same as the projection onto $\tilde{\Psi}_j$. Thus,

$$\|\Psi_j - \tilde{\Psi}_j(\tilde{\Psi}_j{}^T\tilde{\Psi}_j)^{-1}\tilde{\Psi}_j^T\Psi_j\|^2 = \|\Psi_j - Q_iQ_i^T\Psi_j\|^2$$

Suppose the columns of Ψ_j are $\psi_j^1...\psi_j^n$. We can also compute the above norm by column:

$$= \sum_k \|(I - Q_iQ_i^T)\psi_j^k\|^2$$

But since Q is orthonormal and ψ_j is in the span of Q, we can represent $(I - Q_iQ_i^T)$ by the matrix $\hat{Q}_i\hat{Q}_i{}^T$, where $\hat{Q}_i$ contains columns $i+1...n$ of Q. So:

$$= \sum_k \|\hat{Q}_i\hat{Q}_i{}^T\psi_j^k\|^2$$

Note that p is a permutation, so we can rewrite the order as:

$$= \sum_k \|\hat{Q}_i \hat{Q}_i^T \psi_j^{p(k)}\|^2$$

Of course, observe that by the way the QR decomposition works, column $p(k)$ of Ψ_j is:

$$\psi^{p(k)} = \sum_{l \leq k} q_l R_{l,k}$$

Plugging this in, our original norm is: r

$$= \sum_k \|\hat{Q}_i \hat{Q}_i^T \sum_{l \leq k} q_l R_{l,k}\|^2 = \sum_k \| \sum_{l \leq k} \hat{Q}_i \hat{Q}_i^T q_l R_{l,k}\|^2$$

Since the columns of q_l are orthonormal:

$$= \sum_k \| \sum_{l \leq k} \mathbf{1}\{l > i\} q_l R_{l,k}\|^2$$

Applying Parseval's identity:

$$= \sum_k \sum_{l \leq k} \mathbf{1}\{l > i\} R_{l,k}^2$$

Which is precisely equal to the *sums of squares* of rows $i+1...n$ of R. Thus,

$$\|\Psi_j - \tilde{\Psi}_j (\tilde{\Psi}_j^T \tilde{\Psi}_j)^{-1} \tilde{\Psi}_j^T \Psi_j\|^2 = \sum_{k>i} \sum_{l \geq k} R_{k,l}^2 = \|R\|^2 - \sum_{k \leq i} \sum_{l \geq k} R_{k,l}^2$$

By design,

$$\|\Psi_j\|^2 - \sum_{k \leq i} \sum_{l \geq k} R_{k,l}^2 \leq \epsilon \|\Psi_j\|^2$$

Thus, the claim must be true as claimed $\square$.