Isometrically reconstructing data using the Diffusion Maps Laplacian

Orton Babb

Abstract

In topological data analysis we think of data points as having been sampled from a manifold. We used diffusion maps to approximate the Laplace-Beltrami Operator of the underlying manifold, and thus approximate the eigenfunctions of the Laplacian. Using a variety of numerical methods we attempted to use these eigenfunctions as a basis for isometrically reconstructing the data in a lower dimension

This research was co-presented at the Joint Mathematics Meetings' Mathematical Association of America's Undergraduate Poster Session with Aneesh Malhotra and received an Outstanding Poster Award. The early research was also shared with faculty and students of GMU at Mason Experimental Geometry Lab Fall Symposiums 2017, 2018. The project was directed by Dr. Tyrus Berry and other collaborators include Yemeen Ayub and Ryan Vaughn. I have done my best to faithfully document the project results here and highlight some of my contributions to the project. All errors my own!

1 Introduction

Dimensionality reduction (DR) has found its practical motivation in a myriad of use cases including processing of sensor arrays, image processing, multivariate data analysis and data mining. DR involves simplifying the data into a smaller set of variables which can then provide greater understanding of the current data set and generalize to new out-of-sample data (Lee 2007). In each case, information extracted from data can be used (either by the analyst or another algorithm) to perform other tasks such as data visualization.

Recovering the Laplacian (or more precisely the Laplace-Beltrami operator) has become one popular means of this information extraction in manifold learning. Diffusion maps, due to Coifman and Lafon, offers a method of construction for this Laplacian and has become a point of reference for these methods.

As shown in Figure 1, diffusion maps have the capacity to handle difficult nonlinear cases. Porte et al. have motivated their introduction to diffusion maps via previously existing methods such as Principal Component Analysis (PCA), Multidimensional Scaling(MDS) and Isometric Feature Map (Isomap). Following this outline, we give a summary of each method and some practical

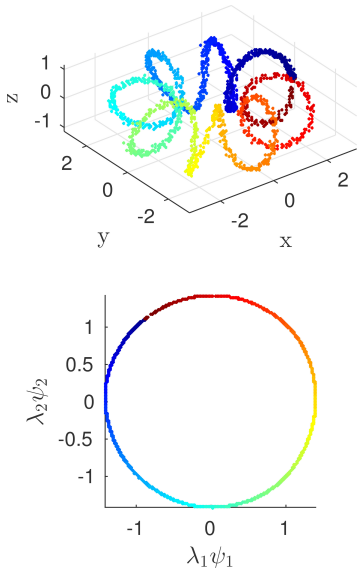

Figure 1: Given non-uniformly sampled data points on a toroidal helix (top), the first two Diffusion Map coordinates with Laplace–Beltrami normalization are plotted (bottom). (via Wikipedia)

limitations. Next, the diffusion maps approach is covered in detail. Finally, we point to a new non-linear method, which extends the diffusion maps construction of the Laplacian, with the goal of augmenting existing methods.

2 Existing Linear and Non-linear Dimensionality Reduction

In linear dimensionality reduction, two well known techniques are Principal Component Analysis (PCA) and Multidimensional Scaling (MDS).

PCA aims to find a linear projection from from an n-dimensional space down to an m dimensional subspace (where $m < n$). (See Jolliffe's Principal Component Analysis, for the definitive text on PCA). Once the eigenvectors and values of the covariance matrix of the centered data have been computed, this m-dimensional space is a projection of the points onto the first m principal components. However, while the implementation is simple, linearity of the data is only a special case.

In a similar setting, MDS (introduced by Torgerson) works by computing the distance matrix (classically, the pairwise Euclidean distances) of data X embedded in an n-dimensional space and tries to find a new embedding Y which minimizes the cost, $\rho_{strain}(D_X, D_Y) = \|J^T(D_X^2 - D_Y^2)J\|_F^2$ where J is a matrix such that $J^T X J$ is centered and the Frobenius norm $\|X\|_F$ is just

the root sum-of-squares of the components. While the minimization problem has a solution given by the dominant eigenvectors of the matrix $-\frac{1}{2}J^T D_X^2 J$, MDS is similarly limited in that the Euclidean distance can only capture local similarities so that large distances are only informative when the shape of the data is linear.

In non-linear dimensionality reduction, Isometric Feature Map (introduced by Tenebaum et al.) extends MDS by preserving the geodesic distance, rather than the Euclidean distance, between points. The geodesic distance is approximated locally by Euclidean distances, while along further points, it is approximated by the shortest connecting path. Once the distance matrix is computed in this way, MDS can be performed. The shortcoming of this method is its lack of robustness when data is noisy and the time complexity of calling a Dijkstra like method for each pairwise comparison.

The Diffusion Maps approach has been described by Cofiman and Lafon as "a framework based upon diffusion processes for finding meaningful geometric descriptions of data sets." It is an improvement over Isomap due to its robustness (De la Porte et al 2008) and can be seen as analogous to doing MDS on the so called "diffusion distances" which can approximate geodesic distances. For our purposes, it remains to show how these diffusion distances are calculated.

Consider a random walk on the data where the probability of jumping from one point the next is determined by the closeness between the points. We can assign these probabilities using kernel methods. A common kernel is the Gaussian kernel which can be defined as

$$k(x_i, x_j) = e^{-\frac{\|x_i - x_j\|_E^2}{\epsilon^2}}$$

for some kernel bandwidth $\epsilon > 0$. Notice that k takes on a value of one when points are exactly on top of each other and approximately zero when points are far apart (particularly due the fast exponential decay as $\|\mathbf{x}_i - \mathbf{x}_j\|_E$ grows larger). Then, we can define

$$d_{\mathbf{x_i}}^{-1} = [\sum_j k(\mathbf{x}_i, \mathbf{x}_j)]^{-1}$$

so that for every $(\mathbf{x}_i, \mathbf{x}_j)$ we have a

$$p(\mathbf{x}_i, \mathbf{x}_j) = d_{\mathbf{x}_i}^{-1} k(\mathbf{x}_i, \mathbf{x}_j).$$

More generally, we can construct a matrix K with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ to derive the row stochastic matrix $P = D^{-1}K$. The matrix P^1 indicates that the random walk takes only a single time step, whereas for a longer number of time steps t, we have the more general matrix P^t. Diffusion distances are then defined as the Euclidean distances between points called diffusion maps which are a re-embedding of the original points in "diffusion space." That is, for every $\mathbf{x}_i$ we have

$$Y_i = (p(\mathbf{x}_i, \mathbf{x}_1), ..., p(\mathbf{x}_i, \mathbf{x}_N))$$

so that $D(\mathbf{x}_i, \mathbf{x}_j)^2 = \|Y_i - Y_j\|_E^2$. At large time scales the paths will be restricted to the geodesic. The diffusion distance is then

$$D_t(\mathbf{x}_i, \mathbf{x}_j)^2 = \sum_k |P_{ik}^t - P_{kj}^t|^2.$$

This by itself doesn't accomplish dimensionality reduction since points in diffusion space are N-dimensional where N is the number of data points. To accomplish the reduction, find U, V such that $P = UVU^T$, where columns of U are eigenvectors and V is a diagonal matrix with entries $V_{kk} = \lambda_k$. Then,

$$Y_i' = (\lambda_1^t U_{i1}, ..., \lambda_N^t U_{iN})$$

and if we choose to keep only the first $M < N$ eigenvalues/vectors, we have, $Y_i^t = (\lambda_1^t U_{i1}, ..., \lambda_M^t U_{iM})$ which allows for an approximation of the diffusion distance in less than N dimensions:

$$\begin{aligned} D_t(\mathbf{x}_i, \mathbf{x}_j)^2 &= \sum_{k=1}^{N} |P_{ik}^t - P_{kj}^t|^2 \\ &\approx \sum_{k=1}^{M} \lambda_k^{2t} (U_{ik} - U_{jk})^2 \\ &= \|V(U_{i*} - U_{j*})\|^2 \end{aligned}$$

This spectral approach is what makes it more robust to noise than Isomap and is also the reason for our analogy of "doing MDS" but with diffusion distances.

3 New directions

The matrix P described in the previous section has an alternative geometric interpretation as a recovery the Laplacian, L, on some manifold from which the data was sampled. In personal correspondence, Tyrus Berry has suggested a new direction for non-linear dimensionality reduction based on this construction of the Laplacian. The method assumes that the data is sampled from a compact Riemannian manifold, $\mathcal{M}$, embedded in Euclidean space and that the coordinates of the data are functions on $\mathcal{M}$, i.e. $\mathbf{x}_i = (f_1, ..., f_m)$, where $f_i : \mathcal{M} \to R$.

Following Coifman and Lafon, an approximation of the Laplacian operator, $\Delta_{\mathcal{M}}$, can be recovered with the eigenvectors of L approximating the eigenfunctions of $\Delta_{\mathcal{M}}$. Since these eigenfunctions of the Laplacian, U_k, form a basis for the function space $L^2(\mathcal{M})$, which contains $C^\infty(\mathcal{M})$, we can represent any f_j as the linear combination

$$f_j(z) = \sum_{k=1}^{\infty} U_k(z) C_{kj}$$

where C_{kj} are like generalized Fourier coefficients. In the discrete setting, this would be analogous to

$$\bar{f}_j = UC_{*j}$$

where $\bar{f}$ is the evaluation of f on the data, U is the matrix of orthonormal eigenvectors, and C can be thought of as a matrix of generalized Fourier series coefficients. The best approximation for these coefficients is the inner product $C_{*j} = U^T \bar{f}_j$ which has nice reconstruction properties when data is large and the smoothness assumptions are met. Hence the data can be written as,

$$X = (\bar{f}_1, \bar{f}_2, ..., \bar{f}_m)$$
$$= U[C_{*1}|C_{*2}|...|C_{*m}] = UC$$

This is a result of Berry and Sauer (2016) that establishes a diffeomorphism between manifolds (between $\mathcal{M}$ and R^m in this case) by linear maps that send eigenfunctions to eigenfunctions. A more general result was shown where the invertibility requirement on the morphism is relaxed (Berry 2018). This representation allows us to truncate the number of eigenvectors used to represent functions on the data. This will be useful for dimensionality reduction because if we consider this C matrix to be a parametrization of the data, we can truncate it to make the number of parameters small relative to the size of the data set.

While, there are many C's which preserve the local geometries, some may be more useful to have than others—particularly those with minimal extrinsic curvature. "A smooth isometric embedding with minimal extrinsic curvature" is the working definition provided by Berry for the minimal embedding for $\mathcal{M}$. Before we can describe the possible ways to do this using the $X = UC$ factorization, it is constructive to first look at the continuous case of the minimal embedding problem,

$$F' = Fargmin \underbrace{\int_{\mathcal{M}} ||\det HF(x)^{\perp}||\, dvol}_{ExtrinsicCurvature}$$
$$subjectto \underbrace{DF(x)^{\top} DF(x) = g_x}_{Preservethegeometry}.$$

As an example in the continuous case, consider an embedding of the unit circle S^1 given by

$$x = F(\theta)$$
$$= \mu[cos(\theta), sin(\theta), cos(k\theta), sin(k\theta)]$$
$$where \mu = \frac{1}{\sqrt{1+k^2}}$$

.

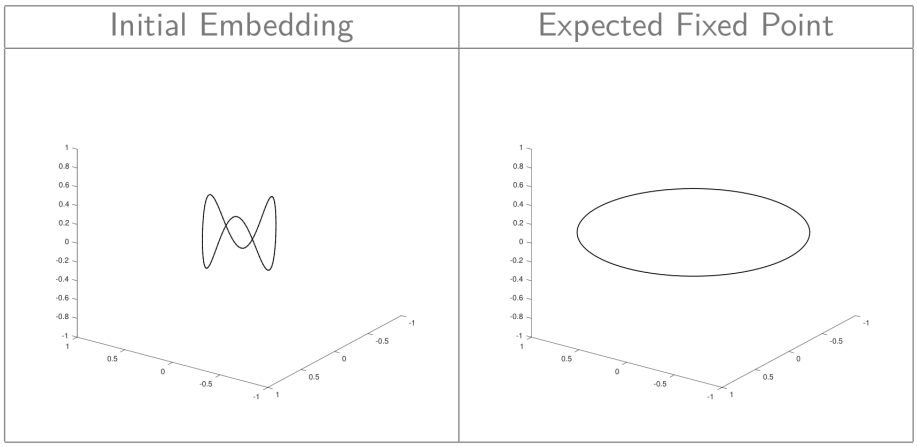

Figure 2: Expected solution for unit circle with an initially high extrinsic curvature

for $k = 3$. Let $F = (f_1, f_2, f_3, f_4)$. We can solve the constrained problem using the method of Lagrange multipliers,

$$\min_{\lambda, f_1, \ldots, f_4} \int_0^{2\pi} \sum_{i=1}^{4} f_i''(\theta)^2 + \lambda(\theta) \left(\sum_{i=1}^{4} f_i'(\theta)^2 - 1 \right) d\theta.$$

Minimizing this functional can be achieved by solving its corresponding Euler-Lagrange equation for which $(cos(\theta), sin(\theta), 0, 0)$ is a solution. This solution, which is plotted in Figure 2, is equal to the flat unit circle (upto rotation and translation) and would also be the expected fixed point for any numerical scheme devised to solve the same problem on a date set sampled from the embedding in this example.

Estimating the extrinsic curvature as a function of C has been accomplished so far by extending a result of Hein et al regarding the continuous analogue of $K\bar{f}$ (Hein et al 2005, p. 476) which is (the discrete version of) the inner product of the kernel with a given function $\int_{\mathcal{M}} k(x,y)f(y)dy$. By the inner product of the kernel matrix with the constant function $K\mathbf{1}$, which is like $\int_{\mathcal{M}} k(x,y) \cdot 1 dy$ we derive a curvature estimate at each data point, with

$$\sum_i (K\mathbf{1})_i = \sum_{ij} K_{ij}$$

being an estimate of the average curvature. Recall that $L = D^{-1}K$. If we fix the original kernel matrix as K_X and define an updatable kernel matrix, K_C, a natural way to find a minimal embedding suggests itself:

$$\min_C \sum_{i=1}^{N} (K_C \mathbf{1})_i$$

$$subjectto (L_X - L_C)_{ij} = 0, \ \forall i, j$$

or with a relaxation terms on the constraint,

$$\min_C \sum_{i=1}^{N} (K_C \mathbf{1})_i$$

$$subjectto (L_X - L_C)_{ij} \leq \varepsilon_{ij}, \ \forall i, j.$$

It is still unclear whether this is the right way to approach minimal embeddings in a discrete environment. There may be ways to isolate the extrinsic curvature more precisely so as not to rely solely on the constraint to prevent distortions of the intrinsic curvature. We know that Nonetheless, this approach may be a viable means of geometric regularization of data, useful in data science where one has reason to believe that the data has the smooth properties we assumed at the outset. One motivating example has been when $\mathcal{M}$ is the circle S^1 embedded in more than 2 dimensions. Minimizing the extrinsic curvature would place the data on a 2D subspace of R^m so that another method such as PCA or MDS can be performed.

4 Contribution

Initially, because of the difficulty of manipulating the Lagrangian, we were using a finite differences method to compute the gradient,

$$\mathcal{L}'_{ij} = \frac{\mathcal{L}(C + hJ_{ij}) - \mathcal{L}(C - hJ_{ij})}{2h}$$

where J_{ij} is a single-entry matrix. However, this method is not only numerically imprecise but relies on nested for-loops which are computationally expensive. If the gradient could be computed analytically it would be help speed up the algorithm.

I found an idea while reviewing Backpropagation for feedforward neural networks that use the sigmoid function, $s(x) = \frac{1}{1+e^{-x}}$, I thought about the fact that if the gradient of the loss function can be computed analytically to exploit the fact that $s'(x) = s(x)(1 - s(x))$, something similar could be done in our algorithm when minimizing the Lagrangian because of the similar exponential in the Gaussian kernel. If E_C is the extrinstic curvature and I_C is the intrinsic curvature, then we need to use matrix calculus to find the gradient of this scalar $E_C + \Lambda I_C$ which is a function of the matrix C. It involves taking the derivative with respect to the entries of C. Here, I show how this can be done for the I_C term and how to implement this efficiently in MATLAB.

Let $I_C = \|L_X - L_{C\,fro}$

$$= D_X^{-1} K_X - D_{\tilde{X}}^{-1} K_{\tilde{X}}(C)_{fro}.$$

Note that,

$$K_C^{ij} = e^{-\frac{(U_{*i} - U_{*j})C^2}{\epsilon^2}}$$

Then,

$$\left(\frac{dI_C}{dC}\right)_{uv} = \frac{\partial}{\partial c_{uv}} I_C$$

$$= \frac{\partial}{\partial c_{uv}} \sum_{ij} \left(L_X^{ij} - \frac{K_C^{ij}}{\sum_r K_C^{ir}}\right)^2$$

$$= \sum_{ij} \frac{\partial}{\partial c_{uv}} \left(L_X^{ij} - \frac{K_C^{ij}}{\sum_r K_C^{ir}}\right)^2$$

$$= \sum_{ij} 2\left(L_X^{ij} - \frac{K_C^{ij}}{\sum_r K_C^{ir}}\right) \frac{\partial}{\partial c_{uv}} \left(-\frac{K_C^{ij}}{\sum_r K_C^{ir}}\right)$$

$$= \sum_{ij} -2\left(L_X^{ij} - \frac{K_C^{ij}}{\sum_r K_C^{ir}}\right)$$

$$\times \left(\frac{\sum_r K_C^{ir} \frac{\partial}{\partial c_{uv}} K_C^{ij} - K_C^{ij} \frac{\partial}{\partial c_{uv}} \sum_r K_C^{ir}}{\left[\sum_r K_C^{ir}\right]^2}\right)$$

Let,

$$Z^{ij} = U_{*i} - U_{*j}$$

.

Note that

$$\frac{\partial}{\partial c_{uv}} K_C^{ij} = -\frac{2c_{uv}}{\epsilon^2} K_C^{ij} (Z_u^{ij})^2$$

and,

$$\frac{\partial}{\partial c_{uv}} \sum_r K_C^{ir} = -\sum_r \frac{2cuv}{\epsilon^2} K_C^{ir} (Z_u^{ir})^2$$

Therefore,

$$\left(\frac{dI}{dC}\right)_{uv} = \sum_{ij} -2\left(L_X^{ij} - \frac{K_C^{ij}}{\sum_r K_C^{ir}}\right)$$

$$\times$$

$$\left(\frac{\sum_r K_C^{ir}(-\frac{2cuv}{\epsilon^2}K_C^{ij}(Z_u^{ij})^2) - K_C^{ij}(-\sum_r \frac{2c_{uv}}{\epsilon^2}K_C^{ir}(Z^{ir}u)^2)}{\left[\sum_r K_C^{ir}\right]^2}\right)$$

$$= \sum_{ij} -2\left(L_X^{ij} - \frac{K_C^{ij}}{\sum_r K_C^{ir}}\right)K_C^{ij}\left(\frac{-2cuv}{\epsilon^2}\right)$$

$$\times \frac{1}{\left[\sum_r K_C^{ir}\right]^2}\left(\sum_r K_C^{ir}(Z^{ij}u - Z^{ir}u)\right)$$

$$= \frac{4cuv}{\epsilon^2}\sum_{ij}\left(L_X^{ij} - \frac{K_C^{ij}}{\sum_r K_C^{ir}}\right)K_C^{ij}\frac{1}{\left[\sum_r K_C^{ir}\right]^2}$$

$$\times \left(\sum_r K_C^{ir}(Z_u^{ij} - Z_u^{ir})\right)$$

It is well known that looping is inefficient in MATLAB compared to matrix operations. To overcome this, we use 3-dimensional arrays (3-tensors) to do the arithmetic operators involved in the gradient expression which would have looped over the 3 indices i, j, and r.

```
% Assume we have Kxbar KxC N M Zsquared DCinv epsilon lambda
rotatedU = reshape(U, [N,1,M]);
MCopiesofU = repmat(rotatedU, [1,N,1]);
MCopiesofUTranspose = permute(MCopiesofU, [2 1 3]);
Z = MCopiesofU - MCopiesofUTranspose;
Zsquared = Z.*Z;
DCinv = diag(1./sum(KxC));
KxCbar = DCinv*KxC;
MCopiesofKhat = repmat((Kxbar - KxCbar).*KxC, [1,1,M]);
repsumKxC = repmat(sum(KxC,1)', [1 N M]);
Lcurv = (-2/(epsilon^2)).*repmat(KxC, [1 1 M]).* Zsquared;
Lisom = ( 4/(epsilon^2)).*(MCopiesofKhat .* (1./repsumKxC.^2)
        .* (Zsquared.*repsumKxC - repmat(sum(Zsquared
        .*repmat(KxC,[1,1,M]),2), [1,N,1])));
insideSummation = Lisom + lambda.*Lcurv;
gradL = C.*repmat(squeeze(sum(sum(insideSum(),1),2)), [1,m])
```

From here you can just do gradient descent to do the minimization problem to find C:

```
for i = 1:MaxIters
    reconstructKernel()
    gradL = LagrangianGradient();
    C = C - learningRate*gradL;
end
```

By using an analytic gradient we improve speed and accuracy. By using loop-less computations, we further increase the speed of calculating the gradient with only a small memory trade-off. This problem formulation also makes calculations easily amenable to parallelization which could further speed up the computation when several cores are available.

5 Conclusion

As we have seen, each approach to DR makes assumptions which may not be met in practice, and each has shortcomings when used in isolation. The linear dimensionality reduction procedures were limited in applicability and the non-linear procedures were either sensitive to noise (Isomap) or dependent on a diffusion time parameter (Diffusion Maps). The common thread between many of these algorithms is in how they integrate local information to stitch together a global picture of the data. The new minimal embedding approach makes this explicit and aims to extend the diffusion maps construction in a new direction. It has promise as a pre-processing mechanism for other algorithms such as PCA and MDS and is an indication that such linear methods can be extended to non-linear instances via a suitable geometric regularization.

6 References

1. W. S. Torgerson, "Multidimensional scaling: I. Theory and method," Psychometrika, 1952.

2. J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science, vol. 290, pp. 2319–2323, 5500 2000.

3. I. T. Jolliffe, Principal Component Analysis. Second Edition. Springer-Verlag New York, 2002.

4. M. Hein, J.-Y. Audibert, and U. von Luxburg, "From Graphs to Manifolds– Weak and Strong Pointwise Consistency of Graph Laplacians," in Learning Theory, P. Auer and R. Meir, Eds., Springer Berlin-Heidelberg, 2005, pp. 470–485.

5. R. R. Coifman and S. Lafon, "Diffusion maps," Applied and Computational Harmonic Analysis, vol. 21, no. 1, pp. 5–30, 2006.

6. J. A. Lee and M. Verleysen, Nonlinear dimensionality reduction. Springer Science Business Media, 2007.

7. J. De La Porte, B. M. Herbst, W. Hereman, and S. J. Van Der Walt, "An introduction to diffusion maps," Proceedings of the 19th Symposium of the Pattern Recognition Association of South Africa (PRASA 2008), 2008.

8. T. Berry and T. Sauer, "Local kernels and the geometric structure of data," Applied and Computational Harmonic Analysis, vol. 40, no. 3, pp. 439–469,2016.

9. T. Berry and J. Harlim, "Iterated diffusion maps for feature identification," Applied and Computational Harmonic Analysis, vol. 45, no. 1, pp. 84–119,2018.