Uncovering Structure in High-Dimensional Data

Joyce A. Chew

Department of Mathematics
University of California, Los Angeles

Calvin University Mathematics Colloquium

High-Dimensional Data

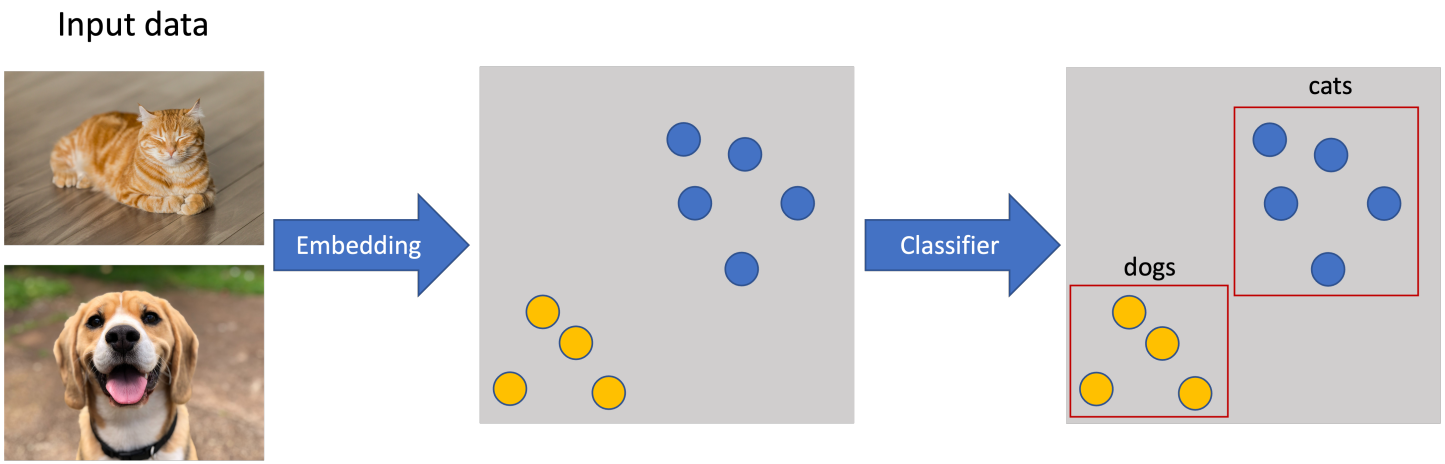
- Question: What is "high-dimensional data"?
- One answer: Each data point is a vector with "many" entries, i.e. there is a lot of information associated with each point.
- Examples:
  - Single-cell sequencing: Each data point corresponds to 1 cell, with counts of the particular enzymes/proteins that cell has
  - Social networks: Each data point corresponds to 1 person, and each person has a relationship with every other person in the network

What can you do with complicated data?

- Use machine learning!
- High-profile successes
  - ChatGPT
  - Image processing/generation
- These applications are all on interesting *domains* (text, images) and take advantage of structure in those domains

Anatomy of a machine learning model

- Many machine learning models can be thought of as an *embedding* together with a *classifier*.
- The embedding transforms each input into a high-dimensional vector.
- The classifier makes predictions based on these high-dimensional vectors.
- Depending on the particular model being used, either part can be "trained"



The manifold assumption

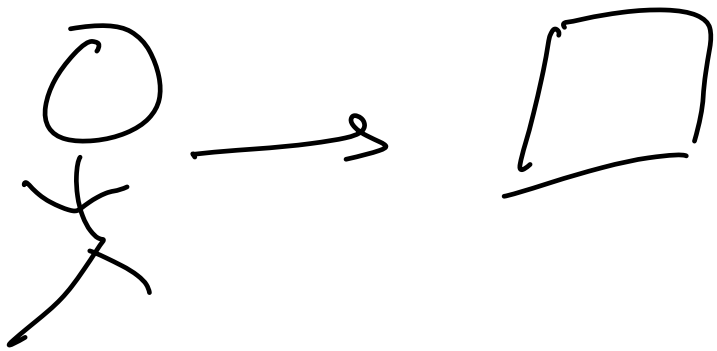
We're going to assume that our data lives on a geometric structure called a *manifold*.

Definition (informal)

A d-dimensional manifold is a structure that "looks like" $\mathbb{R}^d$ when you "zoom in" close enough.

Why make this assumption?

- It works
- Very natural in certain settings
- Manifolds have useful structure!



The Laplace-Beltrami Operator

$(x, f(x))$

Let $\mathcal{M}$ be a d-dimensional manifold. Our goal is to characterize functions $f : \mathcal{M} \to \mathbb{R}$.

Definition

The Laplace-Beltrami operator $\mathcal{L}$ is defined by
$\mathcal{L}f = -\Delta f = -div \circ \nabla f = -\nabla \cdot \nabla f$.

The Laplace-Beltrami operator can be used to write down a heat equation on a manifold,

$$\frac{\partial u(\mathbf{x}, t)}{\partial t} + \mathcal{L}u(\mathbf{x}, t) = 0,$$

where $\mathbf{x} \in \mathcal{M}$ and the solution $u(\mathbf{x}, t)$ represents the temperature at time t at point $\mathbf{x}$.

Eigendecomposition of $\mathcal{L}$

Definition

A function $\varphi_k : \mathcal{M} \to \mathbb{R}$ is an eigenfunction of $\mathcal{L}$ if there exists $\lambda_k \in \mathbb{R}$ such that

$$\mathcal{L}\varphi_k = \lambda_k \varphi_k.$$

- There are countably many pairs (φ_k, λ_k).
- $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \dots$
- We can write any reasonable function $f(\mathbf{x})$ defined on $\mathcal{M}$ as

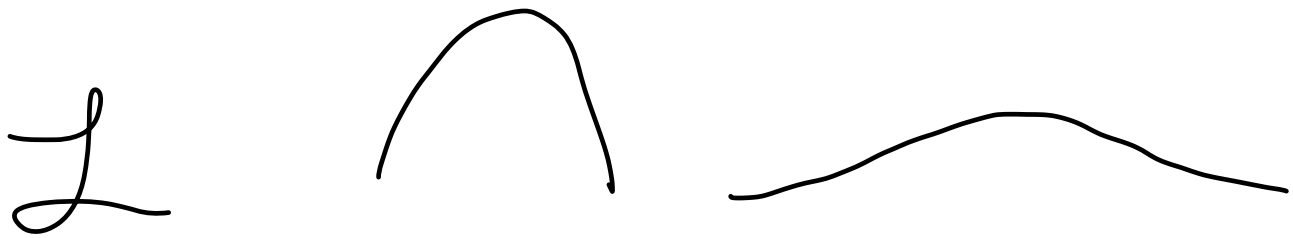
$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \hat{f}_k \varphi_k(\mathbf{x})$$

- We can recover the coefficients $\hat{f}_k \in \mathbb{R}$ through the relationship

$$\hat{f}_k = \langle f, \varphi_k \rangle_{\mathcal{M}}$$

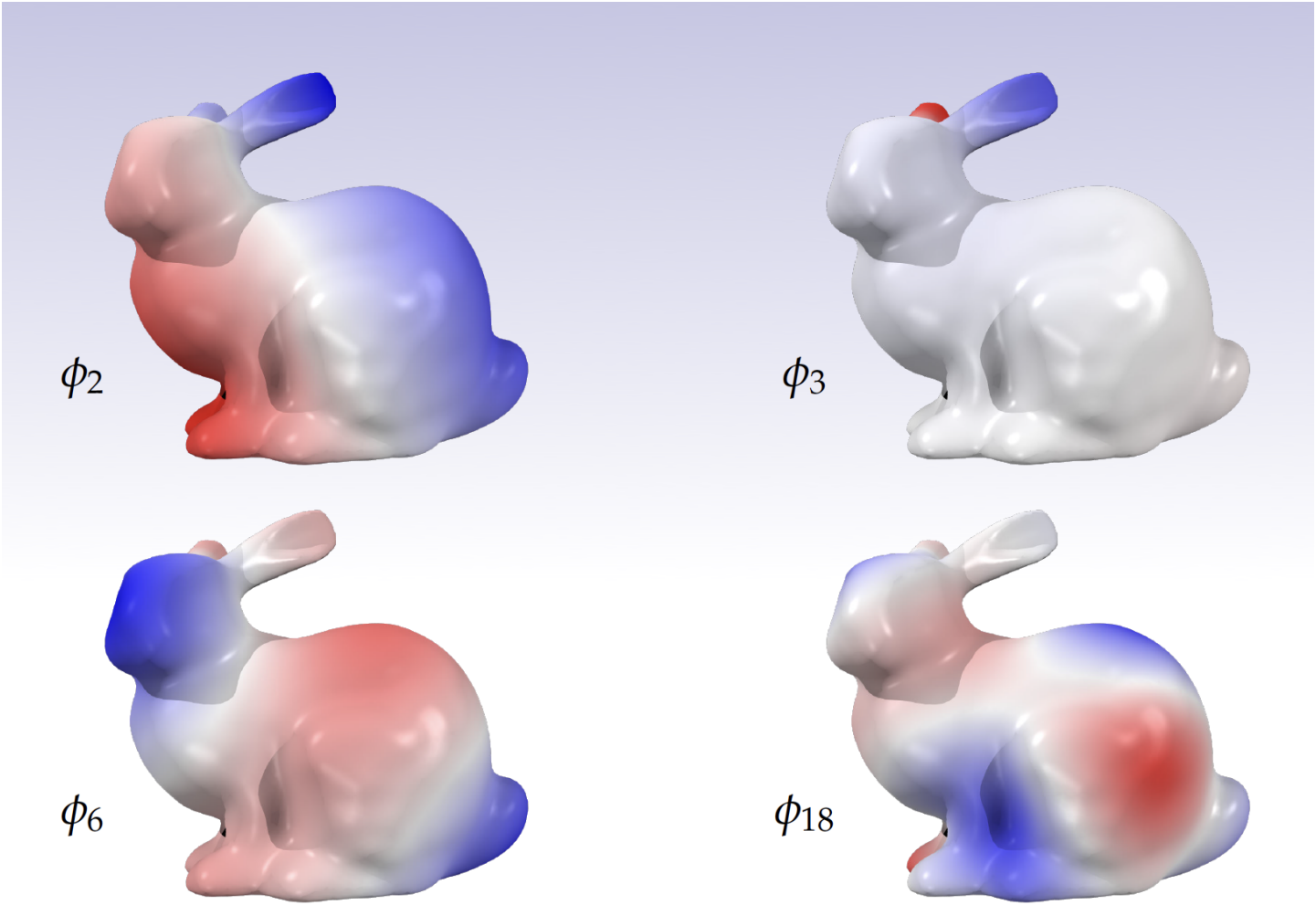
where $\langle f, g \rangle_{\mathcal{M}} = \int_{\mathcal{M}} f(x) g(x) d\mu(x)$

Intuition



- The Laplace-Beltrami operator tells us something about how a signal (function) diffuses over the manifold. Intuitively, this tells us something about the geometry of the manifold.
- The eigenfunctions of the Laplace-Beltrami operator are the "building blocks" of any function on the manifold.
- The first eigenfunctions capture general characteristics, and the later eigenfunctions capture fine details.

Example: Stanford bunny [4]



Convolution on $\mathcal{M}$

Using the φ_k, we can define a convolution operation on $\mathcal{M}$!

$$f \star h(\mathbf{x}) := \sum_k \hat{f}_k \hat{h}_k \varphi_k(\mathbf{x})$$

Why would we want to do this?

- In other settings (most famous example: images), convolutions produce very good embeddings.
- Convolution incorporates the geometry of the domain.
- We will see that these are relatively easy and cheap to compute.

Notation and big-picture goal

$(x, f(x))$

- We will work with data in the form $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^m$.
- To apply the manifold assumption, we'll assume that $\mathbf{x} \in \mathcal{M}$, where $\mathcal{M}$ is a d-dimensional manifold and $d < m$.
- Note that we don't know anything about $\mathcal{M}$
- Goal: Approximate $\mathcal{L}$ and $\{\varphi_k, \lambda_k\}_{k=0}^{\infty}$ and use these to construct a useful embedding via convolutions

Approximating $\mathcal{L}$

We start with a collection of points in $\mathbb{R}^m$, $\{\mathbf{x}_i\}_{i=1}^n$. Given a *bandwidth parameter* ε, we define a weight matrix W whose entries are given by

$$W_{i,j} = \frac{1}{\varepsilon^{d/2}} e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\varepsilon}$$

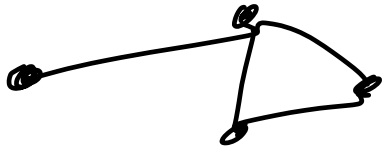
and a diagonal degree matrix D whose diagonal entries are given by

$$D_{i,i} = \sum_{j=1}^{n} W_{i,j}.$$

Then we can define the *graph Laplacian* by

$$L_n = \frac{1}{\varepsilon n}(D - W).$$

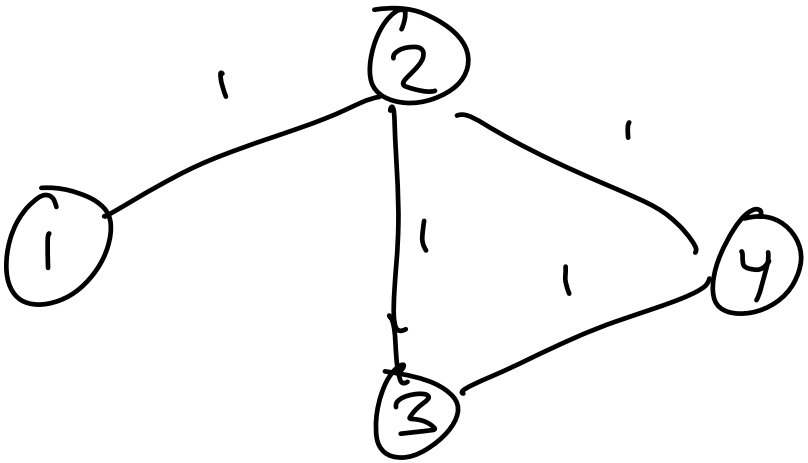
Interlude: Why graphs?



A *graph* (as a mathematical object) is a collection of *nodes* and *edges*.

- One way to study a graph is to write down its adjacency matrix A, where each entry $A_{i,j}$ encodes the weight of the edge between node i and node j.
- The adjacency matrix encodes the arrangement (geometry) of the graph.
- The graph Laplacian encodes how information "flows" throughout the graph.

Interlude: Why graphs?



$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

--- L_n $\rightarrow$ $n \times n$ matrix

Approximating φ_k and λ_k

It turns out that we can approximate φ_k and λ_k by computing the eigenvectors and eigenvalues of the matrix L_n!
Let $\mathbf{v}_k$ and $\tilde{\lambda}_k$ be the eigenvectors and eigenvalues of L_n (that is, $\mathbf{v}_k \in \mathbb{R}^n$ and $\tilde{\lambda}_k \in \mathbb{R}$ such that $L_n \mathbf{v}_k = \tilde{\lambda}_k \mathbf{v}_k$).

Theorem (Belkin and Niyoji, 2006 [1])

As $n \to \infty$, $[\mathbf{v}_k]_i \to \varphi_k(\mathbf{x}_i)$ and $\tilde{\lambda}_k \to \lambda_k$.

In other words, the ith entry of the eigenvector $\mathbf{v}_k$ is a good approximation for the eigenfunction φ_k evaluated at $\mathbf{x}_i$.

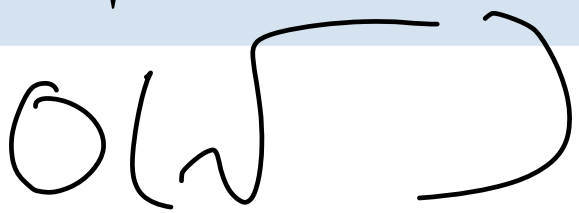
Approximating $\langle \cdot, \cdot \rangle_{\mathcal{M}}$

Recall that we need to be able to compute $\langle f, \varphi_k \rangle_{\mathcal{M}}$ in order to perform convolution. We can approximate this inner product (which involves integration over the manifold $\mathcal{M}$) with a weighted dot product.

Theorem (Chew et al, 2022 [3])

Assume f and g are continuous bounded functions on $\mathcal{M}$, and the points $\{\mathbf{x}_i\}_{i=1}^n$ are drawn uniformly at random from $\mathcal{M}$. Let f_i and g_i denote $f(\mathbf{x}_i)$ and $g(\mathbf{x}_i)$, respectively. Then, with high probability,

$$\left| \frac{1}{n} \sum_{i=1}^{n} f_i g_i - \langle f, g \rangle_{\mathcal{M}} \right| \leq C_{fg} \sqrt{\frac{18 \log(n)}{n}}.$$



Summary

- Start with a collection of points $\{\mathbf{x}_i\}_{i=1}^n$ and some measurement on those points $f(\mathbf{x}_i)$.
- Assume that the $\mathbf{x}_i$ are sampled from some d-dimensional manifold.
- Construct a data-driven graph Laplacian L_n (an $n \times n$ matrix).
- Compute the eigenvectors and eigenvalues of L_n.
- Use the eigenvectors to perform convolution.

$$\hat{f}_k = \langle f, \varphi_k \rangle_n$$

$$f \star h(\mathbf{x}) = \sum_k \hat{f}_k \hat{h}_k \varphi_k(\mathbf{x})$$

$$f \star h(\mathbf{x}_i) = \sum_{k=1}^n \hat{f}_k \hat{h}_k [\mathbf{v}_k]_i, \quad \hat{f}_k = \frac{1}{n} \sum_{i=1}^n f_i [\mathbf{v}_k]_i$$

$$\hat{H} = \begin{bmatrix} \hat{h}_1 & \hat{h}_2 & \cdots \end{bmatrix}$$

$$f \star h = \hat{H} V V^T f$$

The final embedding

Now that we can perform convolution, we can construct the final embedding.

1. Choose some functions h as your filters.
2. Perform convolution with your chosen filters h.
3. Apply a pointwise nonlinearity to the result (example: absolute value).
4. Repeat from step 2 as many times as you would like.

Some practical comments

- The problem has been reduced to linear algebra (eigenvector computation, matrix/vector multiplication)
- Computing eigenvectors can be expensive...
  - It turns out that you can just compute the first κ eigenvectors and eigenvalues of L_n, and you still obtain a decent approximation ([3, 2]).

Example: Single-Cell Data

Will a Melanoma Patient Respond to Immunotherapy?

- 54 Patients
- 11,862 cells per patient
- 30 proteins measured in each cell

Manifold classification task

- Each cell is a point in $\mathbb{R}^{30}$
- Each person is a point cloud of 11,862 points in $\mathbb{R}^{30}$
- We assume each person's points lie upon some d-dimensional manifold for $d < 30$.
- Simple classifier on embedding using approximation to the Laplace-Beltrami operator achieves 83% accuracy for patient outcomes

Takeaways

- It is useful to leverage geometric structure in data, and applying the "manifold assumption" is a trick that can be helpful with certain kinds of data.
- Convert a problem to linear algebra whenever possible!

Contact me!

- Email: joycechew@math.ucla.edu
- Website: joycechew.github.io
- Projects/research interests: Graph and manifold learning, manifold-valued matrix/tensor decompositions, topic modeling via nonnegative matrix factorizations, debiasing of word embeddings, differential privacy

References

[1] Mikhail Belkin and Partha Niyogi. "Convergence of Laplacian eigenmaps". In: *Advances in neural information processing systems* 19 (2006).

[2] Joyce Chew, Deanna Needell, and Michael Perlmutter. "A Convergence Rate for Manifold Neural Networks". In: *arXiv preprint arXiv:2212.12606* (2022).

[3] Joyce Chew et al. "Geometric scattering on measure spaces". In: *arXiv preprint arXiv:2208.08561* (2022).

[4] Justin Solomon, Keenana Crane, and Etienne Vouga. "Laplace-Beltrami: The Swiss Army Knife of Geometry Processing". SGP 2014 Tutorial.

Related Techniques

If your data is from a graph to begin with, you can construct a graph Laplacian using the graph's adjacency matrix.

Directed graph

A directed graph specifies the direction in which an edge "flows".

Examples:

- Communication
- Transit routes

Magnetic Laplacian on Directed Graphs

Hermitian Adjacency Matrix

$$A_s = \frac{1}{2}(A + A^T)$$

$$\Theta = \frac{\pi}{2}(A - A^T)$$

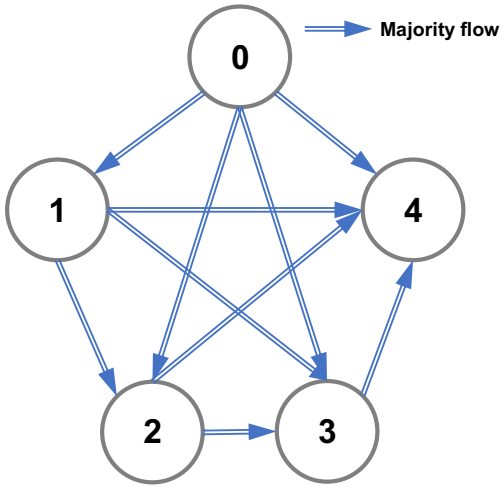
$$H = A_s \odot \exp(i\Theta)$$

The Magnetic Laplacian

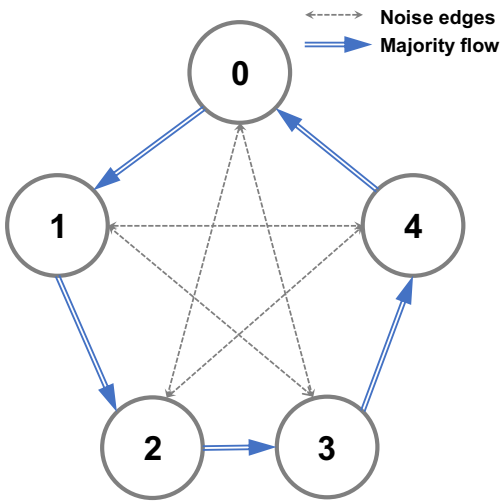
$$L = D_s - H = D_s - A_s \odot \exp(i\Theta)$$

- Undirected geometry is captured by the magnitude of entries.
- Directional information encoded by phase.

Example: Directed Stochastic Block Model



"Ordered" meta-graph



"Cyclic" meta-graph

- A node's cluster determines the probability of existence and direction of edges to nodes in other clusters.
- Task: identify which cluster a node belongs to
- Simple classifier on embedding obtained using magnetic Laplacian achieves accuracy of 97.8%, 99.8%, and 88.5%, respectively, for ordered, cyclic, and cyclic with noise