Data Analysis and Manifold Learning
Lecture 9: Diffusion on Manifolds and on Graphs

Radu Horaud
INRIA Grenoble Rhone-Alpes, France
Radu.Horaud@inrialpes.fr
http://perception.inrialpes.fr/
Outline of Lecture 9

- Kernels on graphs
- The exponential diffusion kernel
- The heat kernel of discrete manifolds
- Properties of the heat kernel
- The auto-diffusion function
- Shape matching
Material for this lecture

The Adjacency Matrix of a Graph (from Lecture #3)

For a graph with $n$ vertices and with binary edges, the entries of the $n \times n$ adjacency matrix are defined by:

$$A := \begin{cases} 
A_{ij} = 1 & \text{if there is an edge } e_{ij} \\
A_{ij} = 0 & \text{if there is no edge} \\
A_{ii} = 0 
\end{cases}$$

$$A = \begin{bmatrix} 
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 
\end{bmatrix}$$

Radu Horaud
Data Analysis and Manifold Learning; Lecture 9
The Walks of Length 2

- The number of walks of length 2 between two graph vertices:
  \[ N_2(v_i, v_j) = A^2(i, j) \]

- Examples:
  \[ N_2(v_2, v_3) = 1; \quad N_2(v_2, v_2) = 3 \]

\[
A^2 = \begin{bmatrix}
  2 & 1 & 1 & 1 \\
  1 & 3 & 1 & 0 \\
  1 & 1 & 2 & 1 \\
  1 & 0 & 1 & 1 \\
\end{bmatrix}
\]
The Number of Walks of Length $k$:

**Theorem**

The number of walks of length $k$ joining any two nodes $v_i$ and $v_j$ of a binary graph is given by the $(i, j)$ entry of the matrix $A^k$:

$$N_k(v_i, v_j) = A^k(i, j)$$

- A proof of this theorem is provided in: Godsil and Royle. (2001). Algebraic Graph Theory. Springer.
The Similarity Matrix of an Undirected Weighted Graph

- We consider \textit{undirected weighted graphs}; Each edge \( e_{ij} \) is weighted by \( \omega_{ij} > 0 \). We obtain:

\[
\Omega := \begin{cases} 
\omega(v_i, v_j) = \omega_{ij} & \text{if there is an edge } e_{ij} \\
0 & \text{if there is no edge} \\
0 & \text{if } i = j
\end{cases}
\]

- We will refer to this matrix as the \textit{base} similarity matrix
Walks of Length 2

- The weight of the walk from $v_i$ to $v_j$ through $v_l$ is defined as the product of the corresponding edge weights:

$$\omega(v_i, v_l) \omega(v_l, v_j)$$

- The sum of weights of all walks of length 2:

$$\omega_2(v_i, v_j) = \sum_{l=1}^{n} \omega(v_i, v_l) \omega(v_l, v_j) = \Omega^2(i, j)$$
Associated Feature Space

- Let $\Omega = \mathbf{U} \Lambda \mathbf{U}^\top$, i.e., the spectral decomposition of a real symmetric matrix.
- We have $\Omega^2 = \mathbf{U} \Lambda^2 \mathbf{U}^\top$ is symmetric semi-definite positive, hence it can be interpreted as a kernel matrix, with:

$$\omega_2(v_i, v_j) = \sum_{l=1}^n \omega(v_i, v_l) \omega(v_l, v_j)$$

$$= \langle (\omega(v_i, v_l))_{l=1}^n, (\omega(v_j, v_l))_{l=1}^n \rangle = \kappa(v_i, v_j)$$

- Associated feature space:

$$\phi : v_i \rightarrow \phi(v_i) = (\omega(v_i, v_1) \ldots \omega(v_i, v_l) \ldots \omega(v_i, v_n))^\top$$

- It is possible to “enhance” the base similarity matrix by linking vertices along walks of length 2.
Combining Powers of the Base Similarity Matrix

More generally one can take powers of the base similarity matrix as well as linear combinations of these matrices:

$$\alpha_1 \Omega + \ldots + \alpha_k \Omega^k + \ldots + \alpha_K \Omega^K$$

The eigenvalues of such a matrix must be nonnegative to satisfy the kernel condition:

$$\alpha_1 \Lambda + \ldots + \alpha_k \Lambda^k + \ldots + \alpha_K \Lambda^K \succeq 0$$
The Exponential Diffusion Kernel

- Consider the following combination of powers of the base similarity matrix:

\[ K = \sum_{k=0}^{\infty} \frac{\mu^k}{k!} \Omega^k \]

- This corresponds to:

\[ K = e^{\mu\Omega} \]

- where \( \mu \) is a decay factor chosen such that the influence of longer walks decreases, since they are less reliable.

- Spectral decomposition:

\[ K = U e^{\mu\Lambda} U^\top \]

- Hence \( K \) is a kernel matrix since its eigenvalues \( e^{\mu\lambda_i} \geq 0 \).
Heat Diffusion on a Graph

- The heat-diffusion equation on a Riemannian manifold:
  \[
  \left( \frac{\partial}{\partial t} + \Delta_M \right) f(x; t) = 0
  \]
- \( \Delta_M \) denotes the geometric Laplace-Beltrami operator.
- \( f(x; t) \) is the distribution of heat over the manifold at time \( t \).
- By extension, \( \frac{\partial}{\partial t} + \Delta_M \) can be referred to as the *heat operator* [Bérard et al. 1994].
- This equation can also be written on a graph:
  \[
  \left( \frac{\partial}{\partial t} + L \right) F(t) = 0
  \]
  where the vector \( F(t) = (F_1(t) \ldots F_n(t))^\top \) is indexed by the nodes of the graph.
The Fundamental Solution

- The fundamental solution of the (heat)-diffusion equation on Riemannian manifolds holds in the discrete case, i.e., for undirected weighted graphs.
- The solution in the discrete case is:

$$F(t) = H(t)f$$

- where $H$ denotes the discrete heat operator:

$$H(t) = e^{-tL}$$

- $f$ corresponds to the initial heat distribution:

$$F(0) = f$$

- Starting with a point-heat distribution at vertex $v_i$, e.g., $(0 \ldots f_i = 1 \ldots 0)^\top$, the distribution at $t$, i.e., $F(t) = (F_1(t) \ldots F_n(t))$ is given by the $i$-th column of the heat operator.
How to Compute the Heat Matrix?

- The exponential of a matrix:

\[ e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} \]

- Hence:

\[ e^{-tL} = \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} L^k \]

- It belongs to the “exponential diffusion” family of kernels just introduced with \( \mu = -t, t > 0 \).
Spectral Properties of $\mathbf{L}$

We start by recalling some basic facts about the combinatorial graph Laplacian:

- Symmetric semi-definite positive matrix: $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$
- Eigenvalues: $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$
- Eigenvectors: $\mathbf{u}_1 = \mathbb{1}, \mathbf{u}_2, \ldots, \mathbf{u}_n$
- $\lambda_2$ and $\mathbf{u}_2$ are the Fiedler value and the Fiedler vector
- $\mathbf{u}_i^\top \mathbf{u}_j = \delta_{ij}$
- $\mathbf{u}_{i>1}^\top \mathbb{1} = 0$
- $\sum_{i=1}^n u_{ik} = 0, \forall k \in \{2, \ldots, n\}$
- $-1 < u_{ik} < 1, \forall i \in \{1, \ldots, n\}, \forall k \in \{2, \ldots, n\}$

$$\mathbf{L} = \sum_{k=2}^n \lambda_k \mathbf{u}_k \mathbf{u}_k^\top$$
The Heat-kernel Matrix

\[ H(t) = e^{-tU\Lambda U^\top} = U e^{-t\Lambda} U^\top \succeq 0 \]

with:

\[ e^{-t\Lambda} = \text{Diag}[e^{-t\lambda_1} \ldots e^{-t\lambda_n}] \]

- Eigenvalues: \( 1 = e^{-t0} > e^{-t\lambda_2} \geq \ldots \geq e^{-t\lambda_n} \)
- Eigenvectors: same as the Laplacian matrix with their properties (previous slide).
- The heat trace (also referred to as the partition function):

\[ Z(t) = \text{tr}(H) = \sum_{k=1}^{n} e^{-t\lambda_k} \]

- The determinant:

\[ \det(H) = \prod_{k=1}^{n} e^{-t\lambda_k} = e^{-t\text{tr}(L)} = e^{-t\text{vol}(G)} \]
The Heat-kernel

- Computing the heat matrix:

\[ H(t) = \sum_{k=2}^{n} e^{-t\lambda_k} u_k u_k^\top \]

where we applied a deflation to get rid of the constant eigenvector: \( H \rightarrow H - u_1 u_1^\top \)

- The heat kernel (en entry of the matrix above):

\[ h(i, j; t) = \sum_{k=2}^{n} e^{-t\lambda_k} u_{ik} u_{jk} \]
Feature-space Embedding Using the Heat Kernel

\[ \mathbf{H}(t) = \left( \mathbf{U} e^{-\frac{1}{2}t\Lambda} \right) \left( \mathbf{U} e^{-\frac{1}{2}t\Lambda} \right)^\top \]

- Each row of the \( n \times n \) matrix \( \mathbf{U} e^{-t\Lambda/2} \) can be viewed as the coordinates of a graph vertex in a feature space, i.e., the mapping \( \phi : \mathcal{V} \rightarrow \mathbb{R}^{n-1}, \mathbf{x}_i = \phi(v_i) : \)

\[ \mathbf{x}_i = \begin{pmatrix} e^{-\frac{1}{2}t\lambda_2}u_{i2} & \ldots & e^{-\frac{1}{2}t\lambda_k}u_{ik} & \ldots & e^{-\frac{1}{2}t\lambda_n}u_{in} \end{pmatrix}^\top \]

\[ = (x_{i2} \ldots x_{ik} \ldots x_{in})^\top \]

- The heat-kernel computes the inner product in feature space:

\[ h(i, j; t) = \langle \phi(v_i), \phi(v_j) \rangle \]
The Auto-diffusion Function

- Each diagonal term of the heat matrix corresponds to the norm of a feature-space point:

\[
h(i, i; t) = \sum_{k=2}^{n} e^{-t\lambda_k} u_{ik}^2 = \|x_i\|^2
\]

- This is also known as the *auto-diffusion function* (ADF), or the amount of heat that remains at a vertex at time \( t \).

- The local maxima/minima of this function have been used for a feature-based scale-space representation of shapes.

- Associated shape descriptor: \( v_i \rightarrow h(i, i; t) \) hence it is a scalar function defined over the graph.
The ADF as a Shape Descriptor

$t = 0$

$t = 50$

$t = 500000$
Spectral Distances

- The heat distance:

\[ d_t^2(i, j) = h(i, i; t) + h(j, j; t) - 2h(i, j; t) = \sum_{k=2}^{n} \left( e^{-\frac{1}{2}t\lambda_k} (u_{ik} - u_{jk}) \right)^2 \]

- The commute-time distance:

\[ d_{CTD}^2(i, j) = \int_{t=0}^{\infty} \sum_{k=2}^{n} \left( e^{-\frac{1}{2}t\lambda_k} (u_{ik} - u_{jk}) \right)^2 dt = \sum_{k=2}^{n} \left( \lambda_k^{-1/2} (u_{ik} - u_{jk}) \right)^2 \]
Principal Component Analysis

- The covariance matrix in feature space:

\[
C_X = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^\top
\]

- With:

\[
X = \left( U e^{-\frac{1}{2} t \Lambda} \right)^\top = [x_1 \ldots x_i \ldots x_n]
\]

- Remember that each column of \( U \) sums to zero.

- \(-1 < -e^{-\frac{1}{2} t \lambda_k} < x_{ik} < e^{-\frac{1}{2} t \lambda_k} < 1, \forall 2 \leq k \leq n\)
Principal Component Analysis: The Mean

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$= \frac{1}{n} e^{-\frac{1}{2} t^T \Lambda} \left( \begin{array}{c} \sum_{i=1}^{n} u_{i2} \\ \vdots \\ \sum_{i=1}^{n} u_{in} \end{array} \right)$$

$$= \left( \begin{array}{c} 0 \\ \vdots \\ 0 \end{array} \right)$$
Principal Component Analysis: The Covariance

\[
C_X = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top
\]

\[
= \frac{1}{n} X X^\top
\]

\[
= \frac{1}{n} \left( U e^{-\frac{1}{2}t\Lambda} \right)^\top \left( U e^{-\frac{1}{2}t\Lambda} \right)
\]

\[
= \frac{1}{n} e^{-t\Lambda}
\]
Result I: The PCA of a Graph

- The eigenvectors (of the combinatorial Laplacian) are the principal components of the heat-kernel embedding: hence we obtain a maximum-variance embedding.
- The associated variances are $e^{-t\lambda_2}/n, \ldots, e^{-t\lambda_n}/n$.
- The embedded points are strictly contained in a hyper-parallelepipedon with volume $\prod_{i=2}^{n} e^{-t\lambda_i}$. 
Dimensionality Reduction (1)

- Dimensionality reduction consists in selecting the $K$ largest eigenvalues, $K < n$, conditioned by $t$, hence the criterion: choose $K$ and $t$, such that (scree diagram):

$$\alpha(K) = \frac{\sum_{i=2}^{K+1} e^{-t\lambda_i} / n}{\sum_{i=2}^{n} e^{-t\lambda_i} / n} \approx 0.95$$

- This is not practical because one needs to compute all the eigenvalues.
An alternative possibility is to use the determinant of the covariance matrix, and to choose the first $K$ eigenvectors such that (with $\alpha > 1$):

$$
\alpha(K) = \ln \frac{\prod_{i=2}^{K+1} e^{-t\lambda_i / n}}{\prod_{i=2}^{n} e^{-t\lambda_i / n}}
$$

which yields:

$$
\alpha(K) = t \left( \text{tr}(L) - \sum_{i=2}^{K+1} \lambda_i \right) + (n - K) \ln n
$$

This allows to choose $K$ for a scale $t$. 

Radu Horaud | Data Analysis and Manifold Learning; Lecture 9
Normalizing the Feature-space

Observe that the heat-kernels collapse to 0 at infinity: \( \lim_{t \to \infty} h(i, j; t) = 0 \). To prevent this problem, several normalizations are possible:

- Trace normalization
- Unit hyper-sphere normalization
- Time-invariant embedding
Trace Normalization

- Observe that \( \lim_{t \to \infty} h(i, j; t) = 0 \)
- Use the trace of the operator to normalize the embedding:

\[
\hat{x}_i = \frac{x_i}{\sqrt{Z(t)}}
\]

with: \( Z(t) \approx \sum_{k=2}^{K+1} e^{-t\lambda_k} \)
- the \( k \)-component of the \( i \)-coordinate writes:

\[
\hat{x}_{ik}(t) = \frac{(e^{-t\lambda_k}u_{ik}^2)^{1/2}}{\left(\sum_{l=2}^{K+1} e^{-t\lambda_l}\right)^{1/2}}
\]
- At the limit:

\[
\hat{x}_i(t \to \infty) = \left( \frac{u_{i2}}{\sqrt{m}} \quad \ldots \quad \frac{u_{im+1}}{\sqrt{m}} \quad 0 \quad \ldots \quad 0 \right)^\top
\]

where \( m \) is the multiplicity of the first non-null eigenvalue.
Unit Hyper-sphere Normalization

- The embedding lies on a unit hyper-sphere of dimension $K$:
  \[
  \tilde{x}_i = \frac{x_i}{\|x_i\|}
  \]

- The heat distance becomes a geodesic distance on a spherical manifold:
  \[
  d_S(i, j; t) = \arccos \tilde{x}_i^\top \tilde{x}_j = \arccos \left( \frac{h(i, j; t)}{(h(i, i; t) h(j, j; t))^{1/2}} \right)
  \]

- At the limit ($m$ is the multiplicity of the largest non-null eigenvalue):
  \[
  \tilde{x}_i(t \to \infty) = \left( \frac{u_{i2}}{(\sum_{l=2}^{m+1} u_{il}^2)^{1/2}} \cdots \frac{u_{im+1}}{(\sum_{l=2}^{m+1} u_{il}^2)^{1/2}} 0 \cdots 0 \right)^\top
  \]
Time-invariant Embedding

- Integration over time:

\[
\mathbf{L}^\dagger = \int_0^\infty \mathbf{H}(t) = \int_0^\infty \sum_{k=2}^n e^{-t\lambda_k} \mathbf{u}_k \mathbf{u}_k^\top \, dt \\
= \sum_{k=2}^n \frac{1}{\lambda_k} \mathbf{u}_k \mathbf{u}_k^\top = \mathbf{U} \Lambda^\dagger \mathbf{U}^\top
\]

- with: \( \Lambda^\dagger = \text{Diag}[\lambda_2^{-1}, \ldots, \lambda_n^{-1}] \).

- Matrix \( \mathbf{L}^\dagger \) is called the \textit{discrete Green’s function} [ChungYau2000], the Moore-Penrose pseudo-inverse of the Laplacian.

- Embedding: \( \mathbf{x}_i = \left( \lambda_2^{-1/2} \mathbf{u}_{i2} \ldots \lambda_{K+1}^{-1/2} \mathbf{u}_{iK+1} \right)^\top \)

- Covariance: \( \mathbf{C}_X = \frac{1}{n} \text{Diag}[\lambda_2^{-1}, \ldots, \lambda_{K+1}^{-1}] \)
Examples of Normalized Embeddings

$t = 50$

$t = 5000$

$t = 500000$
Shape Matching (1)

\[ t = 200, \quad t' = 201.5 \]

\[ t = 90, \quad t' = 1005 \]
Shape Matching (2)
Shape Matching (3)
Sparse Shape Matching

- Shape/graph matching is equivalent to matching the embedded representations [Mateus et al. 2008]
- Here we use the projection of the embeddings on a unit hyper-sphere of dimension $K$ and we apply rigid matching.
- How to select $t$ and $t'$, i.e., the scales associated with the two shapes to be matched?
- How to implement a robust matching method?
Let $C_X$ and $C_{X'}$ be the covariance matrices of two different embeddings $X$ and $X'$ with respectively $n$ and $n'$ points:

$$\det(C_X) = \det(C_{X'})$$

$\det(C_X)$ measures the volume in which the embedding $X$ lies. Hence, we impose that the two embeddings are contained in the same volume.

From this constraint we derive:

$$t' \text{ tr}(L') = t \text{ tr}(L) + K \log n/n'$$
Robust Matching

- Build an association graph.
- Search for the largest set of mutually compatible nodes (maximal clique finding).
- See [Sharma and Horaud 2010] (Nordia workshop) for more details.