Constructing Laplace Operators from Data

Mikhail Belkin

The Ohio State University,
Dept. of Computer Science and Engineering

Collaborators: Partha Niyogi, Jian Sun, Yusu Wang, Vikas Sindhwani
Machine Learning – high dimensional data (100-D).

Typical problems: clustering, classification, dimensionality reduction.
Machine Learning – high dimensional data (100-D).

Typical problems: clustering, classification, dimensionality reduction.

Graphics/CG – low-dimensional data (2-D).

Typical problems: identify, match and process surfaces.
Machine learning:

Probability Distribution ——— Data

Manifold ——— Graph
Machine learning:

- Probability Distribution ——— Data
- Manifold ——— Graph

Graphics/CG:

- Underlying Spatial Object ——— Data
- 2-D Surface ——— Mesh
Laplacian is a powerful geometric analyzer.
Plan of the talk

Laplacian is a powerful geometric analyzer. (As evidenced by Nobel prizes!)
Laplacian is a powerful geometric analyzer. (As evidenced by Nobel prizes!)

Want practical algorithms with theorems.
Laplacian is a powerful geometric analyzer. (As evidenced by Nobel prizes!)

Want practical algorithms with theorems.

Will discuss algorithms/theory for ML/CG.
Laplace-Beltrami operator

\[ \Delta_M f(p) = - \sum_i \frac{\partial^2 f(\exp_p(x))}{\partial x_i^2} \]

Generalization of Fourier analysis.
Algorithmic framework: Laplacian

Natural smoothness functional (analogue of $\text{grad}$):

$$S(f) = (f_1 - f_2)^2 + (f_1 - f_3)^2 + (f_2 - f_3)^2 + (f_3 - f_4)^2 + (f_4 - f_5)^2 + (f_4 - f_5)^2 + (f_5 - f_6)^2$$

Basic fact:

$$S(f) = \sum_{i \sim j} (f_i - f_j)^2 = \frac{1}{2} f^t L f$$
Algorithmic framework
Algorithmic framework
Algorithmic framework

\[ W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}} \]

\[ Lf(x_i) = f(x_i) \sum_j e^{-\frac{\|x_i - x_j\|^2}{t}} - \sum_j f(x_j) e^{-\frac{\|x_i - x_j\|^2}{t}} \]

\[ f^t Lf = 2 \sum_{i \sim j} e^{-\frac{\|x_i - x_j\|^2}{t}} (f_i - f_j)^2 \]
Data representation

\[ f : G \rightarrow \mathbb{R} \]

Minimize \[ \sum_{i \sim j} w_{ij} (f_i - f_j)^2 \]

Preserve adjacency.

Solution: \( Lf = \lambda f \) (slightly better \( Lf = \lambda Df \))

Lowest eigenfunctions of \( L \) (\( \tilde{L} \)).

Laplacian Eigenmaps

Belkin Niyogi 01

Related work: LLE: Roweis, Saul 00; Isomap: Tenenbaum, De Silva, Langford 00
Hessian Eigenmaps: Donoho, Grimes, 03; Diffusion Maps: Coifman, et al, 04
Laplacian Eigenmaps

- Visualizing spaces of digits and sounds.
  Partiview, Ndaona, Surendran 04

- Machine vision: inferring joint angles.
  Corazza, Andriacchi, Stanford Biomotion Lab, 05, Partiview, Surendran

Isometrically invariant representation.  [link]

- Reinforcement Learning: value function approximation.
  Mahadevan, Maggioni, 05
Semi-supervised learning

Learning from labeled and unlabeled data.

- Unlabeled data is everywhere. Need to use it.
- Natural learning is semi-supervised.
Semi-supervised learning

Learning from labeled and unlabeled data.

- Unlabeled data is everywhere. Need to use it.
- Natural learning is semi-supervised.

Idea:
construct the Laplace operator using unlabeled data.

Fit eigenfunctions using labeled data.
SVM

\[ \gamma_A = 0.03125 \quad \gamma_I = 0 \]
Toy example

\[ \gamma_A = 0.03125 \quad \gamma_I = 0 \]

SVM

\[ \gamma_A = 0.03125 \quad \gamma_I = 0.01 \]

Laplacian SVM

\[ \gamma_A = 0.03125 \quad \gamma_I = 1 \]

Laplacian SVM
### Experimental comparisons

<table>
<thead>
<tr>
<th>Dataset →</th>
<th>g50c</th>
<th>Coil20</th>
<th>Uspst</th>
<th>mac-win</th>
<th>WebKB (link)</th>
<th>WebKB (page)</th>
<th>WebKB (page+link)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (full labels)</td>
<td>3.82</td>
<td>0.0</td>
<td>3.35</td>
<td>2.32</td>
<td>6.3</td>
<td>6.5</td>
<td>1.0</td>
</tr>
<tr>
<td>SVM (l labels)</td>
<td>8.32</td>
<td>24.64</td>
<td>23.18</td>
<td>18.87</td>
<td>25.6</td>
<td>22.2</td>
<td>15.6</td>
</tr>
<tr>
<td>Graph-Reg</td>
<td>17.30</td>
<td>6.20</td>
<td>21.30</td>
<td>11.71</td>
<td>22.0</td>
<td>10.7</td>
<td>6.6</td>
</tr>
<tr>
<td>TSVM</td>
<td>6.87</td>
<td>26.26</td>
<td>26.46</td>
<td>7.44</td>
<td>14.5</td>
<td>8.6</td>
<td>7.8</td>
</tr>
<tr>
<td>Graph-density</td>
<td>8.32</td>
<td>6.43</td>
<td>16.92</td>
<td>10.48</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>▽TSVM</td>
<td>5.80</td>
<td>17.56</td>
<td>17.61</td>
<td>5.71</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LDS</td>
<td>5.62</td>
<td>4.86</td>
<td>15.79</td>
<td>5.13</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LapSVM</td>
<td>5.44</td>
<td>3.66</td>
<td>12.67</td>
<td>10.41</td>
<td>18.1</td>
<td>10.5</td>
<td>6.4</td>
</tr>
</tbody>
</table>
What is the connection between point-cloud Laplacian $L$ and Laplace-Beltrami operator $\Delta_M$?

Analysis of algorithms:

Eigenvectors of $L$ $\leftrightarrow$ Eigenfunctions of $\Delta_M$
**Theorem** [convergence of eigenfunctions]

\[ \text{Eig}[L^{t_n}_n] \rightarrow \text{Eig}[\Delta \mathcal{M}] \]

(Convergence in probability)

number of data points \( n \rightarrow \infty \)

width of the Gaussian \( t_n \rightarrow 0 \)

Previous work. Point-wise convergence.
Belkin, 03; Belkin, Niyogi 05,06; Lafon Coifman 04,06; Hein Audibert Luxburg, 05; Gine Kolchinskii, 05, Singer, 06

Convergence of eigenfunctions for a fixed \( t \):
Kolchniskii Gine 00, Luxburg Belkin Bousquet 04
Recall

Heat equation in $\mathbb{R}^n$:

$u(x, t)$ — heat distribution at time $t$.

$u(x, 0) = f(x)$ — initial distribution. $x \in \mathbb{R}^n, t \in \mathbb{R}$.

$$\Delta_{\mathbb{R}^n} u(x, t) = \frac{du}{dt}(x, t)$$

Solution — convolution with the heat kernel:

$$u(x, t) = (4\pi t)^{-\frac{n}{2}} \int_{\mathbb{R}^n} f(y) e^{-\frac{||x-y||^2}{4t}} dy$$
Proof idea (pointwise convergence)

Functional approximation:
Taking limit as $t \to 0$ and writing the derivative:

$$\Delta_{\mathbb{R}^n} f(x) = \frac{d}{dt} \left[ (4\pi t)^{-\frac{n}{2}} \int_{\mathbb{R}^n} f(y) e^{-\frac{\|x-y\|^2}{4t}} dy \right]_0$$
Proof idea (pointwise convergence)

Functional approximation:
Taking limit as $t \to 0$ and writing the derivative:

$$\Delta_{\mathbb{R}^n} f(x) = \frac{d}{dt} \left[ (4\pi t)^{-\frac{n}{2}} \int_{\mathbb{R}^n} f(y) e^{-\frac{\|x-y\|^2}{4t}} \, dy \right]_0$$

$$\Delta_{\mathbb{R}^n} f(x) \approx -\frac{1}{t} (4\pi t)^{-\frac{n}{2}} \left( f(x) - \int_{\mathbb{R}^n} f(y) e^{-\frac{\|x-y\|^2}{4t}} \, dy \right)$$
Proof idea (pointwise convergence)

Functional approximation:
Taking limit as $t \to 0$ and writing the derivative:

$$\Delta_{\mathbb{R}^n} f(x) = \frac{d}{dt} \left[ (4\pi t)^{-\frac{n}{2}} \int_{\mathbb{R}^n} f(y) e^{-\frac{\|x-y\|^2}{4t}} dy \right]_0$$

$$\Delta_{\mathbb{R}^n} f(x) \approx -\frac{1}{t} (4\pi t)^{-\frac{n}{2}} \left( f(x) - \int_{\mathbb{R}^n} f(y) e^{-\frac{\|x-y\|^2}{4t}} dy \right)$$

Empirical approximation:
Integral can be estimated from empirical data.

$$\Delta_{\mathbb{R}^n} f(x) \approx -\frac{1}{t} (4\pi t)^{-\frac{n}{2}} \left( f(x) - \sum_{x_i} f(x_i) e^{-\frac{\|x-x_i\|^2}{4t}} \right)$$
Some difficulties arise for manifolds:

- Do not know distances.
- Do not know the heat kernel.
Some difficulties arise for manifolds:

- Do not know distances.
- Do not know the heat kernel.

Careful analysis required.
Mesh Laplacian

Mesh $K$.

Triangle $t$. Area $A(t)$.

Vertices $v, w$.

$$L_K f(w) = \frac{1}{4\pi t^2} \sum_{t \in K} \frac{A(t)}{3} \sum_{v \in t} e^{-\frac{\|p-w\|^2}{4t}} (f(v) - f(w))$$

Belkin Sun Wang, 07
Mesh Laplacian

Mesh $K$.

Triangle $t$. Area $A(t)$.

Vertices $v, w$.

$$L_K f(w) = \frac{1}{4\pi t^2} \sum_{t \in K} \frac{A(t)}{3} \sum_{v \in t} e^{-\frac{\|p-w\|^2}{4t}} (f(v) - f(w))$$
Mesh Laplacian

Mesh $K$.

Triangle $t$. Area $A(t)$.

Vertices $v, w$.

\[
L_K f(w) = \frac{1}{4\pi t^2} \sum_{t \in K} \frac{A(t)}{3} \sum_{v \in t} e^{-\frac{||p-w||^2}{4t}} (f(v) - f(w))
\]
$K$ is a “nice” mesh (does not fold onto itself).

**Theorem:**

$$L_K f \to \Delta_M f$$

as mesh size $\epsilon$ (biggest triangle) tends to zero.

$$t = \epsilon^{\frac{1}{2.5+0.001}}.$$
$K$ is a “nice” mesh (does not fold onto itself).

**Theorem:**

$$L_K f \to \Delta \mathcal{M} f$$

as mesh size $\epsilon$ (biggest triangle) tends to zero.

$$t = \epsilon^{\frac{1}{2.5+0.001}}.$$
Existing work: several methods, including Desbrun, et al 99, Meyer, et al 02, Xu 04. Numerous applications (smoothing, quadrangulations, deformations, etc). None converge even in $\mathbb{R}^2$. 

None converge even in $\mathbb{R}^2$.

<table>
<thead>
<tr>
<th>$f = x^2$</th>
<th>500 pts</th>
<th>2000 pts</th>
<th>8000 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meyer</td>
<td>0.481</td>
<td>0.256</td>
<td>0.357</td>
</tr>
<tr>
<td>Xu</td>
<td>0.220</td>
<td>0.173</td>
<td>0.197</td>
</tr>
<tr>
<td>MLap</td>
<td>0.069</td>
<td>0.017</td>
<td>0.004</td>
</tr>
</tbody>
</table>
Laplacian carries extensive geometric information.
Conclusions

- Laplacian carries extensive geometric information.
- Theoretically grounded and practical methods.
Conclusions

- Laplacian carries extensive geometric information.
- Theoretically grounded and practical methods. (Yes, it is possible!)
Conclusions

- Laplacian carries extensive geometric information.
- Theoretically grounded and practical methods. (Yes, it is possible!)
- Interesting connections between machine learning and graphics.
Conclusions

- Laplacian carries extensive geometric information.
- Theoretically grounded and practical methods. *(Yes, it is possible!)*
- Interesting connections between machine learning and graphics.
- More things should be possible now.