Sampling from Determinantal Point Processes for Scalable Manifold Learning

Christian Wachinger1,2, Polina Golland1
1Computer Science and Artificial Intelligence Lab (CSAIL), MIT
2Massachusetts General Hospital, Harvard Medical School

Introduction
• Manifold learning is intractable for large datasets
• Approach: Select subset and use out-of-sample extension

Challenges:
• How to select a representative subset of points (landmarks)?
• How to retain the geometry of the space after sampling?

Contributions:
• Efficient, approximate sampling from determinantal point processes on non-Euclidean geometries
• Bhattacharyya distance for graph construction

Determinantal Point Process (DPP)
• Repulsive behavior between particles
• Global, negative correlations: diversity
• Given a positive semi-definite kernel K, sample subset J with probability:

\[ P_K(J) = \frac{\det(K_{j_i,j_j})}{\det(K+I)} \]

J \subseteq X: Subset (Landmarks)
K_{i,j} = \exp(||x_i - x_j||^2/2\sigma^2)

Efficient approximation of DPP sampling on manifolds
• Sampling from DPPs as computationally complex as spectral decomposition
• Efficient approximation with linear complexity

Algorithm 1 Efficient approximation of DPP sampling
Require: Point set X, subset cardinality k, nearest neighbors m
1. Initialize D = 1
2. for 1 to k do
3. Select i \in 1 \ldots n with probability \( p(i) \propto D_i \)
4. J \leftarrow J \cup i
5. Calculate \( \Delta_j = \|x_i - x_j\| \), \forall j \in 1 \ldots n
6. Set m nearest neighbors as neighborhood \( N_j \) based on \( \{\Delta_j\}_{j=1 \ldots n} \)
7. \( D_j = D_j + f(\Delta_j), \forall j \in N_j \)
8. Optional: Calculate covariance \( C_i \) in local neighborhood \( N_j \) around \( x_i \)
9. end for
10. return J and optionally \( \{C_i\}_{i \in J} \)

Robust landmark selection
• Nearest neighbor graph construction suffers from sparse sampling
• Retain local geometry of original points by estimating the covariance matrix
• Bhattacharyya distance between Gaussian distributions \( \mathcal{G}(x_i, C_i) \)

\[ B(G_i, G_j) = \frac{1}{8} (x_i - x_j)^T C^{-1} (x_i - x_j) + \frac{1}{2} \ln \left( \frac{|C|}{\sqrt{|C_i||C_j|}} \right) \]

Algorithm summary
1. Select landmarks with approximate DPP sampling
2. Construct neighborhood graph on landmarks with Bhattacharyya distance
3. Calculate low-dimensional embedding based on neighborhood graph
4. Embed non-landmark points with out-of-sample extension

Experiments
• Embedding of handwritten digits (MNIST, 70K) and patches from head-and-neck CT scans (150K)
• Classification accuracy (nearest neighbor) for performance evaluation (10 numbers; 4 structures: Left/right parotid, brainstem, background)
• Laplacian eigenmaps
• Summary over 20 repetitions
• 15% faster than optimized K-means

Matrix reconstruction error

<table>
<thead>
<tr>
<th>Sampling</th>
<th>25</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>70.4</td>
<td>8.0</td>
<td>4.8</td>
<td>2.8</td>
<td>1.7</td>
<td>0.73</td>
<td>0.44</td>
</tr>
<tr>
<td>K-means Uniform</td>
<td>28.1</td>
<td>3.8</td>
<td>2.3</td>
<td>1.4</td>
<td>0.8</td>
<td>0.40</td>
<td>0.24</td>
</tr>
<tr>
<td>K-means++ Seeding</td>
<td>50.1</td>
<td>5.8</td>
<td>3.0</td>
<td>1.7</td>
<td>1.0</td>
<td>0.68</td>
<td>0.35</td>
</tr>
<tr>
<td>Efficient DPP</td>
<td><strong>25.0</strong></td>
<td><strong>3.6</strong></td>
<td><strong>1.9</strong></td>
<td><strong>1.0</strong></td>
<td><strong>0.7</strong></td>
<td><strong>0.38</strong></td>
<td><strong>0.26</strong></td>
</tr>
</tbody>
</table>

Similarity to K-means++ seeding

<table>
<thead>
<tr>
<th>Sampling</th>
<th>25</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>70.4</td>
<td>8.0</td>
<td>4.8</td>
<td>2.8</td>
<td>1.7</td>
<td>0.73</td>
<td>0.44</td>
</tr>
<tr>
<td>K-means Uniform</td>
<td>28.1</td>
<td>3.8</td>
<td>2.3</td>
<td>1.4</td>
<td>0.8</td>
<td>0.40</td>
<td>0.24</td>
</tr>
<tr>
<td>K-means++ Seeding</td>
<td>50.1</td>
<td>5.8</td>
<td>3.0</td>
<td>1.7</td>
<td>1.0</td>
<td>0.68</td>
<td>0.35</td>
</tr>
<tr>
<td>Efficient DPP</td>
<td><strong>25.0</strong></td>
<td><strong>3.6</strong></td>
<td><strong>1.9</strong></td>
<td><strong>1.0</strong></td>
<td><strong>0.7</strong></td>
<td><strong>0.38</strong></td>
<td><strong>0.26</strong></td>
</tr>
</tbody>
</table>

Robustness and scalability

Efficient DPP approximation performs better than K-means++ on datasets with a large number of points.

Acknowledgement: This work was supported in part by the Humboldt foundation, the National Alliance for Medical Image Computing (US4-EB005149), the NeuroImaging Analysis Center (P41-EB015902) and the Wistron corporation.