Heuristics Based Isomap

Project Group

Student : Mengoli Emanuele
Student : Florido Poka Samir Fernando

Professor Liberti Leo
Course: INF580

March 29, 2024

Overview

Introduction

Project Background

- Investigates the role of the Isomap algorithm in dimensionality reduction.
- Focus: Solving Euclidean Distance Geometry Problems (EDGP).
- Builds upon the foundational work by Tenenbaum et al. [1] and Liberti & D'Ambrosio [2].

Research Objectives

- Aim: To propose six completion algorithms for Isomap applications to EDGP.
- Goal: Enable the transformation of high-dimensional data into insightful, lower-dimensional forms.

ISOMAP Algorithm: Part 1

Algorithm ISOMAP Algorithm (Part 1)

Require: A set of points $X \subset \mathbb{R}^n$

1: Compute all pairwise distances for X to obtain distance matrix D

2: Select a subset of distances to form a simple connected weighted graph $G = (V, E, d)$ where $V \in \mathbb{R}^n$, $d : E \to \mathbb{R}_+$

3: Compute all shortest paths in G to produce the approximate distance matrix $\widetilde{D}$, where $\widetilde{D}_{ij}$ equals the shortest path distance from i to j

The six graph completion algorithms correspond to a variant of step 3 of pseudocode 1, taken from [2].

ISOMAP Algorithm: Part 2

Algorithm ISOMAP Algorithm (Part 2)

Require: The approximate distance matrix $\widetilde{D}$ from Part 1

1: Derive the approximate Gram matrix $\widetilde{B} = -\frac{1}{2}J\widetilde{D}^2J$, where $J = I_n - \frac{1}{n}11^T$

2: Perform eigen decomposition on $\widetilde{B}$ to find Λ and P, such that $\widetilde{B} = P\Lambda P^T$

3: Replace any negative eigenvalues in Λ with zeros to obtain a positive semi-definite (PSD) matrix

4: If there are more than K positive eigenvalues, keep only the largest K ones to form $\widehat{\Lambda}$

5: Get the K-dimensional realization $X = P^T\sqrt{\widehat{\Lambda}}$

6: **return** the lower-dimensional representation $X \subset \mathbb{R}^K$

Floyd-Warshall Completion algorithm

Algorithm Floyd Warshall Algorithm

Require: $G = (V, E, d)$, $\tilde{D}$ initialized as $D \in \mathbb{R}^{n \times n}$ partial matrix
Ensure: completed distance matrix $\tilde{D} \in \mathbb{R}^{n \times n}$

for each $k \in V$ **do**
    for each $i \in V$ **do**
        for each $j \in V$ **do**
            if $i \neq k$ and $j \neq k$ and $i \neq j$ **then**
                if $\tilde{d}[i][j] > \tilde{d}[i][k] + \tilde{d}[k][j]$ **then**
                    $\tilde{d}[i][j] \leftarrow \tilde{d}[i][k] + \tilde{d}[k][j]$
                end if
            end if
        end for
    end for
end for
return $\tilde{D}$

Push and Pull

Another approach to compute $\tilde{D}$ is by solving the following Semi-Definite Program (SDP):

$$\max_x \sum_{\{u,v\}\in E} \|x_u - x_v\|^2 \quad s.t. \quad \|x_u - x_v\|^2 \leq d_{uv}^2 \quad \forall\{u,v\} \in E$$

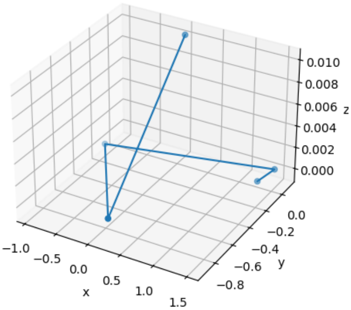

Figure: Isomap realization obtained with PaP with an inital graph of 5 vertices

Modified Breadth first search (BFS): Initialization

Algorithm Modified BFS, realization in $\mathbb{R}^K$ - Initialization

Require: $G(V, E, d)$

Ensure: Positions of nodes $P : V \to \mathbb{R}^K$, Z explored set

1: $r \leftarrow \underset{v \in V}{\operatorname{argmax}} \deg(v)$ ▷ Node with the highest degree
2: $P[r] \leftarrow 0$ ▷ Position r at origin in K-space
3: $Z \leftarrow \{r\}$
4: $Q \leftarrow$ FIFO queue initialized with r

Breadth first search: Execution

Algorithm Modified BFS - Execution

while $|Q| > 0$ **do**
    $v \leftarrow Q.\text{dequeue}()$
    $x_v \leftarrow P[v]$
    for each $w \in I_v$ **do** ▷ I_v is the set of neighbours of v
        if $w \notin Z$ **then**
            $Z \leftarrow Z \cup \{w\}$
            $x_w \leftarrow$ sample from $\mathbb{S}^{K-1}(x_v, d_{vw})$ ▷ surface of a K-dimensional sphere
            $P[w] \leftarrow x_w$
            $Q.\text{enqueue}(w)$
        end if
    end for
end while
$\tilde{D} \leftarrow$ pairwise Euclidean distances on P

Slack/Surplus Variables (SSV)

Another technique for optimizing the objective function involves introducing slack/surplus variables. This is achieved by minimizing the expression:

$$\min_{x,s} \sum_{\{u,v\}\in E} s_{uv}^2$$

$$s.t.$$

$$\|x_u - x_v\|^2 = d_{uv}^2 + s_{uv} \quad \forall\{u,v\} \in E$$

Centers tracker: Identifying Centers

Algorithm Centers tracker - Identifying Centers

Require: $G = (V, E, d)$
Ensure: Intermediate lists Γ and C

1: **Initialize** $\Gamma = [\ ]$ ▷ List to track vertices attained
2: **Initialize** $C = [\ ]$ ▷ List to store centers
3: **while** $|\Gamma| < |V|$ **do** ▷ Until all vertices are attained
4: $\quad r \leftarrow \underset{v \in V}{\operatorname{argmax}} \deg(v) \quad s.t \quad v \notin C$ ▷ Node with highest degree not in centers list
5: $\quad$ C $\leftarrow C \cup \{r\}$ ▷ Append center to centers list
6: $\quad$ **for** w $\in \{I_r \setminus \Gamma\}$ **do** ▷ For unattained neighbours (I_r) of the center
7: $\quad\quad \Gamma \leftarrow \Gamma \cup \{w\}$ ▷ Mark neighbour as attained
8: $\quad$ **end for**
9: **end while**

Centers tracker: Calculating Distances

Algorithm Centers tracker - Calculating Distances

Require: Lists Γ and C from Identifying Centers
Ensure: completed distance matrix $\tilde{D} \in \mathbb{R}^{n \times n}$

```
for all {i, j} ∈ C × C do
    if d_{i,j} is not known then
        d_{i,j} ← Dijkstra(i, j)        ▷ Calculate centers distance using Dijkstra's algorithm
    end if
end for
for a, b in V × V do
    d̃_{a,b} = d(a, center(a)) + d(center(a), center(b)) + d(center(b), b)
end for
return D̃
```

Random Walk - Introduction and Initialization

Algorithm Random walk Initialization

Require: $G = (V, E, d)$, its partial distance matrix $D \in \mathbb{R}^{n \times n}$
Ensure: A completed distance matrix $\tilde{D} \in \mathbb{R}^{n \times n}$
1: Initialize $C = [\ ]$ ▷ List to track visited vertices
2: Choose a random vertex u as the starting point
3: $C \leftarrow C \cup \{u\}$ ▷ Append starting vertex to *seen* list

Random Walk - Main Loop and Path Finding

Algorithm Random walk Path Finding

while $|C| < n$ **do**
    $v \leftarrow 0$
    $weight \leftarrow 0$
    $Q \leftarrow \{0, 1, \ldots, n-1\} \setminus \{u\}$ ▷ Set of all vertices excluding u
    for all $k \in Q$ **do**
        if $\tilde{D}[u, k] \neq 0$ **then**
            $Q \leftarrow Q \setminus \{k\}$ ▷ Remove reachable vertices from Q
            $v \leftarrow k$
        end if
    end for
    $weight \leftarrow \tilde{D}[u, v]$
    Initialize Z as an empty list ▷ *passed* list

Random Walk - Weight Update and Completion

Algorithm Random walk Completion

while $|Q| > 0$ **do**
  $v' \leftarrow v$
  for all $k \in Q$ **do**
    if $\tilde{D}[u, k] = 0$ & $\tilde{D}[v, k] \neq 0$ & $u \neq k$ & $k \notin Z$ **then**
      Update paths: $\tilde{D}[u, k]$, $\tilde{D}[k, u]$ via $weight + \tilde{D}[v, k]$
      $Q \leftarrow Q \setminus \{k\}$; $v \leftarrow k$; $weight \leftarrow \tilde{D}[u, k]$; $Z \leftarrow [\,]$
    end if
  end for
  if No update to v **then** ▷ Stuck at current vertex
    Find new v s.t. $\tilde{D}[u, v] \neq 0$, $v \notin Z$
    $weight \leftarrow \tilde{D}[u, v]$; $Z \leftarrow Z \cup \{v\}$
  end if
end while ▷ Also **end while** of `Path Finding`
$C \leftarrow C \cup \{u\}$; Choose new $u \notin C$ randomly
return $\tilde{D} = 0$

Experimental Results

Simulation Parameters:
$|V| = 15$, 50 simulations.

Type of Graph	Parameters Value
Erdős-Rényi	$p = 0.5$
Barabási–Albert	$\nu = 6$
Regular Graph	$d = 6$

Table: Graph Configuration Simulation Parameters

Metrics:

$$\mathrm{MDE}(x, G) = \frac{1}{|E|} \sum_{(i,j)\in E} \left| \|x_i - x_j\|_2 - d_{ij} \right| \tag{1}$$

$$\mathrm{LDE}(x, G) = \max_{(i,j)\in E} \left| \|x_i - x_j\|_2 - d_{ij} \right| \tag{2}$$

$$\mathrm{RMSD}(x, G) = \sqrt{\frac{1}{|E|} \sum_{(i,j)\in E} \left(\|x_i - x_j\|_2 - d_{ij} \right)^2} \tag{3}$$

Experimental Results

Mesure	Floyd Warshall	Centers tracker	Random path	BFS	Push and Pull	SSV
MDE	9.2964e-02	4.8227e-01	1.5096e-01	6.1542e-01	2.7027e-01	4.9461e-01
LDE	4.1406e-01	1.3468e+00	5.3254e-01	1.7650e+00	9.1143e-01	1.6366e+00
RMSD	1.3050e-01	5.8962e-01	1.9522e-01	7.4056e-01	3.6633e-01	6.3820e-01
CPU Time (s)	3.0566e-03	1.2310e-03	1.1783e-03	1.4362e-03	1.8954e+01	2.4819e+01

Table: Results obtained with the Erdos Renyi graph generator

Mesure	Floyd Warshall	Centers tracker	Random path	BFS	Push and Pull	SSV
MDE	7.9072e-02	2.8877e-01	1.3562e-01	5.4168e-01	2.6711e-01	4.5196e-01
LSE	3.6684e-01	9.7308e-01	4.8669e-01	1.6521e+00	9.0719e-01	1.5129e+00
RMSD	1.1121e-01	3.7104e-01	1.7710e-01	6.8466e-01	3.6470e-01	5.9447e-01
CPU Time (s)	2.8084e-03	1.0771e-03	1.1072e-03	1.4786e-03	1.8056e+01	2.6427e+01

Table: Results obtained with the Barabasi Albert graph generator

Experimental Results

Mesure	Floyd Warshall	Centers tracker	Random path	BFS	Push and Pull	SSV
MDE	3.9891e-01	1.1549e+00	3.2377e-01	6.5284e-01	3.6035e-01	5.7982e-01
LDE	1.1583e+00	3.1226e+00	8.6157e-01	1.8815e+00	9.4409e-01	1.6029e+00
RMSD	5.2543e-01	1.3895e+00	3.9362e-01	7.8684e-01	4.2411e-01	6.8544e-01
CPU Time (s)	3.0887e-03	1.1820e-03	1.1405e-03	1.5191e-03	2.1287e+01	2.4238e+01

Table: Results obtained with the random regular graph generator

Conclusion

Project Summary

- Developed six Isomap-based completion algorithms to address the Euclidean Distance Geometry Problem (EDGP).

Outcomes

- Our results emphasize the importance of considering both accuracy and computational efficiency when selecting heuristics for graph analysis.
- Tailored approaches for different graph topologies are necessary to achieve optimal performance.
- Further optimization and exploration of alternative algorithms may provide valuable avenues for improving performance in specific scenarios.

Q&A

References

J. B. Tenenbaum, V. d. Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

L. Liberti and C. d'Ambrosio, “The isomap algorithm in distance geometry,” in *16th International Symposium on Experimental Algorithms (SEA 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.