Localization III
Localization

• Local optimization:

• Global optimization:
  – multi-dimensional scaling.
  – Semi-definite programming.
Multi-dimensional scaling (MDS)

• Input:
  – A distance matrix P on n nodes.

• Output:
  – Embed nodes in \( \mathbb{R}^m \), s.t. their inter-distances approximate entries in P.

• Observations
  – If the distances are accurate, MDS recreates the configuration.
  – Also works when the distances are not Euclidean metric, then MDS recovers the “best fit”.
  – widely used in social sciences for visualization and similarity-based clustering.
MDS basics

• Measurement matrix $P$: $p_{ij}$.

• Embed into $\mathbb{R}^m$, $x_{ij}$

• Distance matrix $D$:

\[ d_{ij} = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2}. \]

• When $D = P$,

\[ \frac{1}{2} (p^2_{ij} - \frac{1}{n} \sum_{j=1}^{n} p^2_{ij} - \frac{1}{n} \sum_{i=1}^{n} p^2_{i,j} + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} p^2_{i,j}) \]

\[ = \sum_{k=1}^{m} x_{ik} x_{jk} \quad \text{You can verify this equality.} \]
MDS basics

• Now transfer P (shift to the center) to $B = XX^T$.

$$-\frac{1}{2} (p_{ij}^2 - \frac{1}{n} \sum_{j=1}^{n} p_{ij}^2) - \frac{1}{n} \sum_{i=1}^{n} p_{ij}^2 + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij}^2$$

$$= \sum_{k=1}^{m} x_{ik} x_{jk}$$

$$X = \begin{bmatrix}
x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\
x_{21} & x_{22} & x_{23} & \cdots & x_{2m} \\
x_{31} & x_{32} & x_{33} & \cdots & x_{3m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nm}
\end{bmatrix}$$

How to recover $X$ from $B$?
MDS basics

- First transfer $P$ to $B$, $\sum_{k=1}^{m} x_{ik} x_{jk}$
- $B$ is symmetric and positive semi-definite.
- Do eigen-decomposition on $B=VAV^T$.
- Now $X=VA^{1/2}$.
- $X$ is coordinates in dimension $n$.
- What if we want an embedding in $R^2$?
  - Take the largest 2 eigenvalue/eigenvectors.
The MDS algorithm

1. Compute all pairs shortest path lengths.
2. Apply MDS on the matrix P.
3. Retain the largest 2 eigenvalues and eigenvectors to find a 2D map.
Simulations

- Random placement
Simulations
Simulations

- Grid placement with 10% error.
MDS approach

• Experimentally most accurate in general.
• Centralized approach.
• Computationally expensive (can’t be executed at a sensor node).
• When the shortest path length is not a good approximation to the Euclidean distance, the result can be bad.
MDS approach

• When the shortest path length is not a good approximation to the Euclidean distance, the result can be bad.
Semidefinite programming
Linear Programming

minimize \( c^T x \)
subject to \( Ax + b \geq 0, \)
Linear Programming

- Geometric meaning: the constraints cut out a convex polytope $P$ in $\mathbb{R}^d$. Find the extremal point along direction $-c$. The solution is unique and is always realized at a vertex of $P$.

- Simplex method, interior point method.
Convex optimization

• In general, consider the constraints that form a convex domain $P$ in $\mathbb{R}^d$.

• Interior point method still works.
Semidefinite programming

• Relaxation of LP, a special case of convex optimization

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad F(x) \geq 0,
\end{align*}
\]

(1)

where

\[
F(x) \triangleq F_0 + \sum_{i=1}^{m} x_i F_i.
\]

• F’s are symmetric, positive semidefinite.

\[
z^T F(x) z \geq 0 \text{ for each } z \in \mathbb{R}^n.
\]
Graph realization problem

Given a graph $G = (V, E)$ and sets of non-negative weights, say $\{d_{ij} : (i, j) \in E\}$, the goal is to compute a realization of $G$ in the Euclidean space $\mathbb{R}^d$ for a given low dimension $d$, i.e.

- to place the vertices of $G$ in $\mathbb{R}^d$ such that
- the Euclidean distance between a pair of adjacent vertices $(i, j)$ equals to (or bounded by) the prescribed weight $d_{ij} \in E$. 
Sensor localization problem

Given anchors \( a_k \in \mathbb{R}^d \), \( \hat{d}_{kj} \in N_a \) and \( d_{ij} \in N_x \), find \( x_i \in \mathbb{R}^d \) such that

\[
\|x_i - x_j\|^2 = d_{ij}^2, \quad \forall (i, j) \in N_x, \quad i < j,
\]

\[
\|a_k - x_j\|^2 = \hat{d}_{kj}^2, \quad \forall (k, j) \in N_a,
\]

\((ij) ((kj))\) connects points \( x_i \) and \( x_j \) (\( a_k \) and \( x_j \)) with an edge whose Euclidean length is \( d_{ij} (\hat{d}_{kj}) \).

- not convex.
Matrix representation

Let \( X = [x_1 \ x_2 \ldots \ x_n] \) be the \( 2 \times n \) matrix that needs to be determined and \( e_j \) be the vector of all zero except 1 at the \( j \)th position. Then

\[
x_i - x_j = X(e_i - e_j) \quad \text{and} \quad a_k - x_j = [l \ X](a_k; -e_j)
\]

so that

\[
\|x_i - x_j\|^2 = (e_i - e_j)^T X^T X (e_i - e_j)
\]

\[
\|a_k - x_j\|^2 = (a_k; -e_j)^T [l \ X]^T [l \ X](a_k; -e_j) =
\]

\[
(a_k; -e_j)^T \begin{pmatrix} I & X \\ X^T & X^T X \end{pmatrix} (a_k; -e_j).
\]
More

Or, equivalently,

\[(e_i - e_j)^T Y (e_i - e_j) = d_{ij}^2, \quad \forall \ i, j \in N_x, \ i < j,\]

\[(a_k; -e_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (a_k; -e_j) = \hat{d}_{kj}^2, \quad \forall \ k, j \in N_a,\]

\[Y = X^T X.\]
More

Change

\[ Y = X^T X \]

to

\[ Y \succeq X^T X. \]

This matrix inequality is equivalent to

\[
\begin{pmatrix}
I & X \\
X^T & Y
\end{pmatrix} \succeq 0,
\]

This matrix has rank at least 2. If it’s 2, then \( Y = X^T X \), and the converse is also true.
Simulation results

(a) error:0.064, average trace:0.012, radio range=0.30.
(b) error:0.05, average trace:0.012, radio-range=0.40.
Conclusion

• Blackbox solution
• Error bound?
• There are more theoretical understanding of the performance in follow-up work.
  – Is the solution unique?
  – When is the solution exact?
  – New rigidity classes