Mini-Course on Dimensionality Reduction and Manifold Learning
Part 1: Linear Dimensionality Reduction

Keaton Hamm
University of Texas at Arlington

November 23, 2022

Modern data is high-dimensional, large scale

Modern data is high-dimensional, large scale

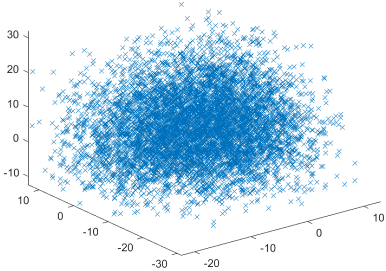
Leads to problems in analysis & inference (curse of dimensionality) and storage & processing

Modern data is high-dimensional, large scale

Leads to problems in analysis & inference (curse of dimensionality) and storage & processing

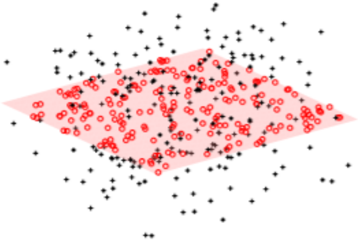
Can be helped by dimensionality reduction, feature selection and compression, quantization

High-dimensional data doesn't typically look like this



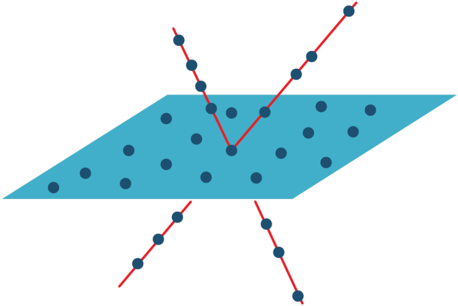
Structure in Data

It could look like this...



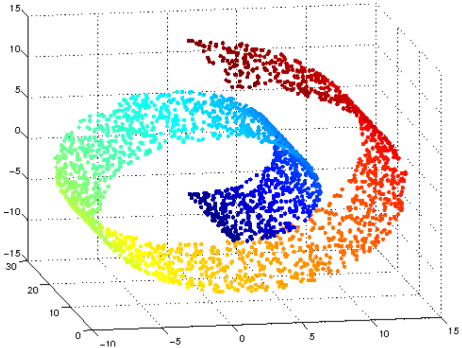
Structure in Data

or this ...



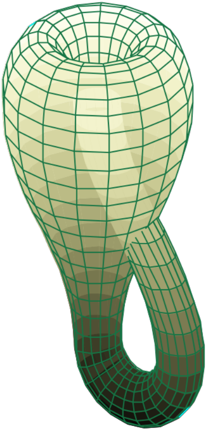
Structure in Data

or maybe this ...



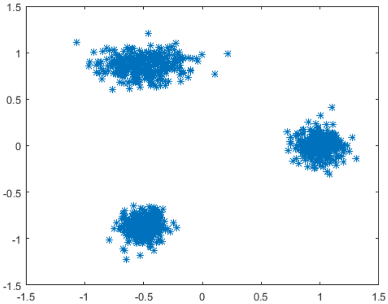
Structure in Data

or what about this?



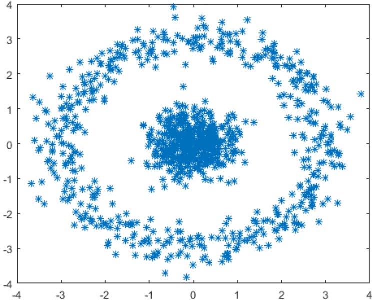
Structure in Data

Or it could be this ...



Structure in Data

or even this!



Structure in Data

The Point: High-dimensional data often exhibits some underlying structure, which may often be low-dimensional

Structure in Data

The Point: High-dimensional data often exhibits some underlying structure, which may often be low-dimensional

Question: So what does this mean for us?

Structure in Data

The Point: High-dimensional data often exhibits some underlying structure, which may often be low-dimensional

Question: So what does this mean for us?

Two Tasks:

Structure in Data

The Point: High-dimensional data often exhibits some underlying structure, which may often be low-dimensional

Question: So what does this mean for us?

Two Tasks:

- Detection (find when a given structure exists)

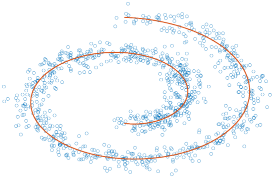
Structure in Data

The Point: High-dimensional data often exhibits some underlying structure, which may often be low-dimensional

Question: So what does this mean for us?

Two Tasks:

- Detection (find when a given structure exists)
- Learning (learn the features necessary to describe the structure – e.g., basis for subspace or charts/tangent spaces for manifold)



Common Structures:

- Subspaces
- Union of subspaces
- Sparsity
- Manifolds
- Union of manifolds
- Clusters (Communities)
- Structure + noise/outliers

Common Data Science/Machine Learning Tasks:

Common Data Science/Machine Learning Tasks:

- Feature Selection:

$$f(x_1, \ldots, x_N) \approx \tilde{f}(x_{i_1}, \ldots, x_{i_d}), \quad d \ll N$$

(e.g., weather modeling)

Common Data Science/Machine Learning Tasks:

- Feature Selection:

$$f(x_1, \dots, x_N) \approx \tilde{f}(x_{i_1}, \dots, x_{i_d}), \quad d \ll N$$

(e.g., weather modeling)

- Dimensionality Reduction:

$$\phi : \mathbb{R}^D \to \mathbb{R}^d, \quad d \ll D$$

(e.g., compression, visualization)

Common Data Science/Machine Learning Tasks:

- Feature Selection:

$$f(x_1,\ldots,x_N) \approx \tilde{f}(x_{i_1},\ldots,x_{i_d}), \quad d \ll N$$

  (e.g., weather modeling)

- Dimensionality Reduction:

$$\phi : \mathbb{R}^D \to \mathbb{R}^d, \quad d \ll D$$

  (e.g., compression, visualization)

- Manifold Learning (e.g., imaging)

$$\mathcal{M} \approx \bigcup S_i$$

Common Data Science/Machine Learning Tasks:

- Feature Selection:

$$f(x_1,\dots,x_N) \approx \tilde{f}(x_{i_1},\dots,x_{i_d}), \quad d \ll N$$

(e.g., weather modeling)

- Dimensionality Reduction:

$$\phi : \mathbb{R}^D \to \mathbb{R}^d, \quad d \ll D$$

(e.g., compression, visualization)

- Manifold Learning (e.g., imaging)

$$\mathcal{M} \approx \bigcup S_i$$

- Clustering/Community Detection

$$X = \{x_1,\dots x_N\} = C_1 \sqcup \cdots \sqcup C_k$$

(e.g., hand-written digits, facial recognition)

Common Data Science/Machine Learning Tasks:

- Feature Selection:

$$f(x_1, \dots, x_N) \approx \tilde{f}(x_{i_1}, \dots, x_{i_d}), \quad d \ll N$$

  (e.g., weather modeling)

- Dimensionality Reduction:

$$\phi : \mathbb{R}^D \to \mathbb{R}^d, \quad d \ll D$$

  (e.g., compression, visualization)

- Manifold Learning (e.g., imaging)

$$\mathcal{M} \approx \bigcup S_i$$

- Clustering/Community Detection

$$X = \{x_1, \dots x_N\} = C_1 \sqcup \cdots \sqcup C_k$$

  (e.g., hand-written digits, facial recognition)

- Classification

$$(x_i, y_i) \Rightarrow \mathcal{C}_\Theta : \mathbb{R}^D \to \{1, \dots, L\}$$

  (e.g., hand-written digits, facial recognition with labelled training

Structure in Data

Techniques:

- Graphs
- (Randomized) Linear Algebra
- Harmonic Analysis
- Statistics
- Optimization
- Neural Networks
- Topological invariants
- . . .

Dimensionality Reduction

Given: $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ with some known/expected structure
Goal: Find “nice" $\phi : \mathbb{R}^D \to \mathbb{R}^d$

Dimensionality Reduction

Given: $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ with some known/expected structure
Goal: Find “nice" $\phi : \mathbb{R}^D \to \mathbb{R}^d$

How do we choose ϕ?

Dimensionality Reduction

Given: $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ with some known/expected structure
Goal: Find “nice" $\phi : \mathbb{R}^D \to \mathbb{R}^d$

How do we choose ϕ?
Task dependent

Dimensionality Reduction

Given: $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ with some known/expected structure
Goal: Find “nice" $\phi : \mathbb{R}^D \to \mathbb{R}^d$

How do we choose ϕ?
Task dependent

- Preserve cluster structure / make separation easier

Dimensionality Reduction

Given: $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ with some known/expected structure
Goal: Find “nice" $\phi : \mathbb{R}^D \to \mathbb{R}^d$

How do we choose ϕ?
Task dependent

- Preserve cluster structure / make separation easier
- Preserve distance/metric / geometric structure

Dimensionality Reduction

Given: $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ with some known/expected structure
Goal: Find “nice" $\phi : \mathbb{R}^D \to \mathbb{R}^d$

How do we choose ϕ?
Task dependent

- Preserve cluster structure / make separation easier
- Preserve distance/metric / geometric structure
- Preserve linear algebraic structure

Dimensionality Reduction

Given: $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ with some known/expected structure
Goal: Find “nice" $\phi : \mathbb{R}^D \to \mathbb{R}^d$

How do we choose ϕ?
Task dependent

- Preserve cluster structure / make separation easier
- Preserve distance/metric / geometric structure
- Preserve linear algebraic structure
- Find a reduced basis

Dimensionality Reduction

Given: $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^D$ with some known/expected structure
Goal: Find “nice" $\phi : \mathbb{R}^D \to \mathbb{R}^d$

How do we choose ϕ?
Task dependent

- Preserve cluster structure / make separation easier
- Preserve distance/metric / geometric structure
- Preserve linear algebraic structure
- Find a reduced basis
- Fine low-dimensional parametrization

Notations

- $A \in \mathbb{R}^{m \times n}$ – data matrix (columns are data)

Notations

- $A \in \mathbb{R}^{m \times n}$ – data matrix (columns are data)
- For $x \in \mathbb{R}^n$, $|x|$ will be its Euclidean norm

Notations

- $A \in \mathbb{R}^{m \times n}$ – data matrix (columns are data)
- For $x \in \mathbb{R}^n$, $|x|$ will be its Euclidean norm
- $\|A\|_2 := \sup_{x \in \mathbb{R}^n, |x|=1} |Ax|$ (spectral norm)

Notations

- $A \in \mathbb{R}^{m\times n}$ – data matrix (columns are data)
- For $x \in \mathbb{R}^n$, $|x|$ will be its Euclidean norm
- $\|A\|_2 := \sup_{x\in\mathbb{R}^n, |x|=1} |Ax|$ (spectral norm)
- $\|A\|_F := \left(\sum_{i,j=1}^n |A_{ij}|^2\right)^{\frac{1}{2}}$ (Frobenius norm)

Notations

- $A \in \mathbb{R}^{m\times n}$ – data matrix (columns are data)
- For $x \in \mathbb{R}^n$, $|x|$ will be its Euclidean norm
- $\|A\|_2 := \sup_{x\in\mathbb{R}^n, |x|=1} |Ax|$ (spectral norm)
- $\|A\|_F := \left(\sum_{i,j=1}^n |A_{ij}|^2\right)^{\frac{1}{2}}$ (Frobenius norm)
- $A(i,:)$ is the i–th column of A and $A(:,j)$ is its j–th row. $A(I,J)$ is a submatrix of entries $(i,j) \in I \times J$

Background: Singular Value Decomposition

Every $A \in \mathbb{R}^{m\times n}$ has a SVD of the form

$$A = U\Sigma V^T = \begin{bmatrix} | & & | \\ u_1 & \dots & u_m \\ | & & | \end{bmatrix} \Sigma \begin{bmatrix} — & v_1 & — \\ & \vdots & \\ — & v_n & — \end{bmatrix}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are *orthogonal* ($U^{-1} = U^T$) and

$$\Sigma = \begin{bmatrix} \mathrm{diag}(\sigma_1, \dots, \sigma_m) & \mathbf{0} \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} \mathrm{diag}(\sigma_1, \dots, \sigma_n) \\ \mathbf{0} \end{bmatrix}$$

if $m < n$ or $m > n$, respectively.

Background: Singular Value Decomposition

Every $A \in \mathbb{R}^{m\times n}$ has a SVD of the form

$$A = U\Sigma V^T = \begin{bmatrix} | & & | \\ u_1 & \dots & u_m \\ | & & | \end{bmatrix} \Sigma \begin{bmatrix} — & v_1 & — \\ & \vdots & \\ — & v_n & — \end{bmatrix}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are *orthogonal* ($U^{-1} = U^T$) and

$$\Sigma = \begin{bmatrix} \mathrm{diag}(\sigma_1, \dots, \sigma_m) & \mathbf{0} \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} \mathrm{diag}(\sigma_1, \dots, \sigma_n) \\ \mathbf{0} \end{bmatrix}$$

if $m < n$ or $m > n$, respectively.

$\sigma_i^2 = \lambda_i(AA^T) = \lambda_i(A^TA)$, and are ordered $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\mathrm{rank}(A)} \geq 0$.

SVD Properties

Can be rewritten:

$$A = \sum_{i=1}^{\min\{m,n\}} \sigma_i u_i v_i^T$$

If $\text{rank}(A) = k < m, n$, then we also have

$$A = U_k \Sigma_k V_k^T = \begin{bmatrix} | & & | \\ u_1 & \dots & u_k \\ | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} — & v_1 & — \\ & \vdots & \\ — & v_k & — \end{bmatrix}$$

$$= \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

SVD Properties

Theorem (Eckart–Young–Mirsky)

Let $k \in \mathbb{N}$ and $A \in \mathbb{R}^{m\times n}$. Then $A_k := U_k \Sigma_k V_k^T$ is a solution of

$$\min_{B:\text{rank}(B)\leq k} \|A - B\|_2$$

(Also true for any Schatten p–norm including Frobenius norm)

SVD Properties

Theorem (Eckart–Young–Mirsky)

Let $k \in \mathbb{N}$ and $A \in \mathbb{R}^{m \times n}$. Then $A_k := U_k \Sigma_k V_k^T$ is a solution of

$$\min_{B:\text{rank}(B) \leq k} \|A - B\|_2$$

(Also true for any Schatten p–norm including Frobenius norm)

Note: $\|A\|_2 = \sigma_1(A)$, and $\|A\|_F = \left(\sum_{i=1}^{\text{rank}(A)} \sigma_i(A)^2\right)^{\frac{1}{2}}$

If 2 is replaced by $p \in [1, \infty]$ these are the family of Schatten p–norms. Unfortunately, $\|\cdot\|_2$ is the Schatten ∞–norm and Frobenius norm is the Schatten 2–norm

PCA

Given: data matrix $A \in \mathbb{R}^{m \times n}$ (columns are data points)

PCA

Given: data matrix $A \in \mathbb{R}^{m\times n}$ (columns are data points)
Step 1 – Centering:

$$\widehat{A}_{ij} := A_{ij} - u_i, \qquad \text{where} \qquad u_i := \frac{1}{n}\sum_{j=1}^{n} A_{ij}.$$

PCA

Given: data matrix $A \in \mathbb{R}^{m \times n}$ (columns are data points)

Step 1 – Centering:

$$\widehat{A}_{ij} := A_{ij} - u_i, \qquad \text{where} \qquad u_i := \frac{1}{n}\sum_{j=1}^{n} A_{ij}.$$

Step 2 – Compute the Covariance Matrix:

$$S_{ij} := \frac{1}{n-1}(\widehat{A}\widehat{A}^T)_{ij} = \frac{1}{n-1}\left\langle \widehat{A}_{i:}, \widehat{A}_{j:} \right\rangle = \frac{1}{n-1}\sum_{k=1}^{n}(A_{ik} - u_i)(A_{jk} - u_j)$$

$$=: \mathrm{Covar}(A_{i:}, A_{j:}).$$

PCA

Given: data matrix $A \in \mathbb{R}^{m\times n}$ (columns are data points)
Step 1 – Centering:

$$\widehat{A}_{ij} := A_{ij} - u_i, \qquad \text{where} \qquad u_i := \frac{1}{n}\sum_{j=1}^{n} A_{ij}.$$

Step 2 – Compute the Covariance Matrix:

$$S_{ij} := \frac{1}{n-1}(\widehat{A}\widehat{A}^T)_{ij} = \frac{1}{n-1}\left\langle \widehat{A}_{i:}, \widehat{A}_{j:}\right\rangle = \frac{1}{n-1}\sum_{k=1}^{n}(A_{ik}-u_i)(A_{jk}-u_j)$$
$$=: \mathrm{Covar}(A_{i:}, A_{j:}).$$

Note: $S_{ii} = \frac{1}{n-1}\sum_{k=1}^{n}(A_{ik}-\mu_i)^2 = \mathrm{Var}(A_{i:})$

Step 3 – Compute the spectral decomposition of S: $S = U\Lambda U^T$

PCA

Step 3 – Compute the spectral decomposition of S: $S = U \Lambda U^T$

Columns of U are called the Principal Components of $\widehat{A}$

PCA

Step 3 – Compute the spectral decomposition of S: $S = U\Lambda U^T$

Columns of U are called the Principal Components of $\widehat{A}$

Columns of U are an orthonormal basis for Col(A), and directions correspond to decreasing directions of variance in the data

Step 3 – Compute the spectral decomposition of S: $S = U\Lambda U^T$

Columns of U are called the Principal Components of $\widehat{A}$

Columns of U are an orthonormal basis for Col(A), and directions correspond to decreasing directions of variance in the data

Note: Typically, we ask only for the first few principal components;

Step 3 – Compute the spectral decomposition of S: $S = U\Lambda U^T$

Columns of U are called the Principal Components of $\widehat{A}$

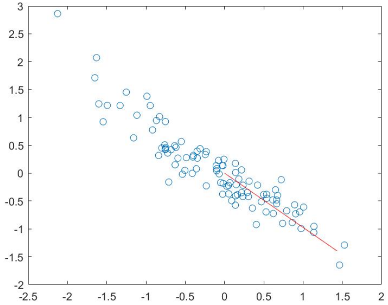
Columns of U are an orthonormal basis for Col(A), and directions correspond to decreasing directions of variance in the data

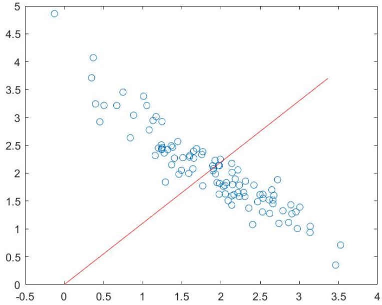
Note: Typically, we ask only for the first few principal components; Eckhart–Young–Mirsky implies that $U_k\Lambda_k U_k^T$ is the best rank k approximation of S

Computational Note: U are the left singular vectors of $\frac{1}{\sqrt{n-1}}\widehat{A}$, so instead of forming the covariance matrix S, we simply take the SVD of $\frac{1}{\sqrt{n-1}}\widehat{A}$

Further Reading: `https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf`

Effect of Centering





PCA on centered data (left) and the same uncentered data (right)

Applications of PCA

Dimensionality Reduction: Given $A \in \mathbb{R}^{m \times n}$, PCA gives us a map from $\mathbb{R}^m \to \mathbb{R}^k$

Applications of PCA

Dimensionality Reduction: Given $A \in \mathbb{R}^{m\times n}$, PCA gives us a map from $\mathbb{R}^m \to \mathbb{R}^k$

$$U_k^T \widehat{A} \in \mathbb{R}^{k\times n}$$

embedding data into a lower dimensional space ($\mathbb{R}^k$)

Applications of PCA

Dimensionality Reduction: Given $A \in \mathbb{R}^{m\times n}$, PCA gives us a map from $\mathbb{R}^m \to \mathbb{R}^k$

$$U_k^T \widehat{A} \in \mathbb{R}^{k\times n}$$

embedding data into a lower dimensional space ($\mathbb{R}^k$)
Why?

Dimensionality Reduction: Given $A \in \mathbb{R}^{m \times n}$, PCA gives us a map from $\mathbb{R}^m \to \mathbb{R}^k$

$$U_k^T \widehat{A} \in \mathbb{R}^{k \times n}$$

embedding data into a lower dimensional space ($\mathbb{R}^k$)
Why?

- Storage (data compression)

Applications of PCA

Dimensionality Reduction: Given $A \in \mathbb{R}^{m \times n}$, PCA gives us a map from $\mathbb{R}^m \to \mathbb{R}^k$

$$U_k^T \widehat{A} \in \mathbb{R}^{k \times n}$$

embedding data into a lower dimensional space ($\mathbb{R}^k$)

Why?

- Storage (data compression)
- Identify low-dimensional patterns in data

Applications of PCA

Dimensionality Reduction: Given $A \in \mathbb{R}^{m\times n}$, PCA gives us a map from $\mathbb{R}^m \to \mathbb{R}^k$

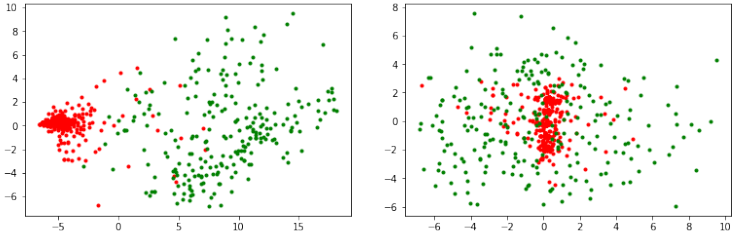
$$U_k^T \widehat{A} \in \mathbb{R}^{k\times n}$$

embedding data into a lower dimensional space ($\mathbb{R}^k$)

Why?

- Storage (data compression)
- Identify low-dimensional patterns in data
- Visualization ($k = 2, 3$)

Example



Wisconsin Breast Cancer Dataset `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`
$A \in \mathbb{R}^{32\times 569}$
(Left) Projection onto first two principal components; (Right) Projection onto first and third principal components. Red points are malignant, Green are benign

Interpretability

PCA can fail to give interpretable results.

Interpretability

PCA can fail to give interpretable results.

Example: $A \in \mathbb{R}^{m \times n}$ consists of m gene expression levels for n patients. PCA represents A in terms of singular vectors, which are linear combinations of genes (the canonical basis vectors)

Interpretability

PCA can fail to give interpretable results.

Example: $A \in \mathbb{R}^{m \times n}$ consists of m gene expression levels for n patients. PCA represents A in terms of singular vectors, which are linear combinations of genes (the canonical basis vectors)

a.k.a. What is an *eigengene*? [M. Mahoney and P. Drineas, *CUR Matrix Decompositions for Improved Data Analysis*, Proceedings of the National Academy of Science, 2009]

Interpretability

PCA can fail to give interpretable results.

Example: $A \in \mathbb{R}^{m \times n}$ consists of m gene expression levels for n patients. PCA represents A in terms of singular vectors, which are linear combinations of genes (the canonical basis vectors)

a.k.a. What is an *eigengene*? [M. Mahoney and P. Drineas, *CUR Matrix Decompositions for Improved Data Analysis*, Proceedings of the National Academy of Science, 2009]

Maybe we should look further for a basis for A that is interpretable

Column Subset Selection: Background

Given $A \in \mathbb{R}^{m\times n}$, its Moore–Penrose pseudoinverse is the unique matrix $A^\dagger \in \mathbb{R}^{n\times n}$ such that

- $AA^\dagger A = A$
- $A^\dagger AA^\dagger = A^\dagger$
- $AA^\dagger$ is symmetric
- $A^\dagger A$ is symmetric

Column Subset Selection: Background

Given $A \in \mathbb{R}^{m \times n}$, its Moore–Penrose pseudoinverse is the unique matrix $A^\dagger \in \mathbb{R}^{n \times n}$ such that

- $AA^\dagger A = A$
- $A^\dagger AA^\dagger = A^\dagger$
- $AA^\dagger$ is symmetric
- $A^\dagger A$ is symmetric

If $A = U\Sigma V^T$, then

$$A^\dagger = V\Sigma^\dagger U^T$$

where $\Sigma^\dagger$ has entries $\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_{\text{rank}(A)}}$ along its diagonal.

Column Subset Selection: Background

Given $A \in \mathbb{R}^{m\times n}$, its Moore–Penrose pseudoinverse is the unique matrix $A^\dagger \in \mathbb{R}^{n\times n}$ such that

- $AA^\dagger A = A$
- $A^\dagger AA^\dagger = A^\dagger$
- $AA^\dagger$ is symmetric
- $A^\dagger A$ is symmetric

If $A = U\Sigma V^T$, then

$$A^\dagger = V\Sigma^\dagger U^T$$

where $\Sigma^\dagger$ has entries $\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_{\text{rank}(A)}}$ along its diagonal.

$AA^\dagger : \mathbb{R}^m \to \mathbb{R}^m$ is the orthogonal projection onto the Col(A)

$A^\dagger A : \mathbb{R}^n \to \mathbb{R}^n$ is the orthogonal projection onto Row(A)

Column Subset Selection

Column Subset Selection Problem (CSSP)

Given $A \in \mathbb{R}^{m\times n}$ and $k < n$, find the column submatrix $C = [a_{i_1} \dots a_{i_k}]$ which minimizes

$$\|A - CC^{\dagger}A\|_2$$

Column Subset Selection

Column Subset Selection Problem (CSSP)

Given $A \in \mathbb{R}^{m \times n}$ and $k < n$, find the column submatrix $C = [a_{i_1} \ldots a_{i_k}]$ which minimizes

$$\|A - CC^{\dagger}A\|_2$$

Note: $C^{\dagger}A$ is a solution to $\underset{X}{min} \|A - CX\| = C^{\dagger}A$, where $\| \cdot \|$ is the 2 or Frobenius norm.

Column Subset Selection Problem (CSSP)

Given $A \in \mathbb{R}^{m \times n}$ and $k < n$, find the column submatrix $C = [a_{i_1} \dots a_{i_k}]$ which minimizes

$$\|A - CC^{\dagger}A\|_2$$

Note: $C^{\dagger}A$ is a solution to $\underset{X}{min} \|A - CX\| = C^{\dagger}A$, where $\|\cdot\|$ is the 2 or Frobenius norm.

Interpretation: Which columns of A are the most representative of the data?

Column Subset Selection

Column Subset Selection Problem (CSSP)

Given $A \in \mathbb{R}^{m\times n}$ and $k < n$, find the column submatrix $C = [a_{i_1} \dots a_{i_k}]$ which minimizes

$$\|A - CC^\dagger A\|_2$$

Note: $C^\dagger A$ is a solution to $\min_X \|A - CX\| = C^\dagger A$, where $\|\cdot\|$ is the 2 or Frobenius norm.

Interpretation: Which columns of A are the most representative of the data?

Issue: CSSP is NP–hard [Shitov, '17]

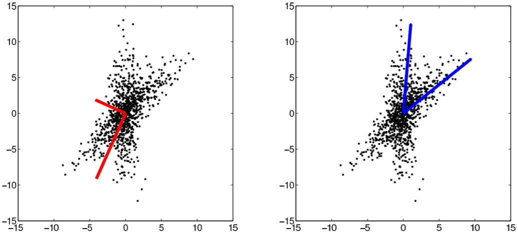
Column Subset Selection: Applications

Using actual columns of the data gives an interpretable representation

Column Subset Selection: Applications

Using actual columns of the data gives an interpretable representation

Can capture multiple directions of variability



[Sorensen–Embree, SICOMP '16] Red vectors are Principal Coordinates, Blue vectors are the columns of $CC^\dagger$ for a certain column submatrix

How?

Random Sampling Methods I: Sample w/ or w/out replacement from some distribution over the column indices

How?

Random Sampling Methods I: Sample w/ or w/out replacement from some distribution over the column indices

- Uniform: $p_i := \frac{1}{n}$

How?

Random Sampling Methods I: Sample w/ or w/out replacement from some distribution over the column indices

- Uniform: $p_i := \frac{1}{n}$
- Column Length: $p_i = \frac{|A(:,i)|^2}{\|A\|_F^2}$

How?

Random Sampling Methods I: Sample w/ or w/out replacement from some distribution over the column indices

- Uniform: $p_i := \frac{1}{n}$
- Column Length: $p_i = \frac{|A(:,i)|^2}{\|A\|_F^2}$
- Leverage Scores: $p_i^{(k)} := \frac{1}{k}|V_k(i,:)|^2$

How?

Random Sampling Methods I: Sample w/ or w/out replacement from some distribution over the column indices

- Uniform: $p_i := \frac{1}{n}$
- Column Length: $p_i = \frac{|A(:,i)|^2}{\|A\|_F^2}$
- Leverage Scores: $p_i^{(k)} := \frac{1}{k}|V_k(i,:)|^2$

Random Sampling Methods II: Bernoulli trials on each column

- Typically Column Length – requires rescaling columns in the reconstruction phase

How?

Random Sampling Methods I: Sample w/ or w/out replacement from some distribution over the column indices

- Uniform: $p_i := \frac{1}{n}$
- Column Length: $p_i = \frac{|A(:,i)|^2}{\|A\|_F^2}$
- Leverage Scores: $p_i^{(k)} := \frac{1}{k}|V_k(i,:)|^2$

Random Sampling Methods II: Bernoulli trials on each column

- Typically Column Length – requires rescaling columns in the reconstruction phase

Deterministic Sampling Methods:

- Discrete Empirical Interpolation Method (DEIM) [Gu–Eisenstat, SICOMP '96, Sorensen–Embree, SICOMP '16]
- Greedy Column Selectin [Avron–Boutsidis, SIMAX '13]

Tradeoffs:

- Computational Complexity: Leverage $\gg$ Col Length $\gg$ Uniform

Tradeoffs:

- Computational Complexity: Leverage $\gg$ Col Length $\gg$ Uniform
- Guarantees: Leverage Scores $>$ Col Length $>$ Uniform

Tradeoffs:

- Computational Complexity: Leverage $\gg$ Col Length $\gg$ Uniform
- Guarantees: Leverage Scores $>$ Col Length $>$ Uniform

Tradeoffs:

- Computational Complexity: Leverage $\gg$ Col Length $\gg$ Uniform
- Guarantees: Leverage Scores $>$ Col Length $>$ Uniform

Exception: If A has incoherent columns, Uniform sampling works very well [Chiu–Demanet, SIMAX '13]

Tradeoffs:

- Computational Complexity: Leverage $\gg$ Col Length $\gg$ Uniform
- Guarantees: Leverage Scores $>$ Col Length $>$ Uniform

Exception: If A has incoherent columns, Uniform sampling works very well [Chiu–Demanet, SIMAX '13]

Dimensionality Reduction: $C = U_d \Sigma_d V_d^T \Rightarrow U_d^T A \in \mathbb{R}^{d \times N}$

Problem 2: Preserve distance structure of data while significantly reducing the dimension

Problem 2: Preserve distance structure of data while significantly reducing the dimension

Given $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ find $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ with $d \ll D$ such that

$$|x_i - x_j| = |\Phi(x_i) - \Phi(x_j)|$$

Problem 2: Preserve distance structure of data while significantly reducing the dimension

Given $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^D$ find $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ with $d \ll D$ such that

$$|x_i - x_j| = |\Phi(x_i) - \Phi(x_j)|$$

Relaxation: Given $\varepsilon > 0$, find Φ such that

$$(1-\varepsilon)|x_i - x_j|^2 \leq |\Phi(x_i) - \Phi(x_j)|^2 \leq (1+\varepsilon)|x_i - x_j|^2 \tag{1}$$

Problem 2: Preserve distance structure of data while significantly reducing the dimension

Given $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^D$ find $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ with $d \ll D$ such that

$$|x_i - x_j| = |\Phi(x_i) - \Phi(x_j)|$$

Relaxation: Given $\varepsilon > 0$, find Φ such that

$$(1-\varepsilon)|x_i - x_j|^2 \leq |\Phi(x_i) - \Phi(x_j)|^2 \leq (1+\varepsilon)|x_i - x_j|^2 \qquad (1)$$

Note: If Q orthogonal and $t \in \mathbb{R}^D$, then

$$|(Qx_i - t) - (Qx_j - t)| = |Q(x_i - x_j)| = |x_i - x_j|.$$

Solution 1: Johnson–Lindenstrauss Embeddings

If Φ is *linear*, then equation reduces to

$$(1-\varepsilon)|y|^2 \leq |\Phi y|^2 \leq (1+\varepsilon)|y|^2, \quad y \in E = \{x_i - x_j\}.$$

Solution 1: Johnson–Lindenstrauss Embeddings

If Φ is *linear*, then equation reduces to

$$(1-\varepsilon)|y|^2 \leq |\Phi y|^2 \leq (1+\varepsilon)|y|^2, \quad y \in E = \{x_i - x_j\}.$$

Do such matrices Φ exist?

Solution 1: Johnson–Lindenstrauss Embeddings

If Φ is *linear*, then equation reduces to

$$(1-\varepsilon)|y|^2 \leq |\Phi y|^2 \leq (1+\varepsilon)|y|^2, \quad y \in E = \{x_i - x_j\}.$$

Do such matrices Φ exist?

Theorem (Johnson–Lindenstrauss Lemma)

Let $\varepsilon \in (0,1)$ and $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$ be arbitrary. Let $d \geq C\varepsilon^{-2}\log(N)$. Then there exists $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ such that the above holds.

Solution 1: Johnson–Lindenstrauss Embeddings

If Φ is *linear*, then equation reduces to

$$(1-\varepsilon)|y|^2 \leq |\Phi y|^2 \leq (1+\varepsilon)|y|^2, \quad y \in E = \{x_i - x_j\}.$$

Do such matrices Φ exist?

Theorem (Johnson–Lindenstrauss Lemma)

Let $\varepsilon \in (0,1)$ and $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$ be arbitrary. Let $d \geq C\varepsilon^{-2}\log(N)$. Then there exists $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ such that the above holds.

Theorem

Let $x \in \mathbb{R}^D$ be fixed. Let $\Phi \in \mathbb{R}^{d \times D}$ be a matrix whose entries are i.i.d. Gaussian ($\mathcal{N}(0, \frac{1}{d})$). Then

$$\mathbb{P}\left[(1-\varepsilon)|x|^2 \leq |\Phi x|^2 \leq (1+\varepsilon)|x|^2\right] \geq 1 - 2e^{-(\varepsilon^2-\varepsilon^3)d/4}.$$

Solution 1: Johnson–Lindenstrauss Embeddings

If Φ is *linear*, then equation reduces to

$$(1-\varepsilon)|y|^2 \leq |\Phi y|^2 \leq (1+\varepsilon)|y|^2, \quad y \in E = \{x_i - x_j\}.$$

Do such matrices Φ exist?

Theorem (Johnson–Lindenstrauss Lemma)

Let $\varepsilon \in (0,1)$ and $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$ be arbitrary. Let $d \geq C\varepsilon^{-2}\log(N)$. Then there exists $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ such that the above holds.

Theorem

Let $x \in \mathbb{R}^D$ be fixed. Let $\Phi \in \mathbb{R}^{d\times D}$ be a matrix whose entries are i.i.d. Gaussian ($\mathcal{N}(0, \frac{1}{d})$). Then

$$\mathbb{P}\left[(1-\varepsilon)|x|^2 \leq |\Phi x|^2 \leq (1+\varepsilon)|x|^2\right] \geq 1 - 2e^{-(\varepsilon^2-\varepsilon^3)d/4}.$$

Corollary

Gaussian matrices satisfy the Johnson–Lindenstrauss Lemma with high probability.

Solution 2: Multi-dimensional Scaling (MDS)

Solution 2: Multi-dimensional Scaling (MDS)

Given: Pairwise distance matrix $D_{ij}^X = |x_i - x_j|$ for a set of points $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^n$ (Note: we do not necessarily have to observe X itself!)

Solution 2: Multi-dimensional Scaling (MDS)

Given: Pairwise distance matrix $D^X_{ij} = |x_i - x_j|$ for a set of points $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^n$ (Note: we do not necessarily have to observe X itself!)

Goal: Find $Y = \{y_1, \ldots, y_N\} \subset \mathbb{R}^d$ with $d \ll n$ such that $|y_i - y_j| \approx |x_i - x_j|$

Solution 2: Multi-dimensional Scaling (MDS)

Given: Pairwise distance matrix $D_{ij}^X = |x_i - x_j|$ for a set of points $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^n$ (Note: we do not necessarily have to observe X itself!)

Goal: Find $Y = \{y_1, \dots, y_N\} \subset \mathbb{R}^d$ with $d \ll n$ such that $|y_i - y_j| \approx |x_i - x_j|$

General Formulation: Given a similarity measure $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, find $Y \subset \mathbb{R}^d$ which minimizes

$$L(f, D^X, y_1, \dots, y_N) := \left(\frac{\sum_{i,j} (f(y_i, y_j) - D_{ij}^X)^2}{\sum_{i,j} (D_{ij}^X)^2} \right)^{\frac{1}{2}}$$

MDS

The Classical MDS algorithm is a bit simpler: it minimizes

$$\text{Strain}(y_1, \ldots, y_N) := \left(\frac{\sum_{i,j} (B_{ij} - \langle y_i, y_j \rangle)^2}{\sum_{i,j} B_{i,j}^2} \right)^{\frac{1}{2}}$$

where B is an auxiliary matrix defined by

$$B = -\frac{1}{2} J (D^X)^{(2)} J, \qquad J := I - \frac{1}{N} \mathbb{1}\mathbb{1}^T$$

where

$$(D^X)_{ij}^{(2)} = (D_{ij}^X)^2 = |x_i - x_j|^2$$

MDS

The Classical MDS algorithm is a bit simpler: it minimizes

$$\text{Strain}(y_1, \ldots, y_N) := \left(\frac{\sum_{i,j} (B_{ij} - \langle y_i, y_j \rangle)^2}{\sum_{i,j} B_{i,j}^2} \right)^{\frac{1}{2}}$$

where B is an auxiliary matrix defined by

$$B = -\frac{1}{2} J (D^X)^{(2)} J, \qquad J := I - \frac{1}{N} \mathbb{1}\mathbb{1}^T$$

where

$$(D^X)^{(2)}_{ij} = (D^X_{ij})^2 = |x_i - x_j|^2$$

B is a “double centering" of the distance matrix

MDS

Definition

A matrix $D \in \mathbb{R}^{N \times N}$ is a distance matrix provided

- $D = D^T$,
- $D_{ii} = 0$ for all i
- $D_{ij} \geq 0$ for all $i \neq j$

MDS

Definition

A matrix $D \in \mathbb{R}^{N \times N}$ is a distance matrix provided

- $D = D^T$,
- $D_{ii} = 0$ for all i
- $D_{ij} \geq 0$ for all $i \neq j$

Definition

A distance matrix $D \in \mathbb{R}^{N \times N}$ is Euclidean if there exists a set $\{x_1, \ldots, x_N\}$ in $\mathbb{R}^d$ for some d such that $D_{ij} = |x_i - x_j|$.

MDS

Definition

A matrix $D \in \mathbb{R}^{N\times N}$ is a distance matrix provided

- $D = D^T$,
- $D_{ii} = 0$ for all i
- $D_{ij} \geq 0$ for all $i \neq j$

Definition

A distance matrix $D \in \mathbb{R}^{N\times N}$ is Euclidean if there exists a set $\{x_1, \ldots, x_N\}$ in $\mathbb{R}^d$ for some d such that $D_{ij} = |x_i - x_j|$.

Note the notion of a distance matrix is much more general. We will characterize when a distance matrix is Euclidean.

MDS

Theorem (Householder–Young, ’38)

Let D be a distance matrix, and $B = -\frac{1}{2}JD^{(2)}J$. Then D is Euclidean if and only if B is SPSD.

MDS

Theorem (Householder–Young, '38)

Let D be a distance matrix, and $B = -\frac{1}{2}JD^{(2)}J$. Then D is Euclidean if and only if B is SPSD.

Theorem (Forward Direction)

If D is a distance matrix for X, then $B = JX^TXJ$, and hence is SPSD.

MDS

Theorem (Householder–Young, '38)

Let D be a distance matrix, and $B = -\frac{1}{2}JD^{(2)}J$. Then D is Euclidean if and only if B is SPSD.

Theorem (Forward Direction)

If D is a distance matrix for X, then $B = JX^TXJ$, and hence is SPSD.

Theorem (Reverse Direction)

Conversely, if B is SPSD and has rank k, and $B = V\Lambda V^T$ (by the Spectral Theorem), then choosing $X^T = V_k\Lambda_k^{\frac{1}{2}}$ gives $X \subset \mathbb{R}^k$ such that $|x_i - x_j| = D_{ij}$.

Classical MDS Algorithm

Given: $D = D^{(2)}$ – pairwise square-distance matrix, embedding dimension d

Classical MDS Algorithm

Given: $D = D^{(2)}$ – pairwise square-distance matrix, embedding dimension d

Compute: $B = -\frac{1}{2}JD^{(2)}J$

Classical MDS Algorithm

Given: $D = D^{(2)}$ – pairwise square-distance matrix, embedding dimension d

Compute: $B = -\frac{1}{2} J D^{(2)} J$

Factor: $B = V_d \Lambda_d V_d^T$ (truncated SVD if B has larger rank)

Classical MDS Algorithm

Given: $D = D^{(2)}$ – pairwise square-distance matrix, embedding dimension d

Compute: $B = -\frac{1}{2}JD^{(2)}J$

Factor: $B = V_d\Lambda_d V_d^T$ (truncated SVD if B has larger rank)

Set: $y_i = (V_d\Lambda_d^{\frac{1}{2}})_{i:}$ – final embedded points

MDS

Claim: This Y minimizes strain

MDS

Claim: This Y minimizes strain

$$\text{Strain}(y_1, \ldots, y_N) := \left(\frac{\sum_{i,j} (B_{ij} - \langle y_i, y_j \rangle)^2}{\sum_{i,j} B_{i,j}^2} \right)^{\frac{1}{2}} = \frac{\|B - Y^T Y\|_F}{\|B\|_F}$$

MDS

Claim: This Y minimizes strain

$$\text{Strain}(y_1, \dots, y_N) := \left(\frac{\sum_{i,j}(B_{ij} - \langle y_i, y_j \rangle)^2}{\sum_{i,j} B_{i,j}^2} \right)^{\frac{1}{2}} = \frac{\|B - Y^T Y\|_F}{\|B\|_F}$$

Recall $V_d \Lambda_d V_d^T$ is a solution to $\min_{X:\text{rank}(X)\leq d} \|B - X\|_F$ by the Eckhart–Young–Mirsky Theorem

MDS

Claim: This Y minimizes strain

$$\text{Strain}(y_1, \dots, y_N) := \left(\frac{\sum_{i,j} (B_{ij} - \langle y_i, y_j \rangle)^2}{\sum_{i,j} B_{i,j}^2} \right)^{\frac{1}{2}} = \frac{\|B - Y^T Y\|_F}{\|B\|_F}$$

Recall $V_d \Lambda_d V_d^T$ is a solution to $\min_{X:\text{rank}(X) \le d} \|B - X\|_F$ by the Eckhart–Young–Mirsky Theorem

So setting $Y^T = V_d \Lambda_d^{\frac{1}{2}}$ so that $X = Y^T Y$ is a solution to $\min_{Y:\text{rank}(Y) \le d} \|B - Y^T Y\|_F$

MDS

Claim: This Y minimizes strain

$$\text{Strain}(y_1, \ldots, y_N) := \left(\frac{\sum_{i,j} (B_{ij} - \langle y_i, y_j \rangle)^2}{\sum_{i,j} B_{i,j}^2} \right)^{\frac{1}{2}} = \frac{\|B - Y^T Y\|_F}{\|B\|_F}$$

Recall $V_d \Lambda_d V_d^T$ is a solution to $\min_{X:\text{rank}(X) \leq d} \|B - X\|_F$ by the Eckhart–Young–Mirsky Theorem

So setting $Y^T = V_d \Lambda_d^{\frac{1}{2}}$ so that $X = Y^T Y$ is a solution to $\min_{Y:\text{rank}(Y) \leq d} \|B - Y^T Y\|_F$

So MDS is E–Y–M in yet another guise!

MDS

Comments:

- The previous algorithm is often called “classical MDS" (but sometimes people call the version that minimizes stress classical MDS, so take care when reading

Comments:

- The previous algorithm is often called “classical MDS" (but sometimes people call the version that minimizes stress classical MDS, so take care when reading
- This version doesn’t necessarily keep our objective of maintaining pairwise distances

MDS

Comments:

- The previous algorithm is often called “classical MDS" (but sometimes people call the version that minimizes stress classical MDS, so take care when reading
- This version doesn’t necessarily keep our objective of maintaining pairwise distances
- But it is easy because the solution is just the SVD!

Metric MDS

Recall our general loss function:

$$L(f, D^X, y_1, \dots, y_N) := \left(\frac{\sum_{i,j} (f(y_i, y_j) - D_{ij}^X)^2}{\sum_{i,j} (D_{ij}^X)^2} \right)^{\frac{1}{2}}$$

Metric MDS

Recall our general loss function:

$$L(f, D^X, y_1, \ldots, y_N) := \left(\frac{\sum_{i,j} (f(y_i, y_j) - D_{ij}^X)^2}{\sum_{i,j} (D_{ij}^X)^2} \right)^{\frac{1}{2}}$$

Metric MDS is when we take $f(y_i, y_j) = |y_i - y_j|$

Metric MDS

Recall our general loss function:

$$L(f, D^X, y_1, \ldots, y_N) := \left(\frac{\sum_{i,j} (f(y_i, y_j) - D_{ij}^X)^2}{\sum_{i,j} (D_{ij}^X)^2} \right)^{\frac{1}{2}}$$

Metric MDS is when we take $f(y_i, y_j) = |y_i - y_j|$
In this case, the loss function is called stress

$$\text{stress}(D^X, y_1, \ldots, y_N) := \left(\frac{\sum_{i,j} (|y_i - y_j| - D_{ij}^X)^2}{\sum_{i,j} (D_{ij}^X)^2} \right)^{\frac{1}{2}}$$

Metric MDS

Unlike Classical MDS, Metric MDS does not have a closed form solution. The embedded points Y are found via optimization (e.g., stress majorization or (stochastic) gradient descent)

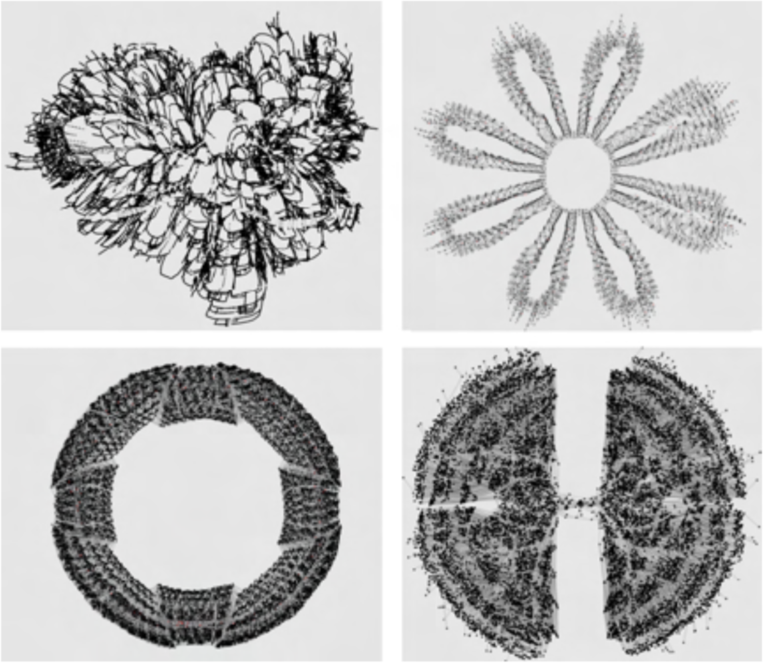
Metric MDS

Unlike Classical MDS, Metric MDS does not have a closed form solution. The embedded points Y are found via optimization (e.g., stress majorization or (stochastic) gradient descent)
Applications: Graph Drawing

Input: $G = (V, E, w)$, and $D_{ij} = d_G(v_i, v_j)$ (graph-theoretic shortest path distance)

Output: drawing of the graph in $\mathbb{R}^2$ (typically) or $\mathbb{R}^3$

Minimizing stress keeps the points from colliding



Exploratory Data Visualization