Fundamentals of AI

Dimensionality reduction

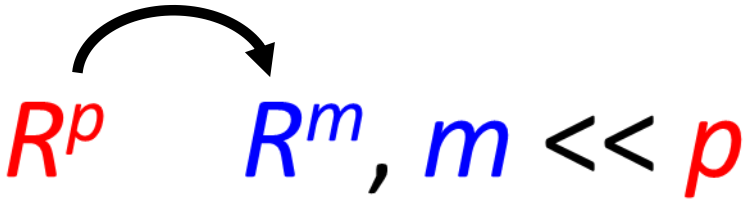
Multi-dimensional scaling (MDS):

example of projective approach to dimensionality reduction

Projective vs Injective methods

Projective

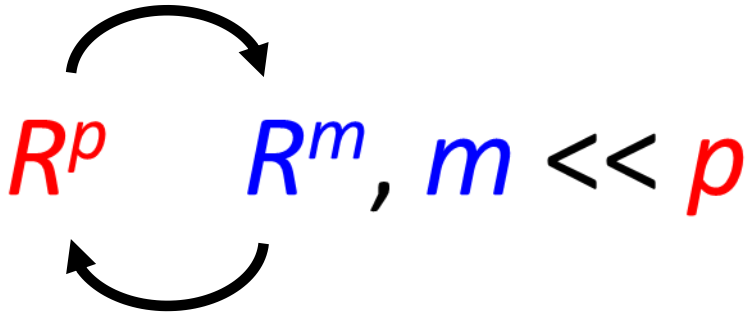
ENCODE or PROJECT



Variant 1: The projector is known for any $\boldsymbol{y} \in \boldsymbol{R^p}$
Variant 2: The projector is know only for $\boldsymbol{y} \in \boldsymbol{X}$

Injective*

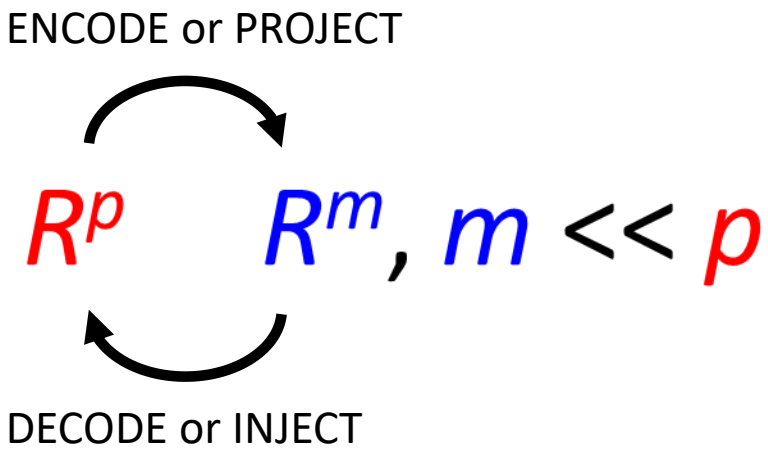
ENCODE or PROJECT



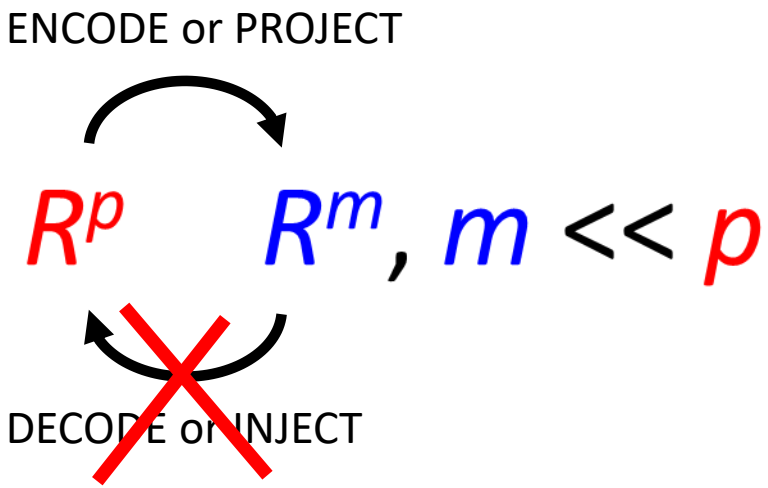
DECODE or INJECT

*we know where to find ANY point from R^m in R^p

PCA, ICA, NMF, FA are injective methods



Multi-dimensional scaling is projective method,



Variant 2: The projector is know only for $\boldsymbol{y} \in \boldsymbol{X}$

Input: a distance or dissimilarity matrix

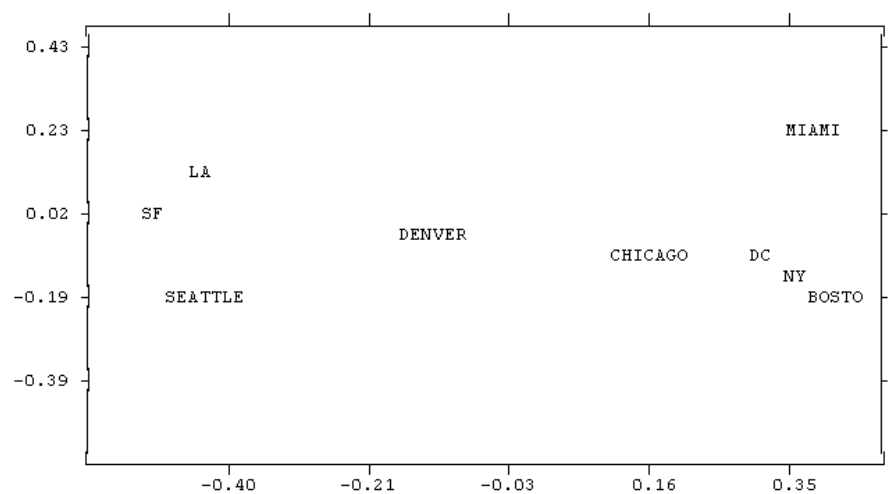
- How to arrange points in 2D (or mD) that the Euclidean distance between them would reproduce the input matrix as much as possible?

From http://www.analytictech.com/networks/mds.htm

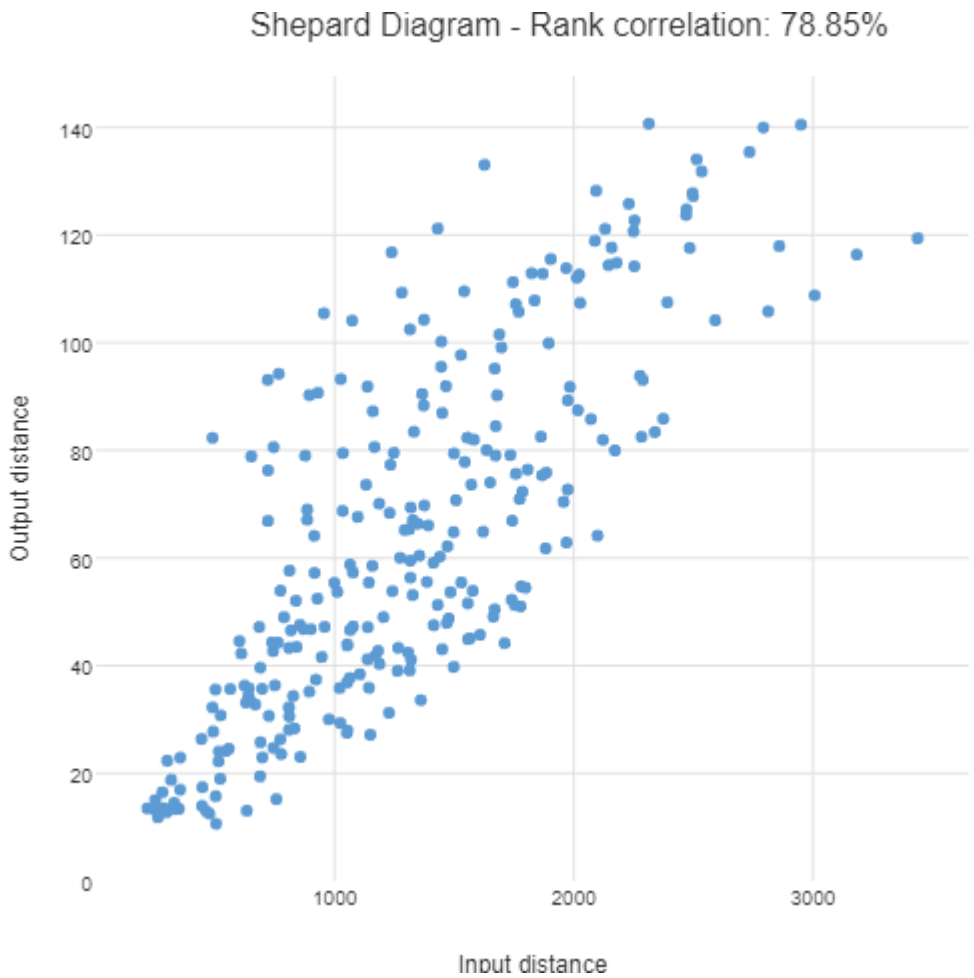
Input distance matrix

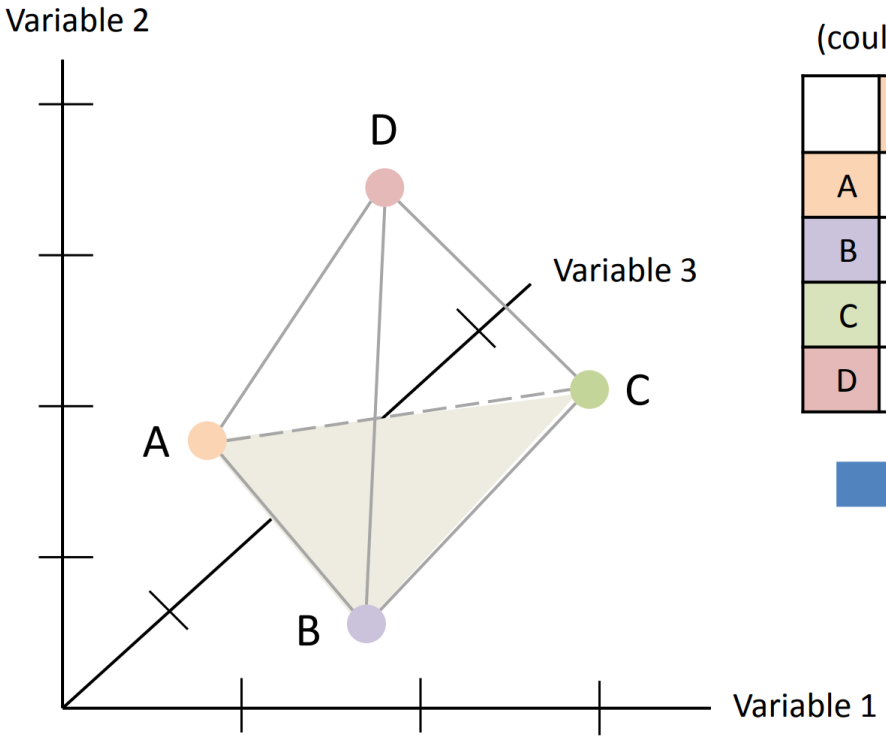
		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0

Output point position



Shepard Diagram

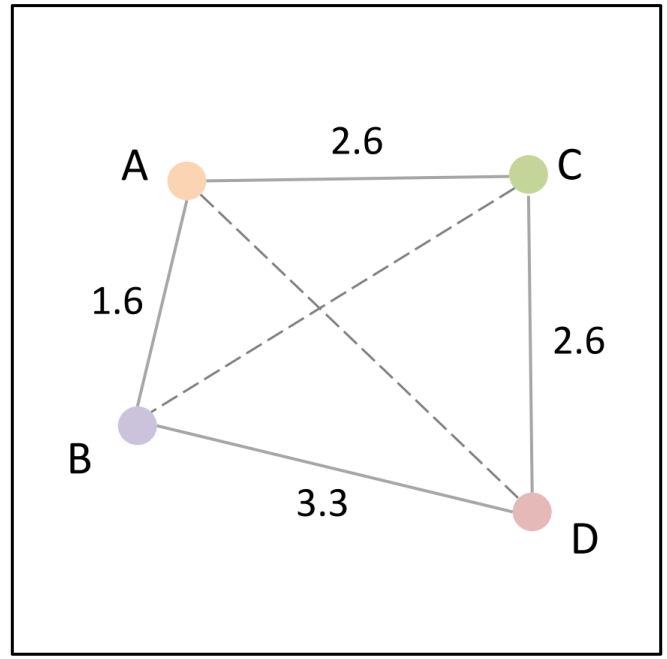




Euclidian
(could be any distance matrix)

	A	B	C	D
A	0	1.6	2.6	2.4
B	1.6	0	2.5	3.3
C	2.6	2.5	0	1.7
D	2.4	3.3	1.7	0

Plot in 2D by distance



Data.ID	Varable1	Variable2	Variable3
A	0.9	1.9	1.5
B	1.7	0.5	1.6
C	3	2	3.1
D	1.9	3.5	3

When we *compress* our 3D image to 2D we cannot accurately plot the true distances
E.g. the distances between AD and BC are too big in the image

The difference between the data point position in 2D (or # of dimensions we consider with NMDS) and the distance calculations (based on multivariate) is the **STRESS** we are trying to optimize

Taken from https://sites.ualberta.ca/~lkgray/uploads/7/3/6/2/7362679/slides-nmds.pdf

Three approaches

- Classical multidimensional scaling
- Metric multidimensional scaling (mMDS)
- Non-metric multidimensional scaling (nMDS)

Classical MDS aka Principal Coordinates Analysis (PCoA), Torgerson Scaling or Torgerson–Gower

- Set up the **squared proximity matrix** $D^{(2)} = \left[d_{ij}^2\right]$
- Apply **double centering** to $D^{(2)}$: $B = -\frac{1}{2} J D^{(2)} J \ , \ J = I - \frac{1}{N} 11'$
- Determine the largest eigenvalues $\lambda_1, \lambda_2, .., \lambda_m$ and corresponding **eigenvectors** $\boldsymbol{e_1}, \ldots, \boldsymbol{e_m}$ of B
- Now, $X = E_m \Lambda_{\mathrm{m}}^{\frac{1}{2}}$, where E_m is the matrix of m eigenvectors and Λ is the diagonal matrix of eigenvalues of B

Classical MDS minimizes strain defined as

$$\mathrm{Strain}_D(x_1, x_2, \ldots, x_N) = \left(\frac{\sum_{i,j} \left(b_{ij} - \langle x_i, x_j \rangle\right)^2}{\sum_{i,j} b_{ij}^2} \right)^{1/2}$$

Classical MDS

Some explanations

- *Double centering*: transforming a rectangular matrix to the one having mean values of rows equal to zero and mean values of columns equal to zero
- Double centering is done by subtracting all row means and column means and adding the global mean to each element of the matrix
- Centering matrix $C_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}$ 3 x 3

$$C_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix}$$

Classical MDS

Some explanations

- Matrix *B* is the matrix of scalar products (Gramm matrix)
- In Euclidean space it is connected with the matrix of squared distances $D^{(2)} = \left[d_{ij}^2\right]$ via double-centering

- *Exercise: prove this or demonstrate in a program*

Classical MDS
Some explanations

- **Reminder**: if we want to approximate a symmetric non-negative matrix *A* by a matrix of rank *m*, we need to compute *m* eigenvectors corresponding to the largest eigenvalues, and then A $\approx E_m \Lambda_{\mathrm{m}} (E_m)^T$

- **Remark**: any symmetric non-negative matrix has only non-negative eigenvalues

Classical MDS
Difference and similarity with PCA

- PCA is based on computing the largest eigenvectors of the empirical covariance matrix $\boldsymbol{C} = \boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$
- $\boldsymbol{C}$ has dimension p x p, where p is the number of variables

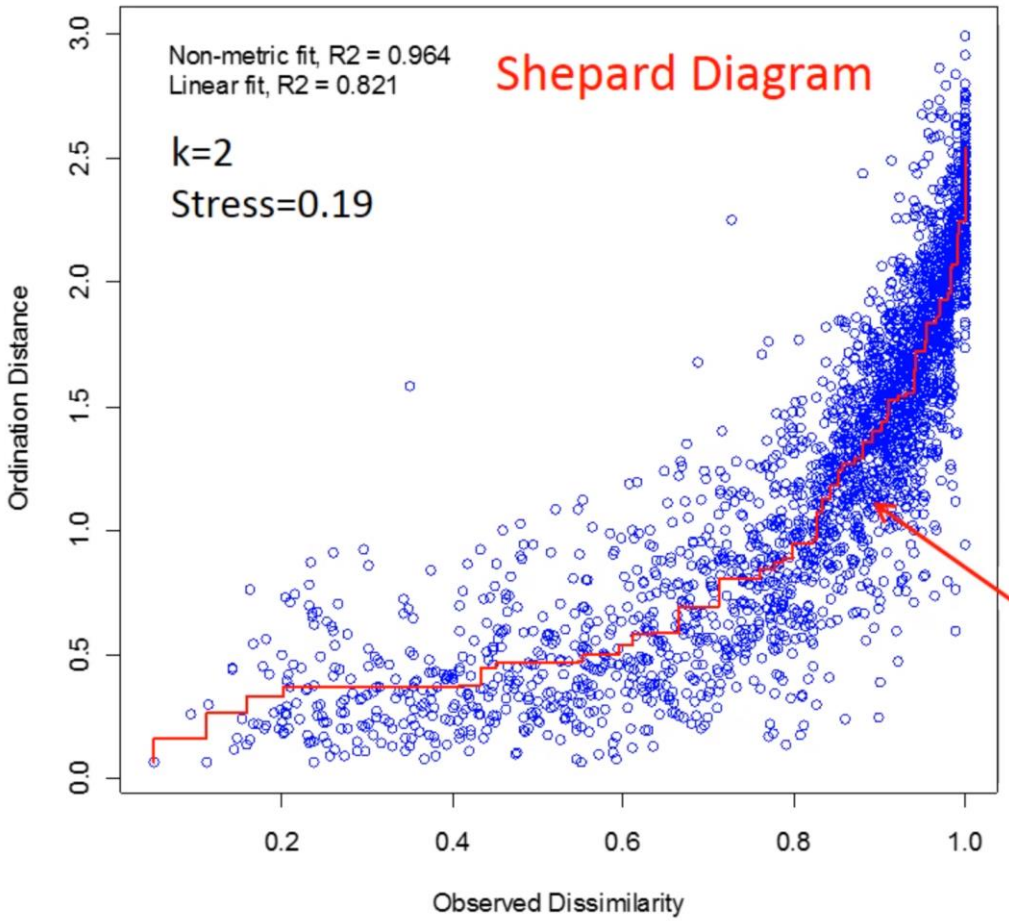
- MDS is based on computing the largest eigenvectors of the Gramm matrix $\boldsymbol{B} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$
- $\boldsymbol{B}$ has dimension N x N, where N is the number of objects

- Eigenvalues Λ are the same
- This corresponds to the difference between right and left singular vectors in SVD : $\boldsymbol{X} = \boldsymbol{U}\Lambda\boldsymbol{V}$

Metric MDS minimizes Stress

$$\mathrm{Stress}_D(x_1, x_2, \ldots, x_N) = \left(\frac{\sum_{i,j} \left(d_{ij} - \|x_i - x_j\| \right)^2}{\sum_{i,j} d_{ij}^2} \right)^{1/2}$$

Non-metric MDS deals with ranks of distances instead of their values



Stress calculated from residuals around monotone regression line

Ideally, all points should fall on monotonic line (increasing ordination distance = increasing observed distance)

From https://www.youtube.com/watch?v=Kl49qI3XJKY

What do you have to take

- MDS takes as input the complete distance matrix : bad scalability
- MDS finds position of points in R^m without offering INJECT function (from any point in R^m to R^p)
- Many variants of MDS, flexibility
- Many non-linear dimensionality reduction methods are based on the principles of MDS (base example: Kernel PCA)