A Brief Introduction To

Dimensionality Reduction

Megh Shukla
25th September, 2020

AGENDA

01 *INTRODUCTION*

Answering the What and the Why

02 CLASSICAL METHODS

PCA, LDA, Laplacian Eigenmaps, Locally Linear Embedding

03 MODERN METHODS

Autoencoders, t-SNE, UMAP

04 CONCLUSION

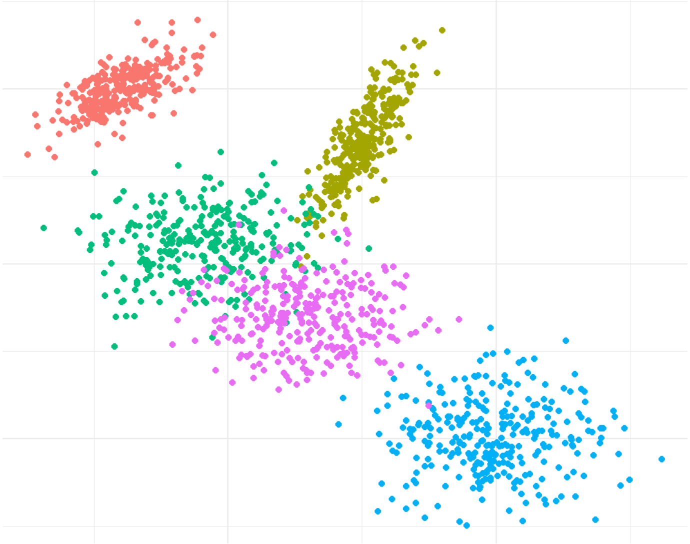
Comparison, Summary and Upcoming Research

“Mere paas
Data hai, GPU hai (sharing basis), CPU hai, Numpy hai.
Tumhare paas kya hai?”
–Naïve Megh

“Mere paas Time Complexity O(n) hai”
–Smart Megh

NEED FOR DIMENSIONALITY REDUCTION

Time and Space Complexity



Time Complexity

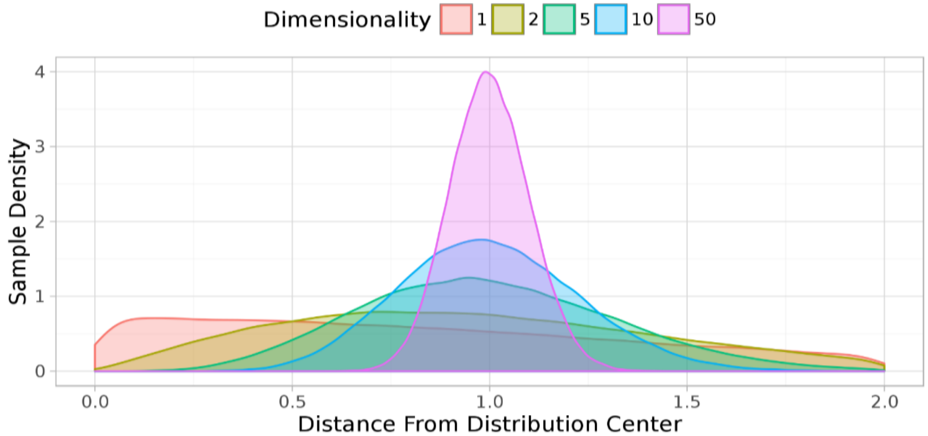
Memory

NEED FOR DIMENSIONALITY REDUCTION

Visualization

NEED FOR DIMENSIONALITY REDUCTION

Curse Of Dimensionality



$$\lim_{d\to\infty} \mathbb{E}\left(\frac{dist_{max}(d) - dist_{min}(d)}{dist_{min}(d)}\right) \to 0$$

VC Dimension - Overfitting

$\text{VC}_{\text{dim}}(\text{NeuralNet}) = O(WL\log W)$	$W = \#weights$ $\quad L = \#layers$

AGENDA

01 INTRODUCTION

Answering the What and the Why

02 *CLASSICAL METHODS*

PCA, LDA, Laplacian Eigenmaps, Locally Linear Embedding

03 MODERN METHODS

Autoencoders, t-SNE, UMAP

04 CONCLUSION

Comparison, Summary and Upcoming Research

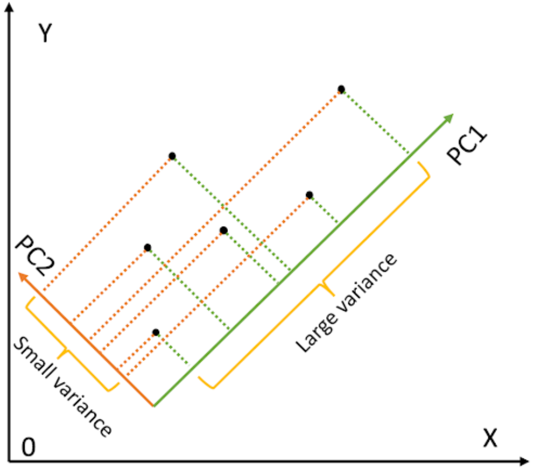
CLASSICAL METHODS

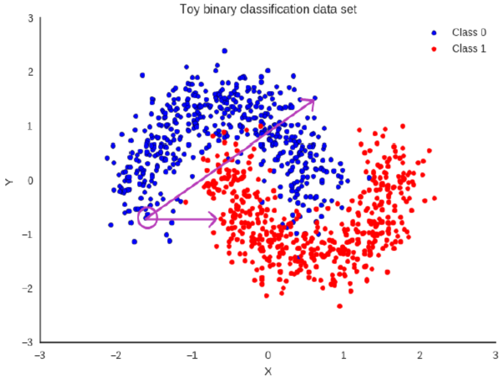
LINEAR METHODS

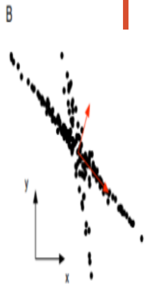
Principal Component Analysis

Explain the variance in the data!

Similarity to Linear Regression?







$\mathbb{X} : Samples \in \mathbb{R}^{N \times d}$
v : Projection Vector

Linear

$$PC_1 = \mathbb{X}v$$

Objective

$$arg\min_{v} ||\mathbb{X} - \mathbb{X}VV^T||$$

Constraint

$$V^T V = I$$

Solution

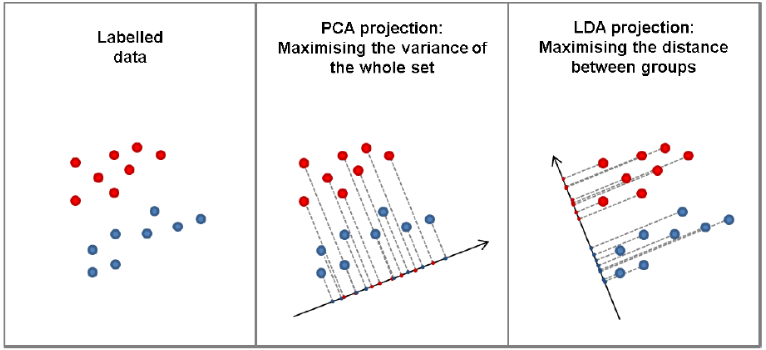
$$\mathbb{X}^T \mathbb{X} V = \lambda V$$

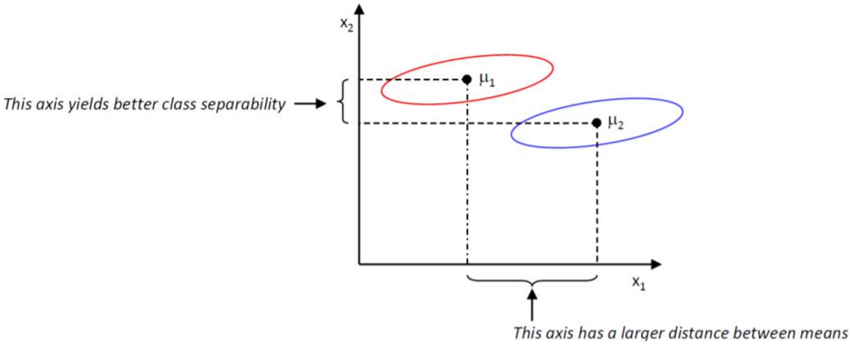
LINEAR METHODS

Linear Discriminant Analysis

Use class information!

Sometimes, no labels better than having labels!





LINEAR METHODS

Linear Discriminant Analysis

- Use class information!
- Sometimes, no labels better than having labels!

$$\mathbb{Y} = w^T \mathbb{X}$$

The embeddings are linearly dependent on the input

$$\mathcal{J}(w) = \frac{|\tilde{\mu_1} - \tilde{\mu_2}|^2}{\tilde{s_1}^2 + \tilde{s_2}^2}$$

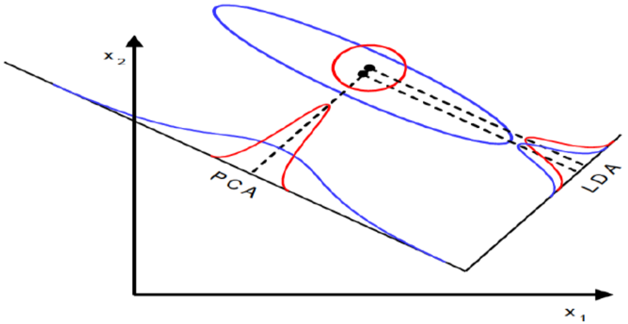
The objective to optimize

$$\mathcal{J}(w) = \frac{w^T S_B w}{w^T S_w w}$$

Optimize with respect to the weights

$$S_w^{-1} S_B w = \mathcal{J} w$$

… Reduced to Eigendecomposition



The solution!

$$w^* = S_w^{-1}(\mu_1 - \mu_2)$$

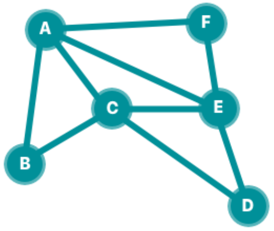
LINEAR METHODS - Graph Based Algorithms

Laplacian Eigenmaps

- Construct a Graph with Adjacency Matrix!
- Preserving local structure over global structure

Graph

- **Degree**
  No of neighbours
- **Adjacency**
  Presence of edge
- **Laplacian**
  Degree - Adjacency



V: {A,B,C,D,E,F}
E: {AB,AC,AF,BC,CD,CE,DE,EF}

=

	A	B	C	D	E	F
A	0	1	1	0	1	1
B	1	0	1	0	0	0
C	1	1	0	1	1	0
D	0	0	1	0	1	0
E	1	0	1	1	0	0
F	1	0	0	0	1	0

LINEAR METHODS - Graph Based Algorithms

Laplacian Eigenmaps

- Construct a Graph with Adjacency Matrix!
- Preserving local structure over global structure

$$\mathbb{J}(y) = \sum_{i,j} (y_i - y_j)^2 a_{ij}$$

$$\mathbb{J}(y) = \sum_{i,j} (y_i^2 + y_j^2 - 2y_i y_j) a_{ij}$$

$$\mathbb{J}(y) = \sum_{i} y_i^2 D_i + \sum_{j} y_j^2 D_j - 2 \sum_{i,j} y_i y_j a_{ij}$$

$$\mathbb{J}(y) = 2Y^T L Y$$

Constraint

$$Y^T D Y = 1$$

$$Y^T D \mathbf{1} = 0$$

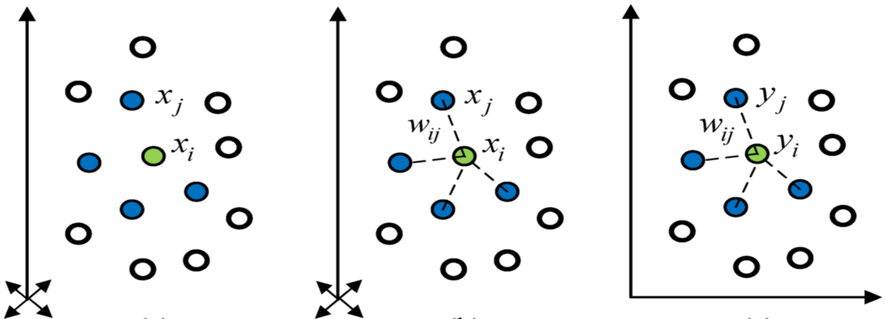
Eigenvalue Eigenvector everywhere!

LINEAR METHODS

Locally Linear Embedding

A node is known by the company he keeps!

Locally linear implies dense sampling!



(a) (b) (c)

The E-step…?

$$\mathcal{E}(W) = \sum_i |x_i - \sum_j W_{ij} x_j|^2$$

The M-step…?

$$\sum_i |y_i - \sum_j w_{ij} y_j|^2$$

AGENDA

01 INTRODUCTION

Answering the What and the Why

02 CLASSICAL METHODS

PCA, LDA, Laplacian Eigenmaps, Locally Linear Embedding

03 *MODERN METHODS*

Autoencoders, t-SNE, UMAP

04 CONCLUSION

Comparison, Summary and Upcoming Research

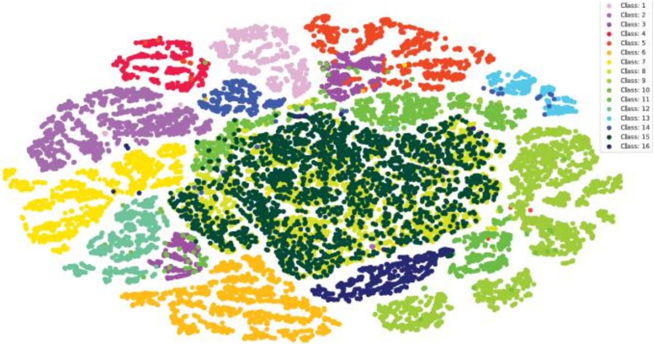
MODERN METHODS

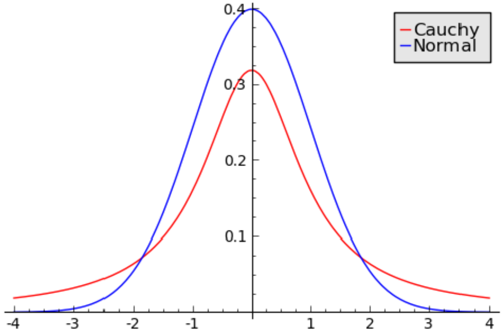
MODERN APPROACHES

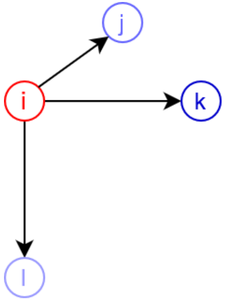
t-Distributed Stochastic Neighbour Embedding

Use class information!

Sometimes, no labels better than having labels!







$$p_{j|i} = \frac{exp(-\| x_i - x_j \|^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(-\| x_i - x_k \|^2 / 2\sigma_i^2)}$$

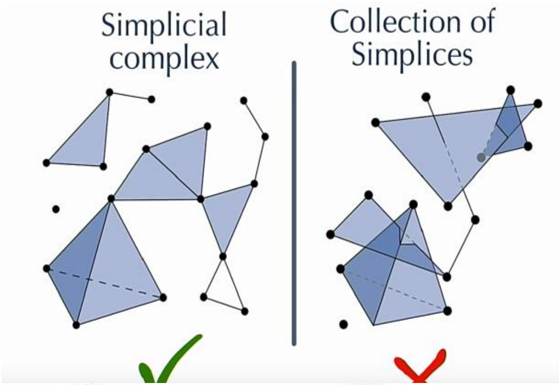
$$q_{j|i} = \frac{exp(-\| y_i - y_j \|^2)}{\sum_{k \neq i} exp(-\| y_i - y_k \|^2)}$$

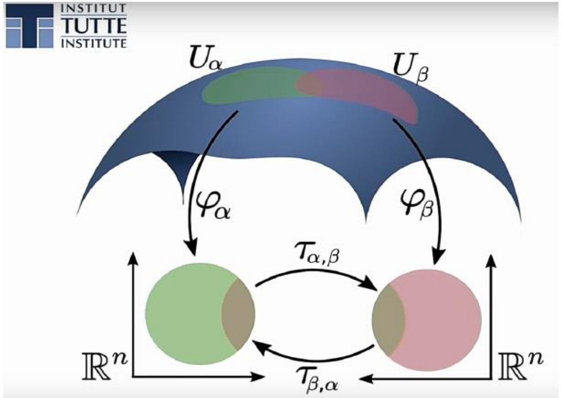
KL Divergence!

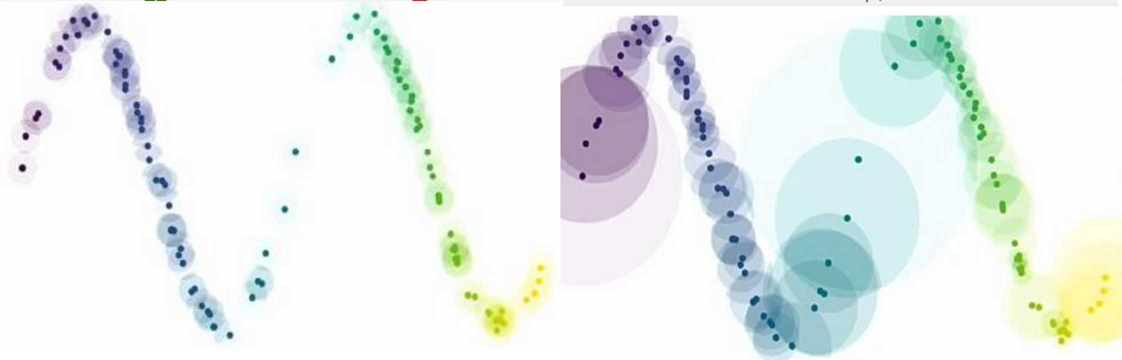
MODERN APPROACHES

Uniform Manifold Approximation and Projection

- Projection on a Reimannian Manifold
- Interpretability…?



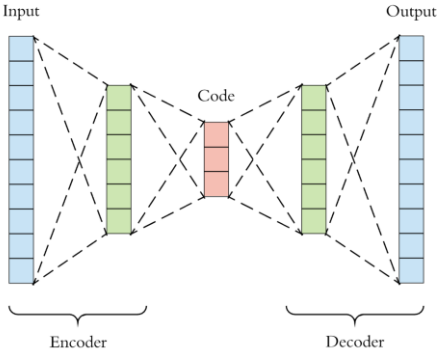




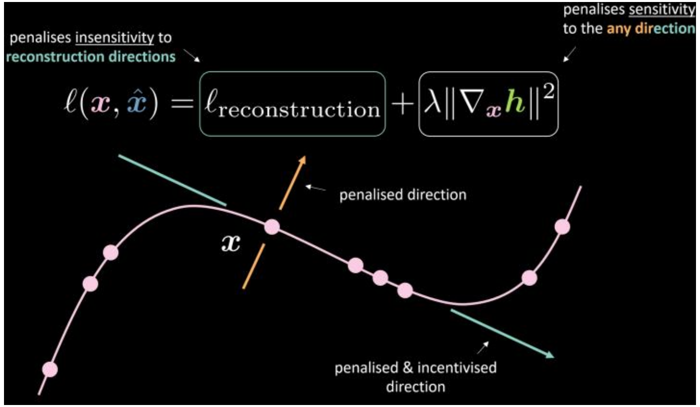
MODERN APPROACHES

Autoencoders

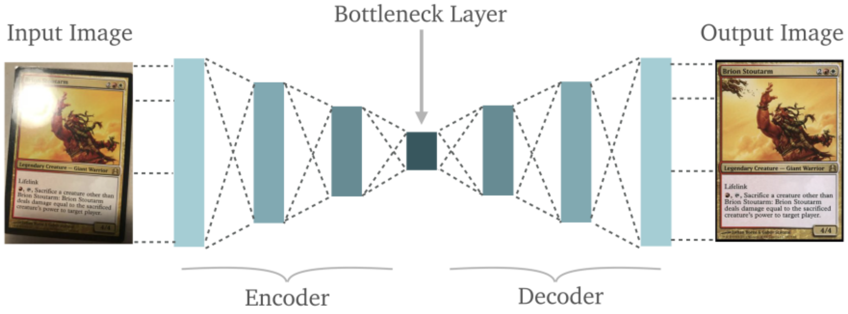
- Deep Learning magic!
- Overfitting … ?



Contractive Autoencoders



Denoising Autoencoders



AGENDA

01 INTRODUCTION

Answering the What and the Why

02 CLASSICAL METHODS

PCA, LDA, Laplacian Eigenmaps, Locally Linear Embedding

03 MODERN METHODS

Autoencoders, t-SNE, UMAP

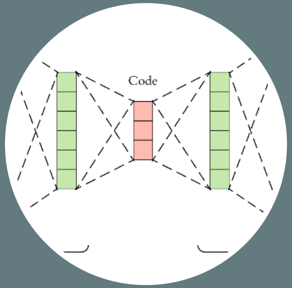
04 *CONCLUSION*

Comparison, Summary and Upcoming Research

IS DIMENSIONALITY REDUCTION SOLVED?

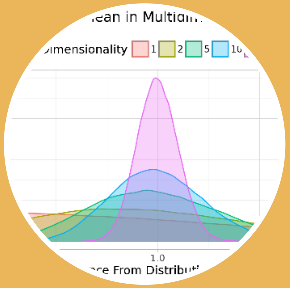
Sure the answer is No. But why?

SHORTCOMINGS



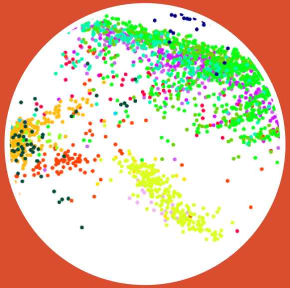
Parameterization

What if we have new data? Need more dimensions?



Curse Of Dimensionality

Euclidean distance can sometimes fail in high-dimensions



Visualization and Clustering

Are they the same problem? Or are they different?

THANK YOU!