Week 12 :- Dimension Reduction (11-1 - 11-4)

Learning Objectives :

- Advantages
- Steps of PCA
- Expand to K-PCA

Objective of dim red^n :

For $x \in \mathbb{R}^D$ find mapping $f: x \rightarrow y \in \mathbb{R}^d$

Such that $d \ll D$ [Eg: 5 & 100 etc]

We also want to ensure $\exists f^{-1}$ such that

we can almost reconstruct x given y.

Advantages:

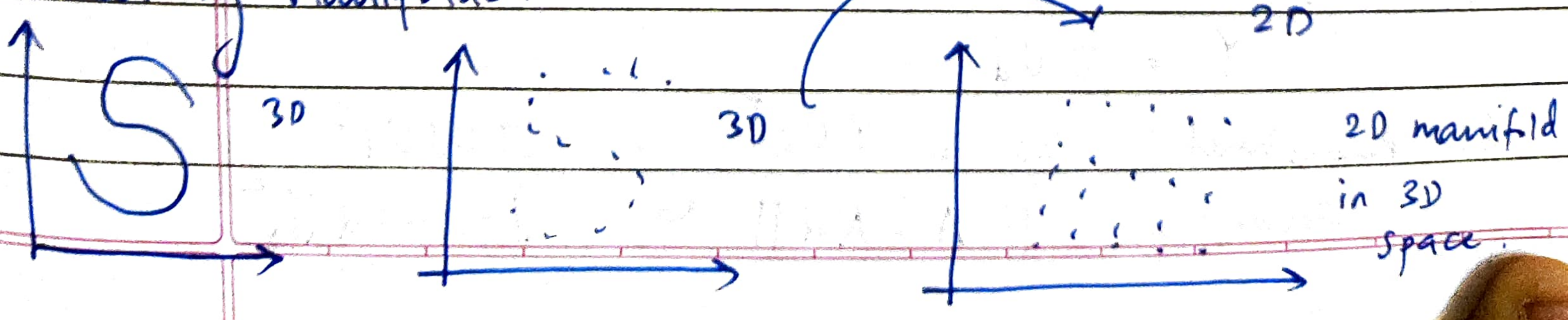
- Less redundancy, easier classification
- Smaller storage, faster search
- Unravel meaningful latent variables

Example: Face images (100x100 for eg)

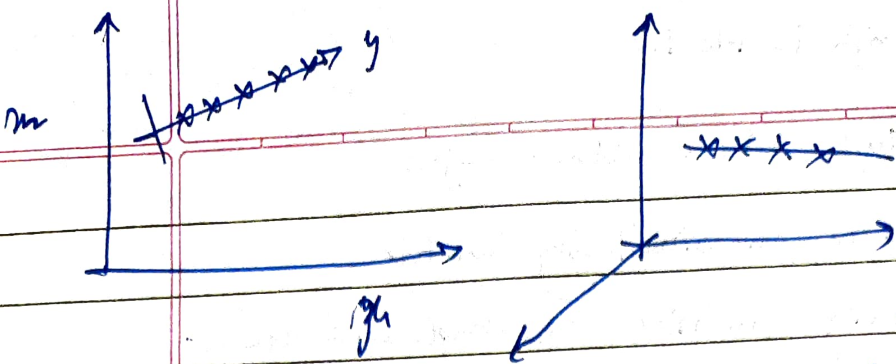
Differences $\Rightarrow$ Eye dimension, dist b/w eyes, size of nose etc.

These variables $\ll$ Pixels of face

Visualizing Manifolds:



Inherent linear dim of data



3D but on one line

or 2D (in plane)

They may not lie EXACTLY on the underlying hyperplane
(Some deviations ⇒ Reconstruction Error)

(11-2) Principal Component Analysis :

Obj:
- To select top $d < D$ orthogonal directions that explain max. variance in data

Outline:
- Mean-center the data
- Loop
  - Find direction of maximum variance (orthogonal to all previous dir)

⇒ Until desired % variance obtained

Algorithm:

1. Mean centering $\quad z_i = x_i - \mu_x$ $\quad$ (C = PSD
2. Covariance $\quad \cancel{N}C = Z^T Z / N$ $\quad$ Eigen non neg.
3. Eigen decomp $\quad C^{d\times d} = U \Lambda U^T$;

$$\lambda_j u_j = C u_j \quad (\text{obv})$$

4. Select

$$C_d = U \Lambda_d U^T = U_d \Lambda_d U_d^T \qquad d < D$$

5. Project

$$Y = Z U_d$$

6. Reconstruct $\quad Z_d = Y U_d^T = Z U_d U_d^T$

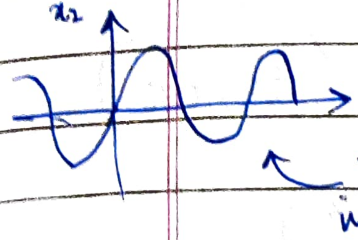
$$\text{Error} = \| \Lambda - \Lambda_d \| \propto \| Z - Z_d \|_2^2$$

Example : Eigen Faces (for face images)
7 PC of face images (7 only)

PAGE No.
DATE / /

(11-3) Dimension Reduction - Kernel PCA :

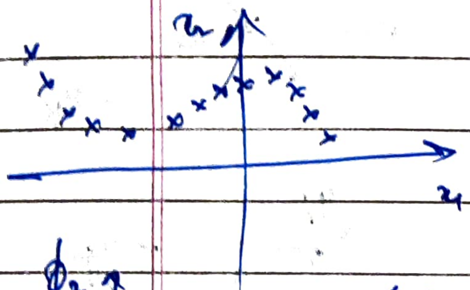
Sometimes data may lie on a non-linear manifold



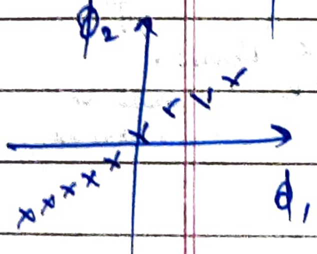
Data may lie on non linear manifolds and cannot use linear PCA to reduce the dimension well in this case

$\Rightarrow$ We need non linear techniques for dimension redn.

Kernel PCA introduces non linearity by mapping data to feature space



$x_i^T x \qquad \phi(x_i)^T \phi(x_j) \qquad K(x_i, x_j)$



✓ Infinite dimensions occur, but linear in one direction. $\Rightarrow$ Dimension reduction possible

✱ Avoids calculating features directly by using kernel trick

$C u_i = \lambda_i u_i \quad ; \quad C = Z^T Z / N$

We replace $Z^T Z$ with

$$\sum_n \phi(z_n)\, \phi(z_n)^T . u_i = N \lambda_i u_i$$

$$u_i = \sum_n a_{in} \phi(z_n) \qquad K_{ij} = \phi(z_i)^T \phi(z_j)$$

$$\sum_n K_{ln} \sum_m a_{im} K_{nm} = N \lambda_i \sum_n a_{in} K_{ln}$$

$$\Leftrightarrow \quad K^2 a_i = \lambda_i N K a_i \Rightarrow \boxed{K a_i = \lambda N a_i}$$

Essentially, we find Eigen Decomposition of the Kernel Matrix & the Evectors are $\{\vec{a_i}\}$

then the decomposition is in $N \times N$ space.

<u>Conditions:</u>

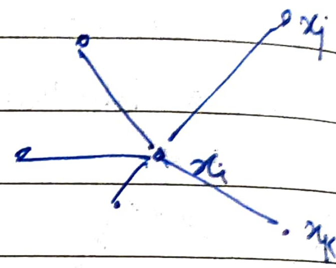
ϕ transformed vectors need to be mean centred.

<u>(11-4) Stochastic Neighbor Embedding</u> :- (t-SNE)

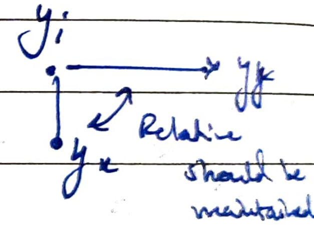
$$p_{j|i} = \text{softmax}\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)$$

$$q_{j|i} = \text{softmax}\left(-\|y_i - y_j\|^2\right)$$

$$\frac{e^{(1+\|y_i - y_j\|)^{-1}}}{\sum_k (\ldots)}$$



P and Q are $N \times N$ matrices (Stochastic as sum of rows = 1)



<u>KL divergence reduces:</u>

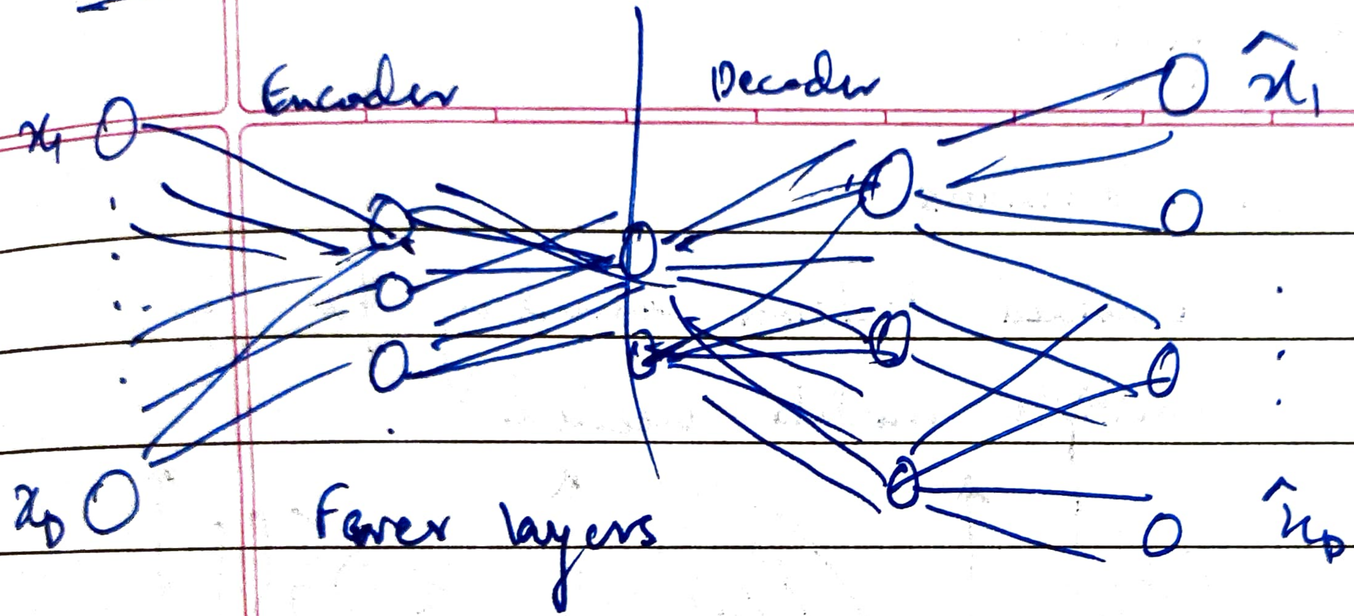
$$\sum_i \sum_j p_{j|i} \log\left(\frac{p_{j|i}}{q_{j|i}}\right)$$

Pick y such that minimized

For locally linear embedding methods, we want points to be dense because we want to connect them using edges and form a graph of the vertices & edges.
The graph gives unrolling directions.

In t-SNE we look at <u>all</u> point-point relation.
- Computationally expensive
- Hyperparameter tuning needed.

Auto-Encoders bottlenecks for dim red:



We wish to reduce error between $\hat{x}$ and x

Once trained, we get encoder and without losing MSE

$\Rightarrow$ Since activation fn of neurons are non linear, we introduce non linearity as well.