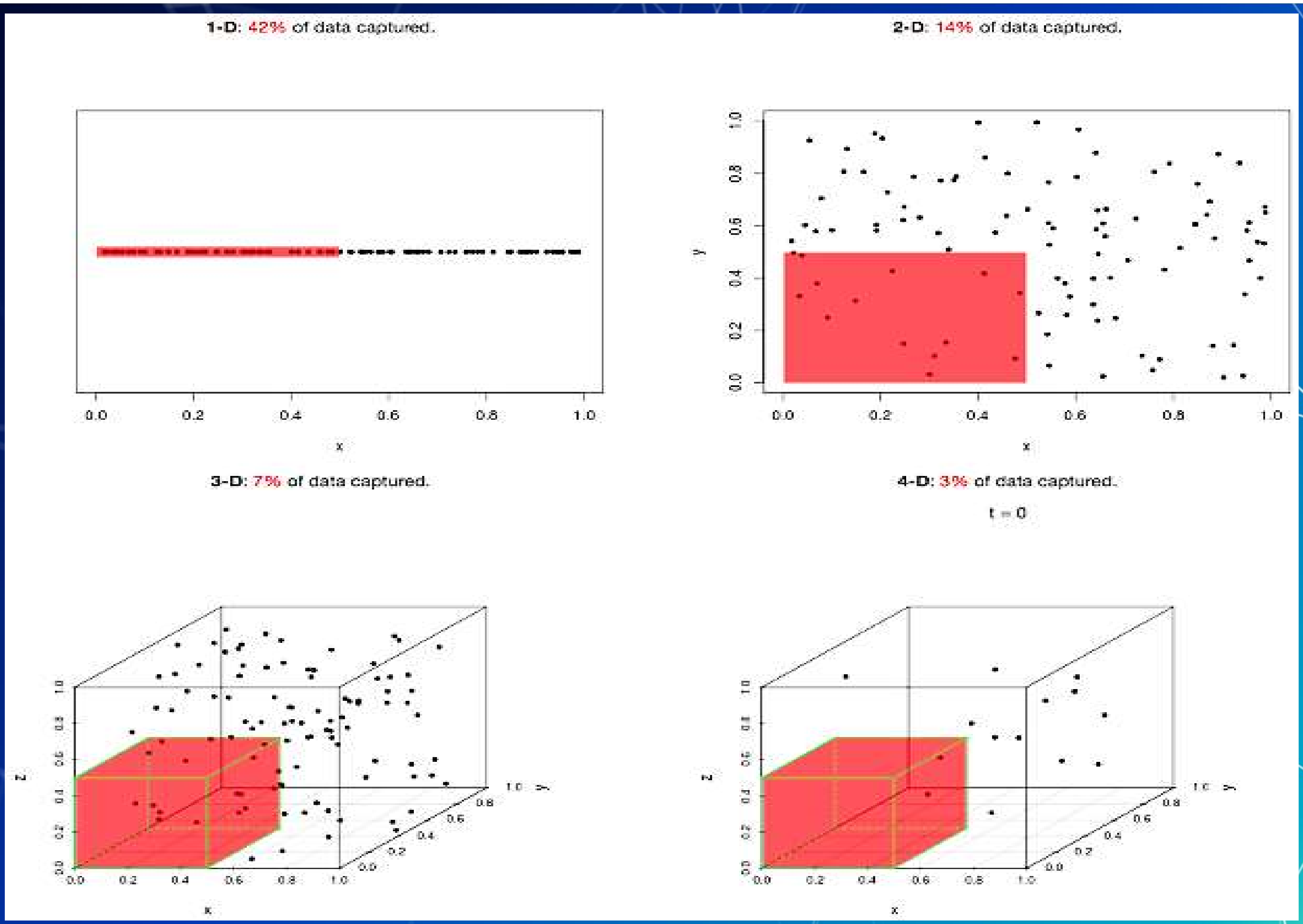
Principal Component Analysis and its applications

The Curse of Dimensionality

- Thousands or even millions of features for each instance of training data.

- Training is slower
- Harder to find a good solution

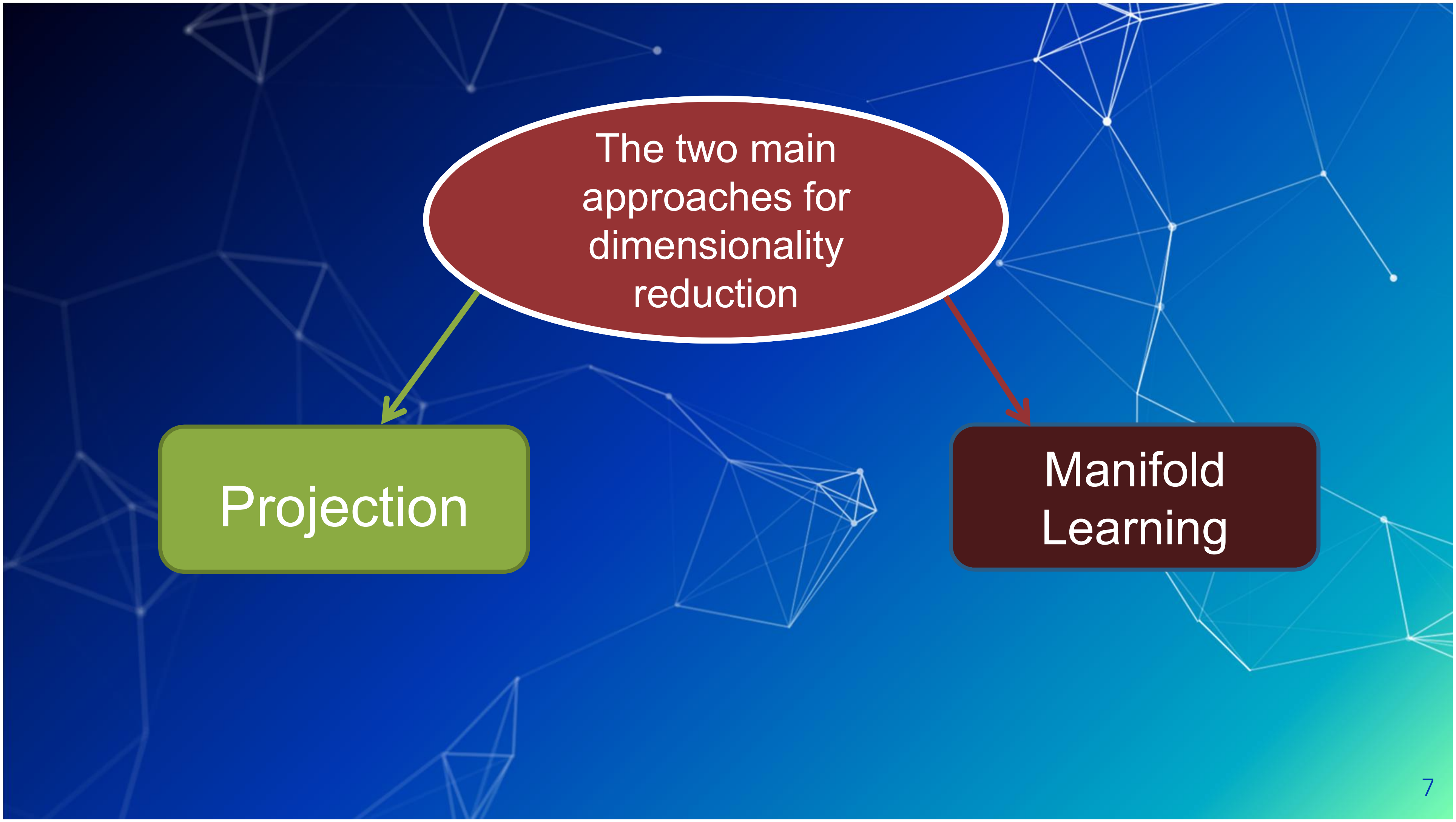
- Real world problems , It is possible to reduce the number of features cosiderably , without losing much information (e.g. MNIST data set)

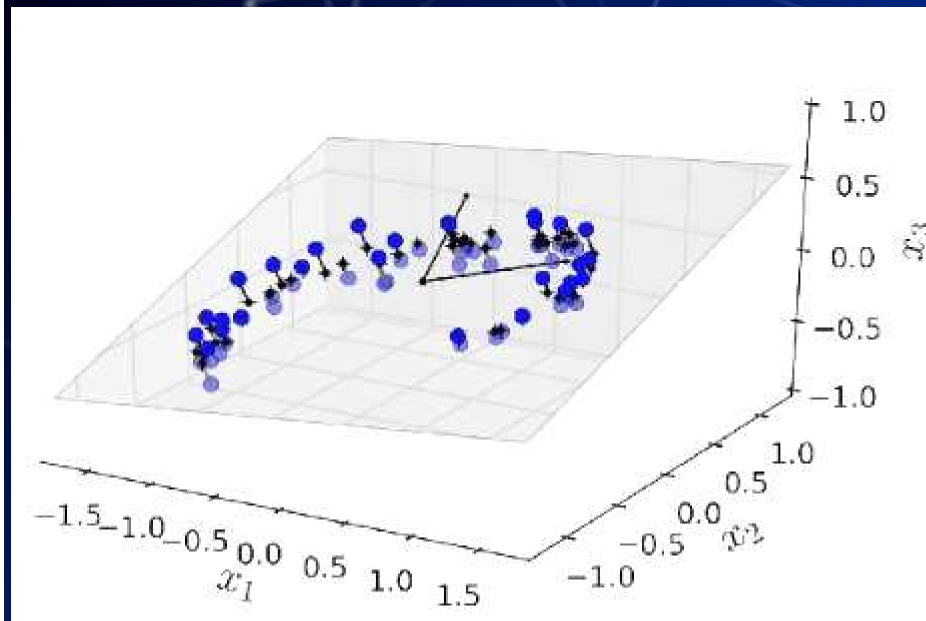

1-D: 42% of data captured.
2-D: 14% of data captured.
3-D: 7% of data captured.
4-D: 3% of data captured.
t = 0

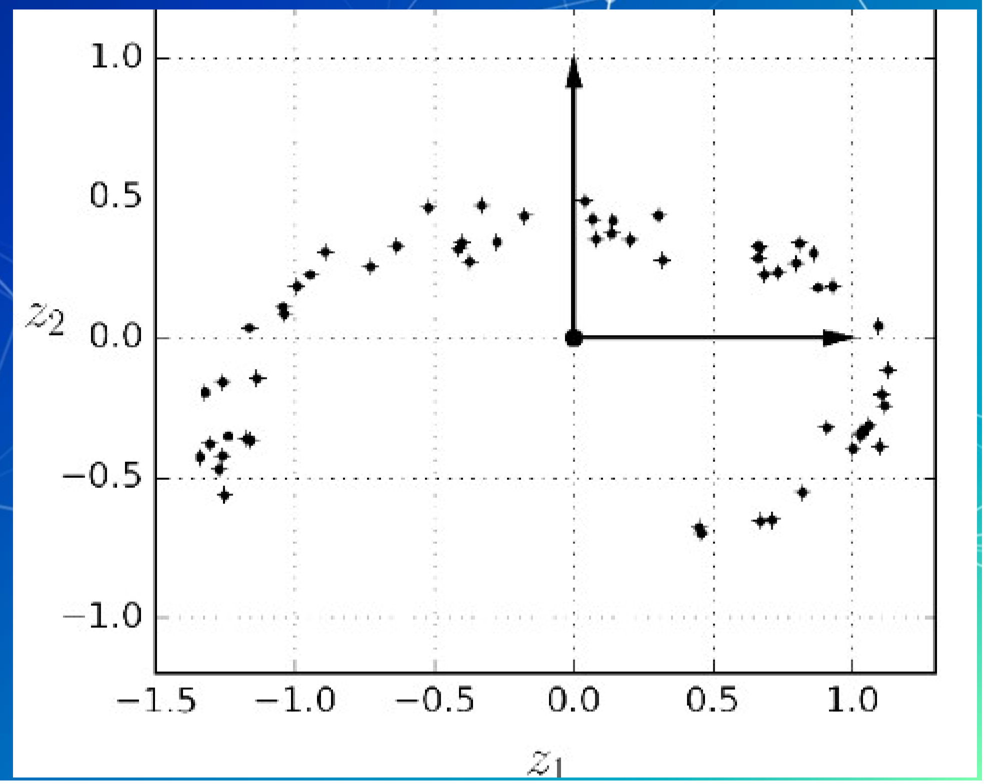
- The number of training instances required to reach a given density grows exponentially with number of dimensions.

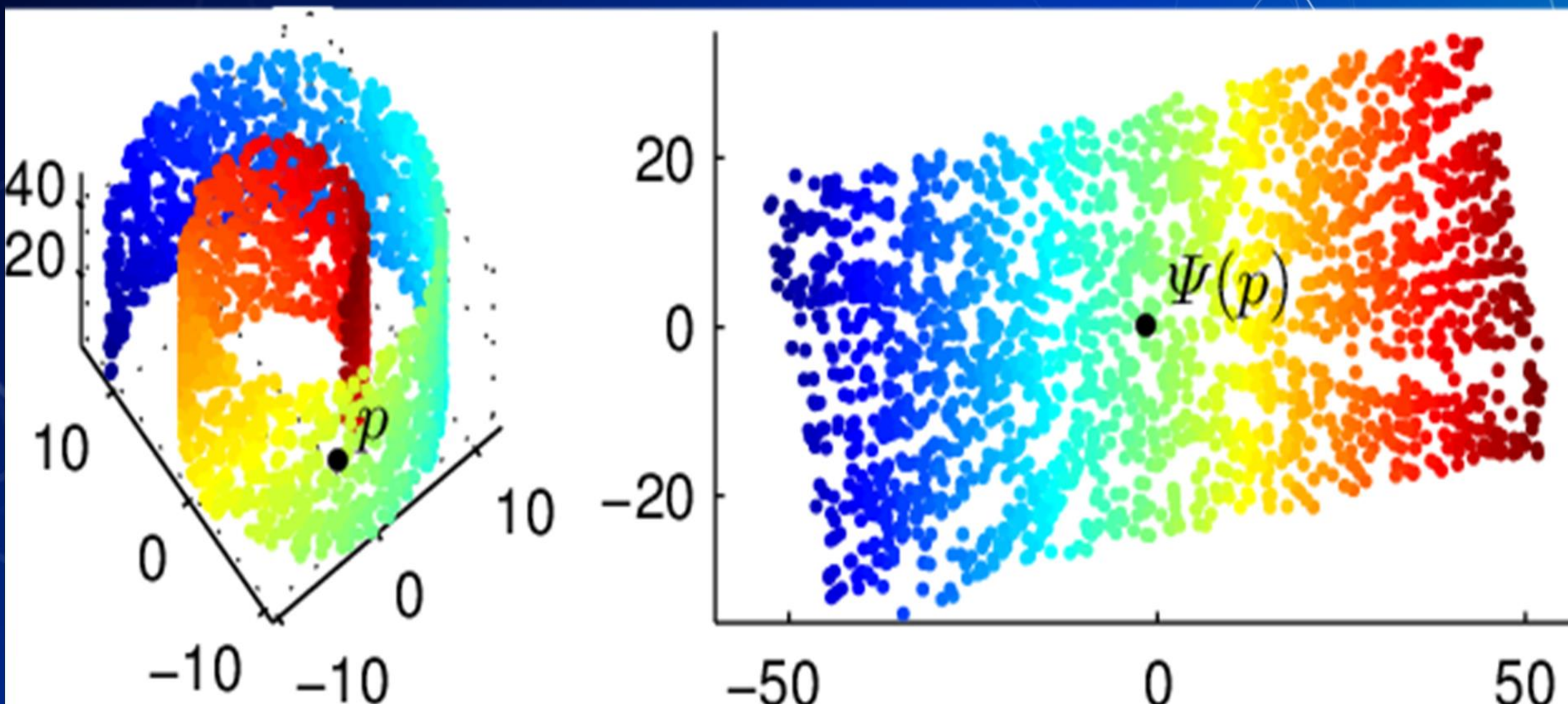
- More Dimensions = Greater Risk of Overfitting

Dimensionality Reduction


The two main approaches for dimensionality reduction
Projection
Manifold Learning


1.0
0.5
0.0
−0.5
−1.0
x_3
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x_1
−1.0 −0.5 0.0 0.5 1.0
x_2


1.0
0.5
0.0
−0.5
−1.0
z_2
−1.5 −1.0 −0.5 0.0 0.5 1.0
z_1

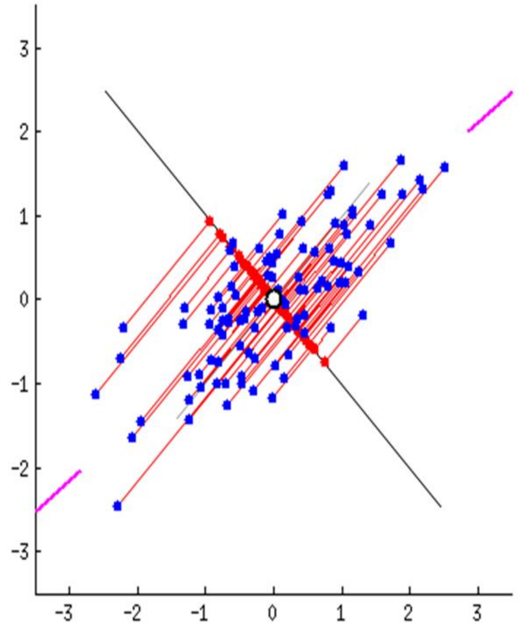

40
20
10
0
−10
−10
0
10
p
20
0
−20
−50
0
50
$\Psi(p)$

Principal component analysis

- PCA is by far most popular dimensionality reduction algorithm .
- PCA in simple terms:

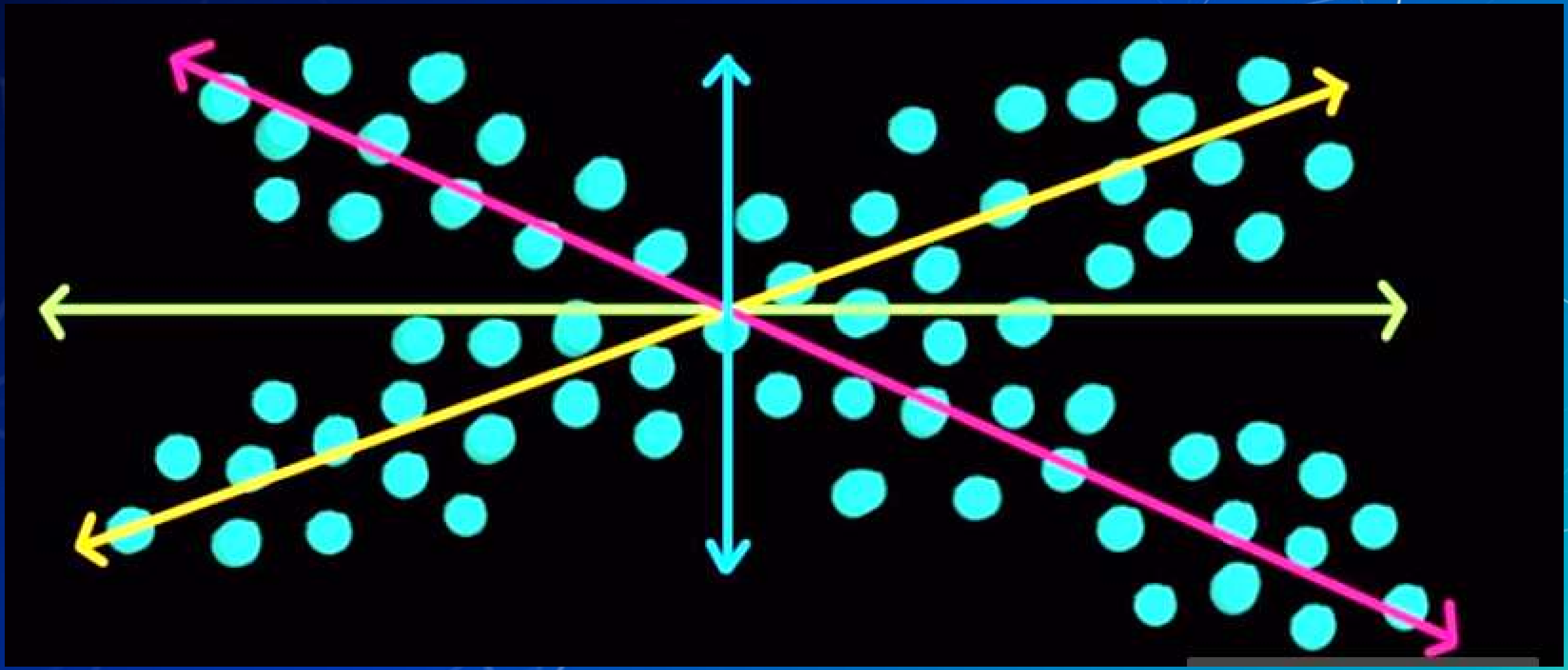
It finds the closest hyperplane to the data points and then it projects the data onto it .

Choosing the right Hyperplane



- Select the hyperplane that Preserves the maximum variance.
- PCA identifies the axis that preserves the maximum variance, and second axis orthogonal to first that accounts for largest remaining variance
- i^{th} Principal component

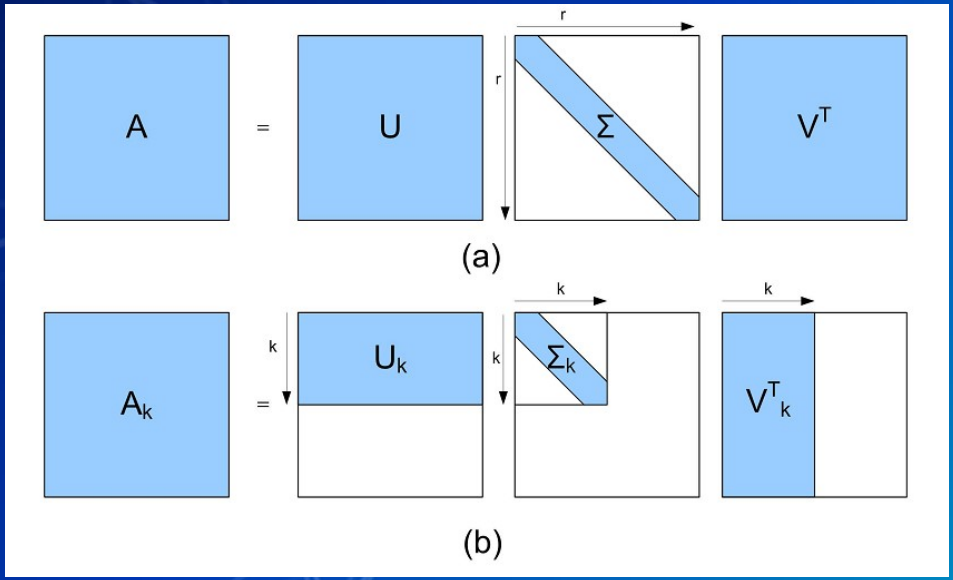
Guess the PCs for the given data points.



So how can we find the principal components of a training set ?

The Singular value decomposition

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



- The SVD separates any matrix into simple rank one pieces in the order of importance.
- $A = \sigma_1 u_1 {v_1}^T + \sigma_2 u_2 {v_2}^T + \ldots\ldots\ldots$
- The size of the singular values ($\sigma_i' s$) will decide whether to retain or ignore a value.
- Keep larger $\sigma_i' s$, discard smaller $\sigma_i' s$.
- The principal components are The orthogonal vectors that are retained.

Performing SVD

- The singular value theorem for A is Eigen value theorem for $A^T A$ and AA^T.
- $A \rightarrow A^T A \rightarrow$ Eigen values and vectors of $A^T A$
- The Eigen vectors of $A^T A$ are row entries of V^T
- The matrix V^T contains the principal components
- Now , using $AV = U\Sigma$ find U
- Or in simple terms $u_i = Av_i/\sigma_i$

What is the reduction in size ?

- Instead of transmitting or processing whole training data set A
- Just use the dominant linear combination of rank 1 matrices.

"

When compression is well done, you can't notice the difference between original and compressed versions

Performing PCA with SVD

- Data = n samples and m features per sample.
- Center each row of matrix A by subtracting mean from each measurement
- Then apply the SVD for covariance matrix
- Largest singular value (σ_1) → greatest variance → most information in u_1

PCA for face recognition

Eigen faces

Yale data base

The mean face

Eigen faces

Convert each N x N image to N^2 x 1 vector

$$[\Gamma_1 \quad \Gamma_2 \quad \dots\dots \quad \Gamma_M]$$

Calculate the Mean face from the Training data

$$\psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

$$\phi_i = \Gamma_i - \psi$$

$$\phi = [\phi_1 \quad \phi_2 \quad \dots\dots \quad \phi_M]$$

$$C = \frac{1}{M}\phi\phi^T$$

$$C = \frac{1}{M}\phi\phi^T$$

Find the Eigen values and vectors of covariance matrix

These dominant Eigen vectors(PCs) are the Eigen faces

But ....

For the covariance matrix the no of Eigen values and vectors are N^2 (too many)

Intractable solution

Computationally feasible method

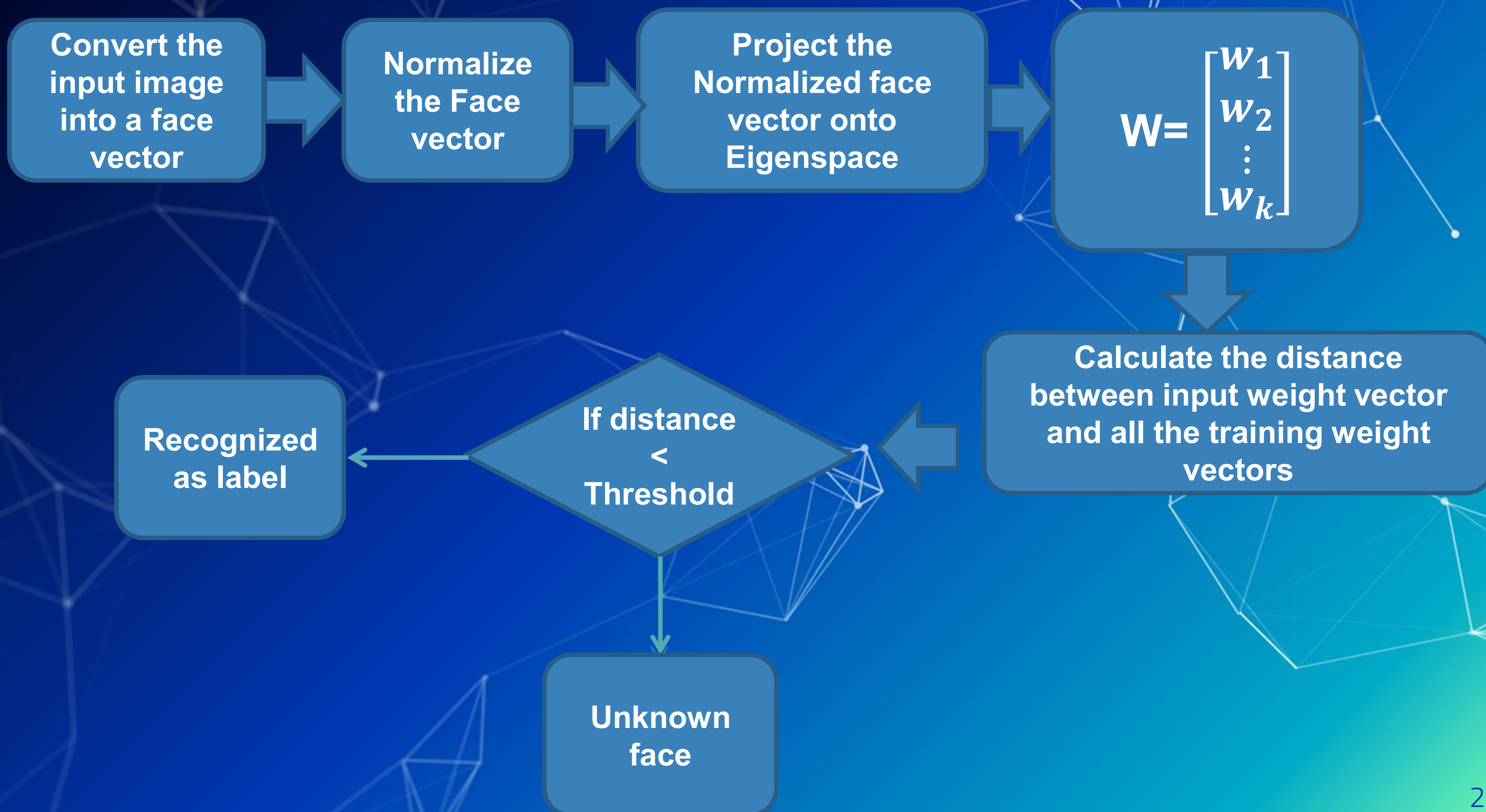
$$A = \frac{1}{M}\phi^T\phi$$

Find the Eigen values and vectors of matrix A, then using this find PCs

For the matrix A the no of Eigen values and vectors are M (far less than covariance matrix)

Using the matrix A and its Eigen vector, matrix it is easy to find the principal components u_is

$$AV = U\Sigma$$


Convert the input image into a face vector
Normalize the Face vector
Project the Normalized face vector onto Eigenspace
$W=\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}$
Calculate the distance between input weight vector and all the training weight vectors
If distance < Threshold
Recognized as label
Unknown face

References

- Matthew Turk and Alex Pentland, Eigenfaces for Recognition , 1991, journal of cognitive neuroscience.
- Gilbert Stang, Introduction to Linear Algebra
- Aurélien Géron , Hands on Machine learning with scikit learn and tensor flow
- Udacity course on Computer vision - Georgia Tech University

Thank you

Any Queries ?

"

$A^T A$ and $A A^T$, have the same non-zero eigenvalues, and if one has more eigenvalues than the other, then these are all equal to 0.