Dimensionality

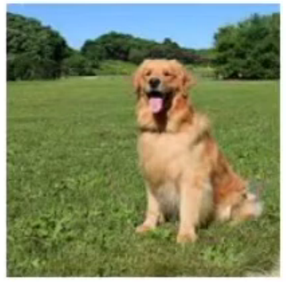
Curse of Dimensionality:

- Machine learning deals with data, as we record them, have millions and billions of dimensions.

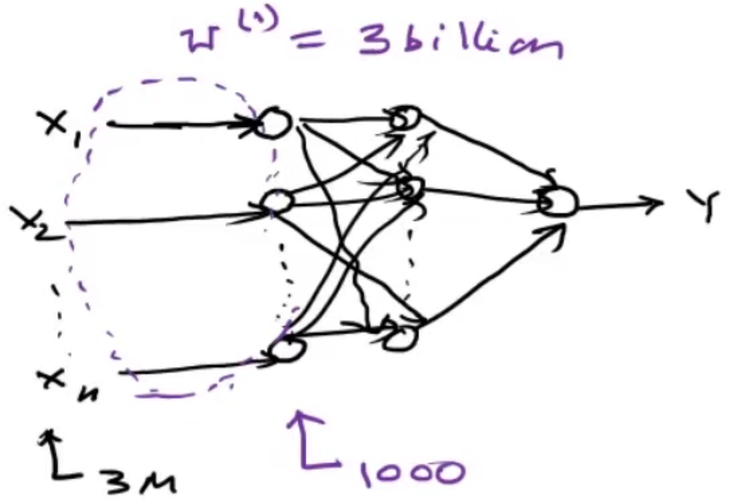
0,1 → 2^{20} → 400

  - Examples: hand written digits represented by 20X20 bit map

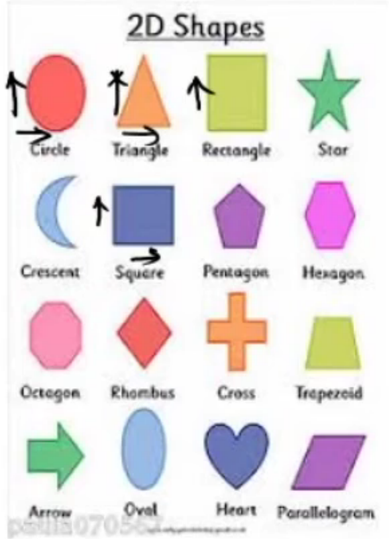
64x64x3 = 12288



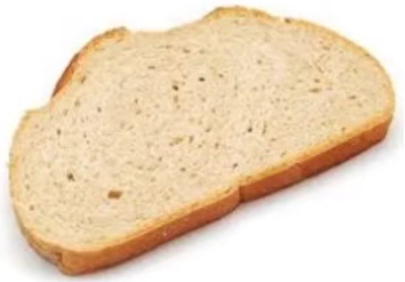
1024x1024x3 ≃ 3M

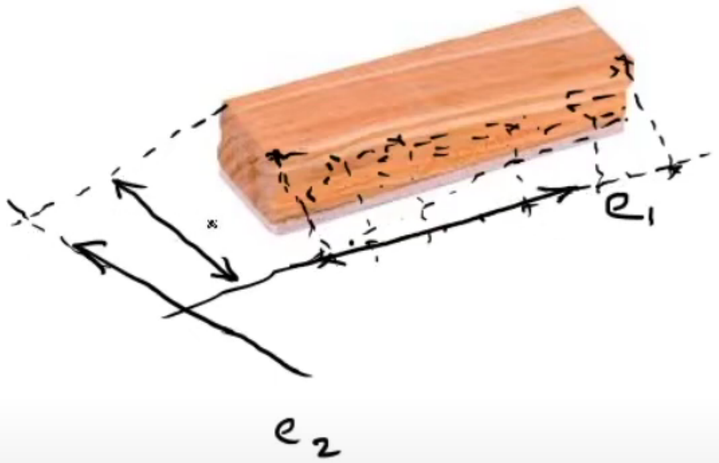


Dimensionality Reduction- Principal Component Analysis (PCA)

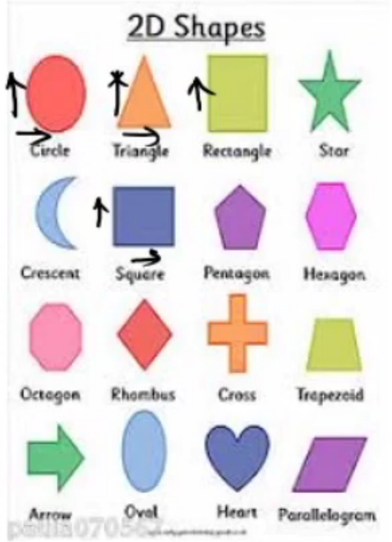


2 - D
→ 1 - D





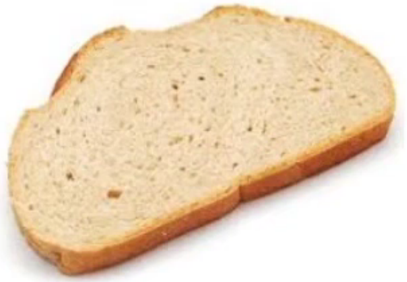
Dimensionality Reduction- Principal Component Analysis (PCA)

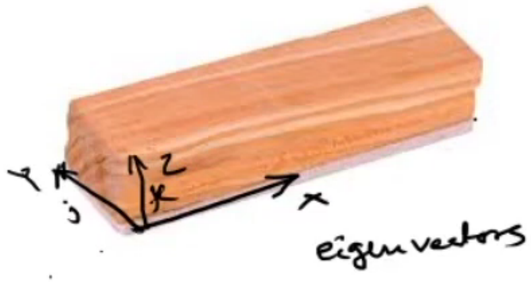


2-D
→ 1-D

64x64

8 → features





[1000x1000] $D_1 \rightarrow \lambda_1$ ↓ High

$D_2 \rightarrow \lambda_2$

$D_3 \rightarrow \lambda_3$

$D_{1000} \rightarrow \lambda_{1000}$

Advantages and disadvantages of PCA

Advantages:

- Removes correlated features and by doing so PCA improves the performance of ML algorithms
- It reduces over fitting

Disadvantages:

- Difficult to interpret the independent variables
- Data standardization is an issue
- Loss of information

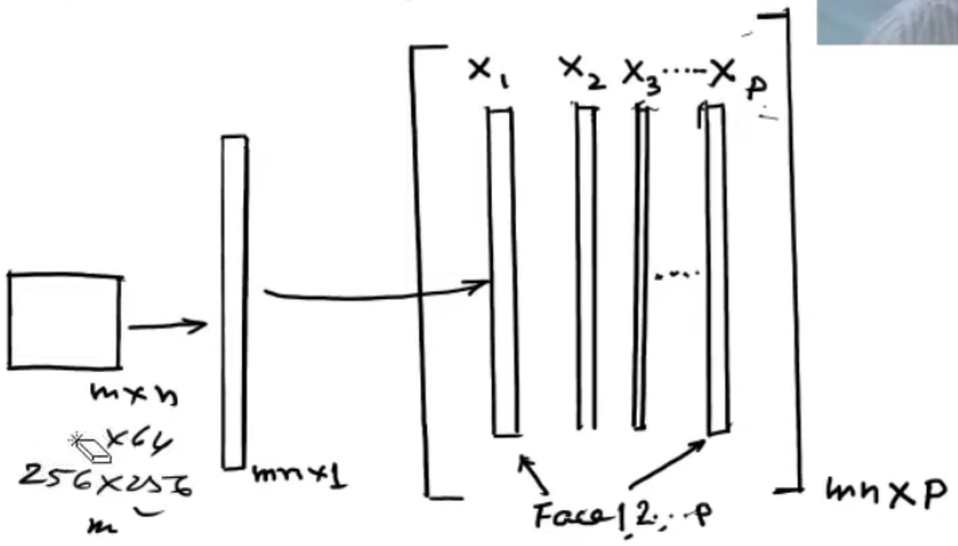
Applications of PCA

- It is a great data-dimensionality reduction tool in the following areas:
  - Facial Recognition.
  - Image compression.
  - Finding patterns in data of high dimension in the field of finance, bio informatics, psychology, social cybernetics.
  - Finding features from data for classifier design/regression analysis.

PCA Algorithm for face recognition

Steps:

1. Let us consider P training images of dimension **mXn**
2. Convert these images into vector of size **mnX1** →

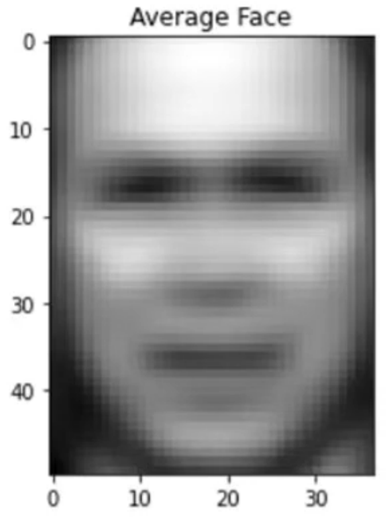


3. Mean Subtraction (Pre processing)
   Calculate the average of all these vectors and subtract it from each vector:

$$\mu = \frac{1}{P}\sum_{i=1}^{P} X_i \quad \text{——— (1)}$$

4. $W_i = X_i - \mu$ ——— (2) (zero mean)

W_i: $mn \times i$, X_i: $mn \times i$, μ: $mn \times i$



5. Take all the face vectors so that we get a matrix of size **mn X P**

$$W = [W_1\ W_2\ W_3 \cdots W_P]_{mn \times P} \quad \text{——— (3)}$$

6. **We calculate the Covariance matrix:** $\text{Cov} = \frac{\sum_{i=1}^{P} W_i W_i^T}{(P-1)} = W_{mn \times p} W^T_{p \times mn}$ — (4)

$(\text{Cov})_{mn \times mn} \longrightarrow$ Computational Problem.

In 1991 → Turk and Peterland suggested a method based on surrogate covariance.

Let us write $C = W^T_{p \times mn} \cdot W_{mn \times p} = (W^T W)_{p \times p}$ ——— (5)

7. Calculate Eigen value and Eigen vector of $C_{p \times p}$ → denote $(V)_{p \times p}$ — Eigen vector
$(\lambda)_{p \times p}$ — Eigen value.

8. Find best directions: Select K eigen value in descending order.
On the basis of K, generate feature vector $(\Psi)_{p \times K}$

9. Generate Eigen face (Φ) by projecting the mean-aligned face to the generated feature vector $(\Phi)_{K \times mn} = (\Psi)^T_{K \times p} \cdot W_{p \times mn}$

Construction

1. **Formula**:

$$\text{Cov}(X) = \frac{1}{P-1}\sum_{i=1}^{P}(w_i - \bar{w})(w_i - \bar{w})^T$$

Here, P is the number of observations, w_i represents the data vector, and $\bar{w}$ is the mean vector of the data. This formula shows how the covariance matrix is computed from the deviations of the data points from their mean.

2. **Matrix Form**:

$$C = W^T W$$

where W is a data matrix. This illustrates a computational approach to deriving the covariance matrix.

10. Generate signature of each face'.

For generating signature of each face (Ω), Project each mean aligned face to the Eigen faces.

$$(\Omega)_{K \times i} = (\Phi)_{K \times mn} * W_{mn \times i}$$

where $i \in 1, 2, \cdots P$. Hence Ω will be of size $K \times P$

Testing: 1. Let us have an image (I) → make it as a Column vector $(I)^1_{mn \times 1}$

2. Do zero mean → $(I)^2_{mn \times 1} = (I)^1_{mn \times 1} - \mu_{mn \times 1}$

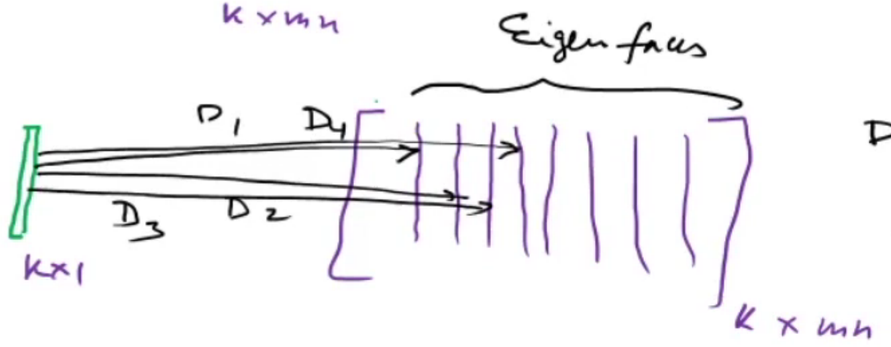
3. Project $(I)^2$ to eigen faces (Φ) to get projected test face (PTF)

$$(PTF)_{K \times 1} = \Phi_{K \times mn} * (I)^2_{mn \times 1}$$

4. Now we have projected test face (PTF) and signature of each face

5. Calculate the Eclidian distance between $(PTF)_{K\times 1}$ and

Each column of Φ : $K \times mn$



$D_1, D_2, D_3 \cdots D_K$

Minimum distance is the matching class.

Limitation:

- Proper centered face is required for training/testing.
- Sensitive to illumination and scale of face
- Front view of the face is required.

Recapitulation of basics of mean and standard deviation

Say we have two sets of samples X_i's

1) 0, 8, 12, 20 $\Rightarrow \bar{X}_1 = 10$

2) 8, 9, 11, 12 $\Rightarrow \bar{X}_2 = 10$

Standard deviation (S.D): $\sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}$

S.D for set-1 samples → 0, 8, 12, 20 $\Rightarrow$

X	$X-\bar{X}$	$(X-\bar{X})^2$
0	−10	100
8	−2	4
12	2	4
20	10	100
		Σ 208

$\Rightarrow$ S.D $\sqrt{\frac{208}{3}} = 8.3 = 5$

S.D for set-2 samples → 8, 9, 11, 12 $\Rightarrow$ S.D = 1.8 = 5

Concept of variance → Make just s^2 to accommodate higher dimensions of samples

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

Recapitulation of basics of mean, standard deviation and co-variance

For two dimensional data co-variance can be calculated as :

$$Cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$ and you may observe $Cov(X,Y) = Cov(Y,X)$

When the dimensionality of data is large then you can't visualize the data but your covariance matrix can help you in finding the pattern hidden inside the data.

For three dimensions the matrix will look like:

$$C = \begin{pmatrix} Cov(X,X) & Cov(X,Y) & Cov(X,Z) \\ Cov(Y,X) & Cov(Y,Y) & Cov(Y,Z) \\ Cov(Z,X) & Cov(Z,Y) & Cov(Z,Z) \end{pmatrix}$$

For n dimensional data $C \in \mathbb{R}^{n \times n} \rightarrow (C_{ij}, C_{ij} = Cov(Dim_i, Dim_j))$

Dim_x is the x^{th} dimension

Recap of fundamentals of statistics

i.e, if you have an n-dimensional data set, then the matrix has n-rows and n-columns and each entry in the matrix is the covariance between two seperate dimensions:

$$C = \begin{pmatrix} Cov(1,1) & Cov(1,2) & Cov(1,3) & \cdots & Cov(1,n) \\ Cov(2,1) & Cov(2,2) & Cov(2,3) & \cdots & Cov(2,n) \\ \vdots & \vdots & \vdots & & \\ Cov(n,1) & Cov(n,2) & Cov(n,3) & \cdots & Cov(n,n) \end{pmatrix}_{n \times n}$$

Concept of Eigenvectors and Eigenvalues:

Let us consider any matrix → $\begin{pmatrix} 3 & 4 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 11 \\ 4 \end{pmatrix}$

$\Rightarrow \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ is not an Eigen vector

- Eigen vectors can only be found for the square matrices.
- Not every square matrix (nXn) has eigen vectors, but if they have, then there will be n.
- All the eigen vectors are orthogonal to each other.

$\begin{pmatrix} 3 & 4 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 10 \\ 5 \end{pmatrix} = 5 \begin{pmatrix} 2 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ is an Eigen vector.

5 is the eigen value.

PCA Algorithm

Step -1 Get Some Data and calculate the mean, $\mu = \frac{\sum_{i=1}^{n} x_i}{n}$

Step-2 Subtract the mean from the data and form new data, PCA will work properly, if we subtract mean from each of the data dimension.

Step-3 Calculate the Covariance matrix

Step-4 Calculate the eigenvectors and eigenvalues of the covariance matrix

Step-5 Estimating high valued eigen vectors and forming a feature vector

- Arrange the eigen values, λ_i in the descending order
- Choose a threshold, θ
- Number of high-valued λ_i can be chosen so as to satisfy the relationship:

$$\left(\sum_{i=1}^{s} \lambda_i\right)\left(\sum_{i=1}^{n} \lambda_i\right)^{-1} \geq \theta,$$ where s is the number of high valued λ_i chosen

- Select eigen vectors corresponding to selected high valued λ_i

Step-6 Deriving the new data set.

PCA example: Eigen Faces