Principal Component Analysis (PCA)

→ It is a dimensionality reduction.

→ n-dim → d'-dimen
$x_i \in \mathbb{R}^d$ $d' < d$.

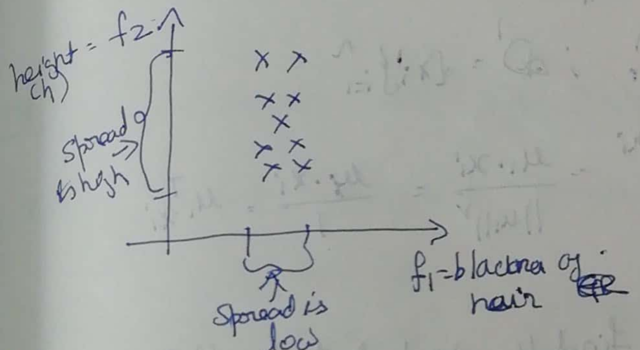
| mnist → 784 dim → 2-dim (visualize)

application

① To Visualize

② d-dim → d'-dim. $(d'=10)$
$d' < d$

14.2 Geometric Intuition of PCA



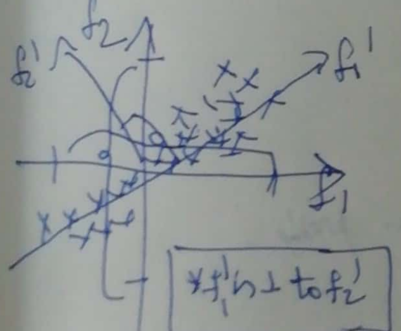
* The spread on f_2 axes is very high when compared to f_2
* Spread is Variance
* If i am force to skip data we can skip f_1 and keep f_2 as there more spread in f_2

$$\underset{n\times 2}{X} = \begin{array}{c|cc} & f_1 & f_2 \\ \hline 1 & & \\ 2 & & \\ 3 & & \\ \vdots & & \\ n & & \end{array} \quad ; \quad \underset{n\times 1}{x'} = \begin{array}{c|c} & f_2 \\ \hline 1 & \\ 2 & \\ \vdots & \\ n & \end{array}$$

* I am preserving the direction with maximal spread/variance
→ more spread more information.

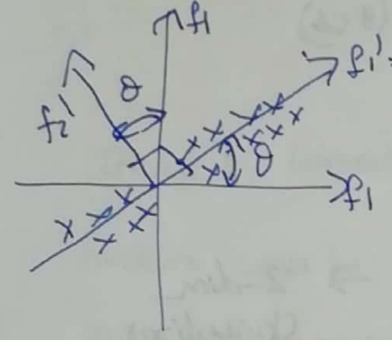
Ex-2 : 2-dimen dataset, both collum are standardize

$\{$ mean $\{f_1\}$ = mean $\{f_2\} = 0$
Var $\{f_1\}$ - var $\{f_1\} = 1$



* Enough spread on both the axis
* In the direction f_1' there is lot of spread
* Spread on f_2' << Spread on f_1'
* drop f_2' and project on to f_1'

2D → 1D



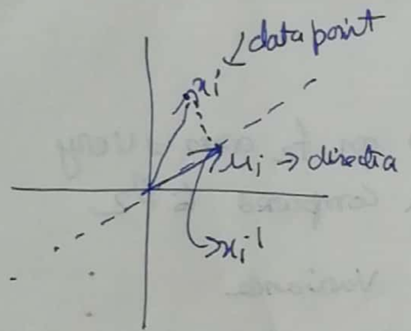
* Rotated f_1' with some θ & with the same θ Rotate f_2'

* f_1' has maximum spread, we drop f_2' and project on to f_1'

* we want to find direction f_1' such that the variance of x_i's projected on to f_1' is maximum.

* we are most intrested in finding the direction on the line f_1'

* we represent direction is u_1 $\quad \|u_1\| = 1$



* $x_i' = \text{proj}_{u_1} x_i$

$\mathcal{D} = \{x_i\}_{i=1}^{n}$; $\mathcal{D}' = \{x_i'\}_{i=1}^{n}$

$$x_i' = \text{project}_{u_1} x_i = \frac{u_1 \cdot x_i}{\|u_1\|^2} = \frac{u_1 \cdot x_i}{1} = u_1^T x_i$$

$x_i' = u_1^T \bar{x}_i$ → mean vector $\{x_i\}_{i=1}^{n}$

$\{x_i\}_{i=1}^{n}$ → mean vector

* find u_1 such that the variance of u_1 projected onto x_i is maximal

$$\text{Var}\{u_1^T x_i\}_{i=1}^{n} = \frac{1}{n}\sum_{i=1}^{n} (\underbrace{u_1^T x_i}_{(1\times n)(n\times 1) \text{ column vector}} - \underbrace{u_1^T \bar{x}}_{0})^2$$

X: col. standardize ; $\bar{x} = \{0, 0, 0, 0\}$

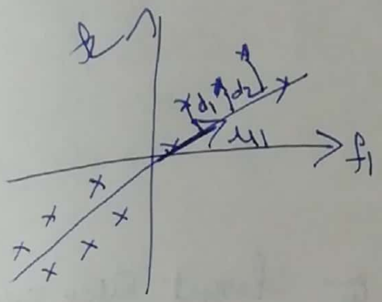
$$\text{Var}\{x_i'\}_{i=1}^{n} = \frac{1}{n}\sum_{i=1}^{n} (u_1^T x_i)^2$$

$$\boxed{\max_{u_1} \underbrace{\frac{1}{n}\sum_{i=1}^{n} (u_1^T x_i)^2}_{\text{Var}\{x_i'\}}}$$ ← objective of an optimization problem

→ such that $u_1^T u_1 = 1 = \|u_1\|^2$

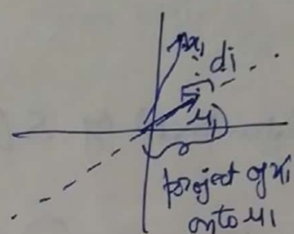
14.4 Alternative formulation of PCA: Distance minimization.

→ find u_1 which maximaizes projected variance



$x_i \to d_i$: dist from x_i to u_1

$$\min_{u_1} \sum_{i=1}^{2} d_i^2$$



u_1 = unit vector

$u_1^T u_1 = 1 = \|u_1\|^2$

project of x_i onto u_1 $proj_{u_1} x_i$

$$\min_{u_1} \sum_{i=1}^{n} \left(x_i^T x_i - (u_1^T x_i)^2\right)$$

such that $u^T u = 1$

$\|x_i\|$, d_i, u_1 : $d_i^2 = \|x_i\|^2 - (u_1^T x_i)^2$

$= x_i^T x_i - (u_1^T x_i)^2$

14.5 Eigen Values and Eigen vectors (PCA): dimensionality Reduction.

solution for our optimization problem (max & min)

$$X = \begin{array}{c|c} & 1, 2, 3 \cdots d \\ \begin{matrix} 1 \\ 2 \\ 3 \\ \vdots \\ n \end{matrix} & \left[\begin{matrix} \\ \\ \\ \\ \end{matrix} \right] \end{array}_{n \times d}$$

col std

$\mu = 0$

$\sigma = 1$

Covariance matrix of $X = S$

$$\underset{d \times d}{S} = \underset{d \times n}{X^T} \underset{n \times d}{X}$$

sq. symmetric.

Eigen values $(\lambda_1, \lambda_2 \cdots \lambda_d)$

Eign vector $(V_1, V_2 \cdots V_d)$

maximum Eigen-value

$\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \lambda_d$

$S_{d \times d}$ → Eigen values of $S = \lambda_1, \lambda_2, \lambda_3 \cdots \lambda_d$

→ Eigen vector of $S = V_1 \; V_2 \; V_3 \cdots V_d$

* Every Eigen Value there is corresponding Eigen Vector

definition $\lambda_1 V_1 = \underset{d \times d}{S} V_1$; V_1 → $d \times 1$ vector

scalar

$d \times 1$ vector

λ_1 : Eigen Value of S

V_1 : Eigen vector of S corresponding to λ_1

If $\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \lambda_d$ for matrix $S_{d \times d}$

$$\begin{array}{cccc} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_d \\ \downarrow & \downarrow & \downarrow & \downarrow \\ v_1 & v_2 & v_3 & v_d \end{array}$$

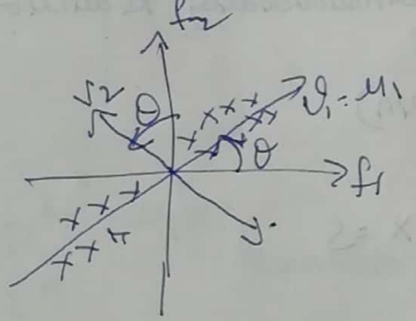
$v_i \perp v_j$: $v_i^T v_j = 0$ = $v_i \cdot v_j = 0$

Every pair of Eigen vector are $\perp$ to Each other.

$u_1 = v_1$ = Eigen vector of $S (= X^T X)$ corresponding to largest Eigen value $(= \lambda_1)$

$X = \begin{bmatrix} \ & \ \\ \ & \ \end{bmatrix}$

1. col. std of X is done
2. $S = X^T X$
3. Eigen value & vector of S
   $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_d$
   $v_1, v_2 \cdots v_d$
4. $u_1 = v_1$ (why?)



2dim
$d = 2$
$\lambda_1 \geq \lambda_2$
$v_1 \perp v_2$

$x_i \in \mathbb{R}^{10}$ $d = 10$

I will have 10 Eigen value & vector

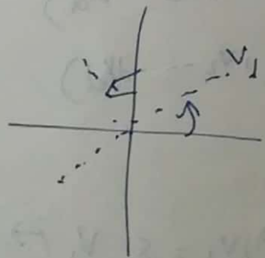
$$\begin{array}{cccc} \lambda_1 \geq \lambda_2 \cdots & \lambda_3 & \cdots \geq & \lambda_{10} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ v_1 & v_2 & v_3 & v_{10} \end{array}$$

v_1: direction with max-variance

v_2: 2nd direct with most variance

v_3: 3rd maximal variance

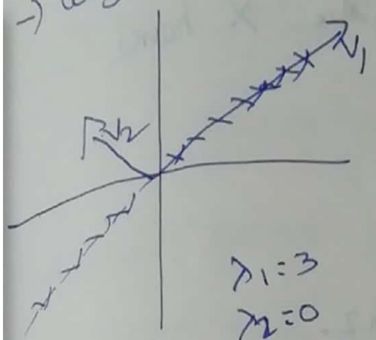
v_{10}: least variance in the direction of v_{10}



What are λ_i's ? $(\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \geq \lambda_d)$

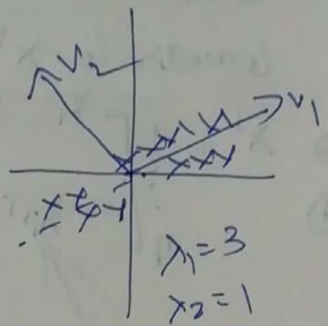
$\sum \lambda_i$ $\sum_{i=1}$

→ Why λ_i's



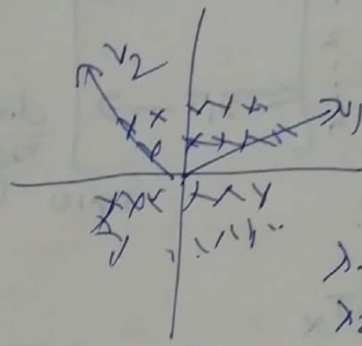
$\lambda_1 = 3$
$\lambda_2 = 0$

Perfectly aligned with V_1 $\frac{3}{3} = 1$



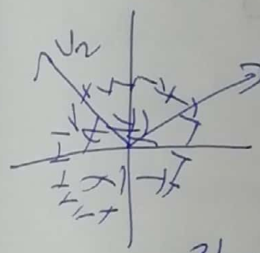
$\lambda_1 = 3$
$\lambda_2 = 1$

There is some spread. $= 3/4 = 75\%$



$\lambda_1 = 3$
$\lambda_2 = 2$

Then more spread. $3/5 = 60\%$



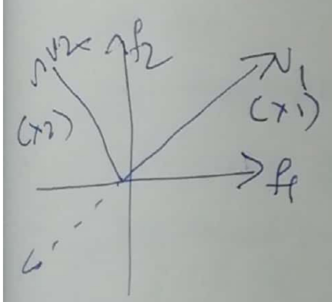
$\lambda_1 = 3$
$\lambda_2 = 3$

$3/6 = 50\%$

$$\frac{\lambda_1}{\lambda_1 + \lambda_2} = 1$$ (% of variance explained)

$$\frac{\lambda_i}{\sum_{i=1}^{d} \lambda_i}$$ (% of variance of explained / Retained by that direction)

14.6 PCA for dimensionality Reduction and Visualization



$$X = \begin{array}{c c} & \begin{matrix} f_1 & f_2 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ \vdots \\ n \end{matrix} & \begin{bmatrix} \\ \leftarrow x_i^T \rightarrow \\ \\ \end{bmatrix} \end{array}$$

$\downarrow$

1 − D

$S = X^T X$

Instead of projection data onto ~~f_1~~ f_1 we are projecting onto V_1 to Explain with more variance.

$$X' = \begin{array}{c c} & V_1 \\ \begin{matrix} 1 \\ 2 \\ \vdots \\ n \end{matrix} & \begin{bmatrix} \\ x_i' \\ \\ \end{bmatrix} \end{array}$$

$x_i' = x_i^T v_1$

(2D)

$\downarrow$ maximum Variance method.

(1D)

$$X = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ n \end{array} \begin{array}{c} f_1 \ f_2 \cdots f_{10} \\ \begin{bmatrix} \\ \\ \leftarrow x_i^T \rightarrow \\ \\ \end{bmatrix} \end{array} \xrightarrow[\text{redu}]{\text{dim}}$$

$n \times 10$ (PCA)

* I want to visualize X hence converting to X'

$$X' = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ n \end{array} \begin{array}{c} v_1 \quad v_2 \\ \begin{bmatrix} \\ \\ \leftarrow x_i'^T \rightarrow \\ \\ \end{bmatrix} \end{array}$$

$n \times 2$

$S = X^T X$

$\text{eigen}(S) = \lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \geq \lambda_{10}$

$\quad\quad\quad\quad\quad\quad v_1 \quad v_2 \quad v_3 \quad\quad v_{10}$

$x_i' = [x_i^T v_1, \ x_i^T v_2]$

$\quad\quad (v_1) \quad\quad (v_2)$

Ex:- $x_i \in \mathbb{R}^{100}$; $x_i' \in \mathbb{R}^{d'}$ $\quad d' < 100$

* Preserve 99% of the variance.

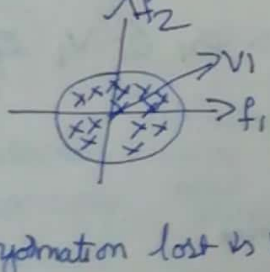
Let $\dfrac{\lambda_1 + \lambda_2 + \cdots \lambda_{51}}{\sum_{i=1}^{100} \lambda_i} = 0.99$

14.7 Visualize m-NIST dataset

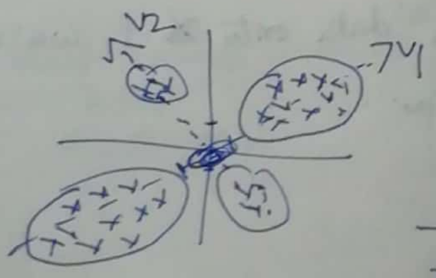
MNIST $\rightarrow$ 780 $\xrightarrow{\text{PCA}}$ 2 (v_1, v_2)

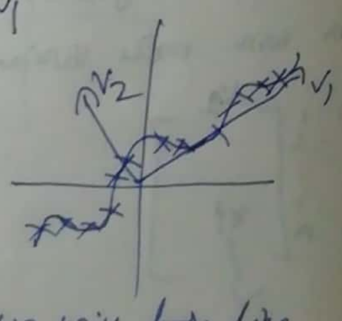
$y_i \in \{0, 1, 2, \cdots 9\}$

14.8 Limitations of PCA



Information lost is very high projected on to v_1





* we will look like structure when we project to v_1

14.9 PCA Code Example

2D Visualization using PCA

```
# Load MNIST Data

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
os.chdir('        ')

d0 = pd.read_csv('mnist_train.csv')
l = d0['label']
d0 = d0.drop('label', axis=1)
print(l.shape)          # (42000,)
print(d0.shape)         # (42000, 784)

# plotting one row in the dataset

plt.figure(figsize=(7, 7))
idx=100
grid_data = d0.iloc[idx].as_matrix().reshape(28, 28)
plt.imshow(grid_data, interpolation='none', cmap='gray')
plt.show()

#2D Visualization using PCA
# pick first 15K data-points to work on for time-efficiency
# Exercise: Perform the same analysis on all the 42K data-points
labels = l.head(15000)
data = d.head(15000)
print("The shape of sample data = ", data.shape)
```

Data Preprocessing: Standardizing the data

```
from sklearn.preprocessing import StandardScaler
standardized_data = StandardScaler().fit_transform(data)
print(standardized_data.shape)        # (15000,784)
```

~~#~~. Scaling results in a list

find The Co-Variance matrix which is : AT * A

```
sample_data = standardized_data
covar_matrix = np.matmul(sample_data.T, sample_data)
print(covar_matrix.shape)        # 784,784
```

finding the top two eigen-values and corresponding eigen vectors
for projecting onto 2-Dim space

```
from scipy.linalg import eigh
```

the parameter 'eigvals' is defined (low value to high value)
eigh function will return the eigen values in ascending order
this code generates only the top2 (782 & 783) Eigen values

```
values, vectors = eigh(covar_matrix, eigvals=(782,783))
print("shape of eigen vectors =", vectors.shape) #(784, 2)
```

Converting the eigen vectors into (2,d) shape for easyness of further

```
vectors = vectors.T
print("updated shape of eigen vectors =", vectors.shape)  # (2, 784)
```

here the vectors[1] represent the eigen vector corresponding 1st Principal Component

here the vectors[0] represent the eigen vector corresponding 2nd P.C

Projecting the original data sample on the plane
formed by two principal eigen vectors by vector-vector multiplication

```
import matplotlib.pyplot as plt
new_coordinates = np.matmul (vectors, sample_data.T)
print ("resultant new data points' shape", vector.shape, "x",
        sample_data.T, "=", new_coordinates)
```

resultant new data point's shape (2, 784) x (784, 15000) = (2x15000)

```
import pandas as pd
# appending label to 2d projected data
new_coordinates = np.vstack ((new_coordinates, labels)).T
# creating a new data frame for ploting the labeled points
dataframe = pd.DataFrame (data = new_coordinates, colums = ('1st_pri', '2nd_pri', 'label'))
datframe.shape   # (15000,3)
```

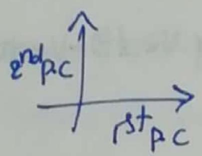
plotting the 2d data points with seaborn.

```
import seaborn as sn
sn.FacetGrid (dataframe, hue = 'label', size = 6). map (~~plt~~ plt.scatter, '1st_prin', '2nd_pri').
                                                      add_legend ()
```

PCA using Scikit-Learn

```
from sklearn import decomposition
pca = decomposition.PCA ()
# configuring the parameters   # the # of components = 2
pca.n_components = 2
pca_data = pca.fit_transform (sample_data)
print ("shape of pca_reduced.shape =", pca_data.shape)        # (15000,2)
```

14.10 PCA for dimensionality reduction (not-Visualization)

PCA : 784 → 2 

→ Visualization.

$784 \xrightarrow{PCA} 10$ → ML-models

Ex: $784 \xrightarrow{PCA}$ 200 dim

$X_{15000 \times 784}$ $V_{784 \times 200}$ $Cov = X^T X$

What is the right demensions (10 & 20 & 50 & 100 & 200 & 500 & 700)

PCA :- maximize the variance of projected points

784 → 10 → How much of original variance is Explained (784 → 10)

PCA for dimensionality reduction

```
Pca.n_components = 784

Pca data = pca.fit_transform(sample_data)

percentage_Var_Explained = pca.Explained variance_/np.sum(pca.Explained_variance_)

Cum_Var_Explained = np.cumsum(percentage_Var_Explained)

# plot the PCA spectrum

plt.figure(1, figsize=(6,4))

plt.clf()

plt.plot(cum var Explained, linewidth=2)

plt.axis('tight')

plt.grid()

plt.xlabel('n components')

plt.ylabel('cumulative_Exp_Variance') → plt.show()
```