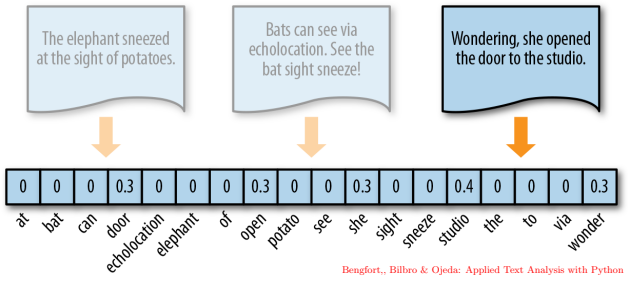
Principal Component Analysis

- Imdadullah Khan

High Dimensional Data

High Dimensional Data is common in many applications

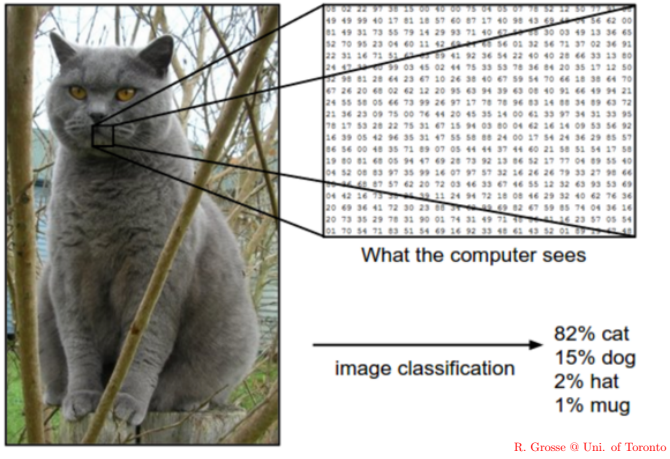
VSM representation of text



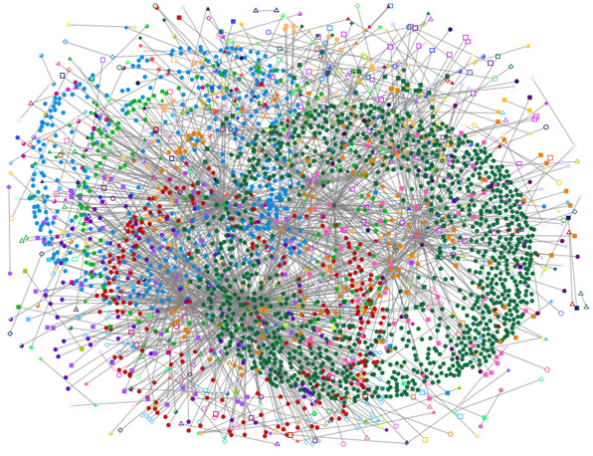
Utility matrix for recommenders

	p_1	p_2	p_3					p_j								p_m
u_1	1		2	1		4			2		3	2		5				2
u_2		1				2		1			2			1				3
u_3		1	1	2				1							1		2	
				3		2			5			2			3	4		
		1				2								5				
u_i			3	2	1		4	5	?	1		3	1		2			1
			4													4		
			5			1										5		
		1		4							1	3		5		1	2	
u_n			3		1	1		2	1					4				5

Multi-mega pixels images



Social networks as adjacency matrix

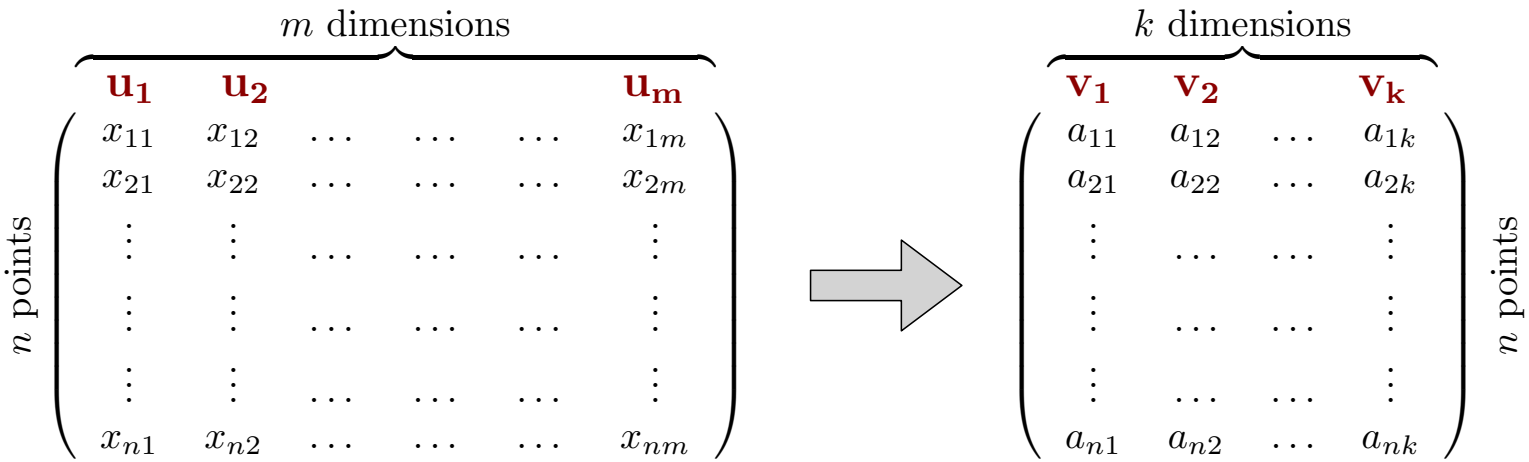


Curse of dimensionality

- In general as features increase redundancy also increases
  - more noise added to data than signal
- High dimensional data is hard to visualize and interpret
- Computationally challenging
  - Processing time
  - Storage capacity
  - Communication bandwidth
- Distance/Angle Concentration
- Nearest Neighbor Instability

Dimensionality Reduction

- The focus of dimensionality reduction through PCA are
- High dimensional data is hard to visualize and interpret
- The computational aspect of the curse
  - Processing time
  - Storage capacity
  - Communication bandwidth
- Our goal: reduce dimensionality of the dataset with low 'distortion'



Aims of PCA

- Low dimensional data visualization
- Get a sense of source of variations in data
- Understand pairwise correlation between attributes of data
- Reduce dimensions with little *'distortion'*

PCA vs JL-Transform

1. JL transform is data oblivious
   - Does not utilize structure in data
   - Computationally efficient
   - Can pre-compute transformation

1. PCA is data dependent
   - Fully exploits structure in data
   - Computationally expensive
   - Cannot pre-compute

PCA vs JL-Transform

1. JL transform is data oblivious
2. JL transform is randomized
   - Projects data onto random vectors

1. PCA is data dependent
2. PCA is deterministic
   - Projects onto vectors computed based on data

PCA vs JL-Transform

1. JL transform is data oblivious
2. JL transform is randomized
3. JL transform preserves pairwise distances between compressed points

1. PCA is data dependent
2. PCA is deterministic
3. PCA attempts to keep compressed points almost the same as original

PCA vs JL-Transform

1. JL transform is data oblivious
2. JL transform is randomized
3. JL transform preserves pairwise distances between compressed points
4. Coordinates in new space are meaningless
   - New bases are random axes

1. PCA is data dependent
2. PCA is deterministic
3. PCA attempts to keep compressed points almost the same as original
4. Coordinates in new space are meaningful
   - New bases are linear combination of old ones

PCA vs JL-Transform

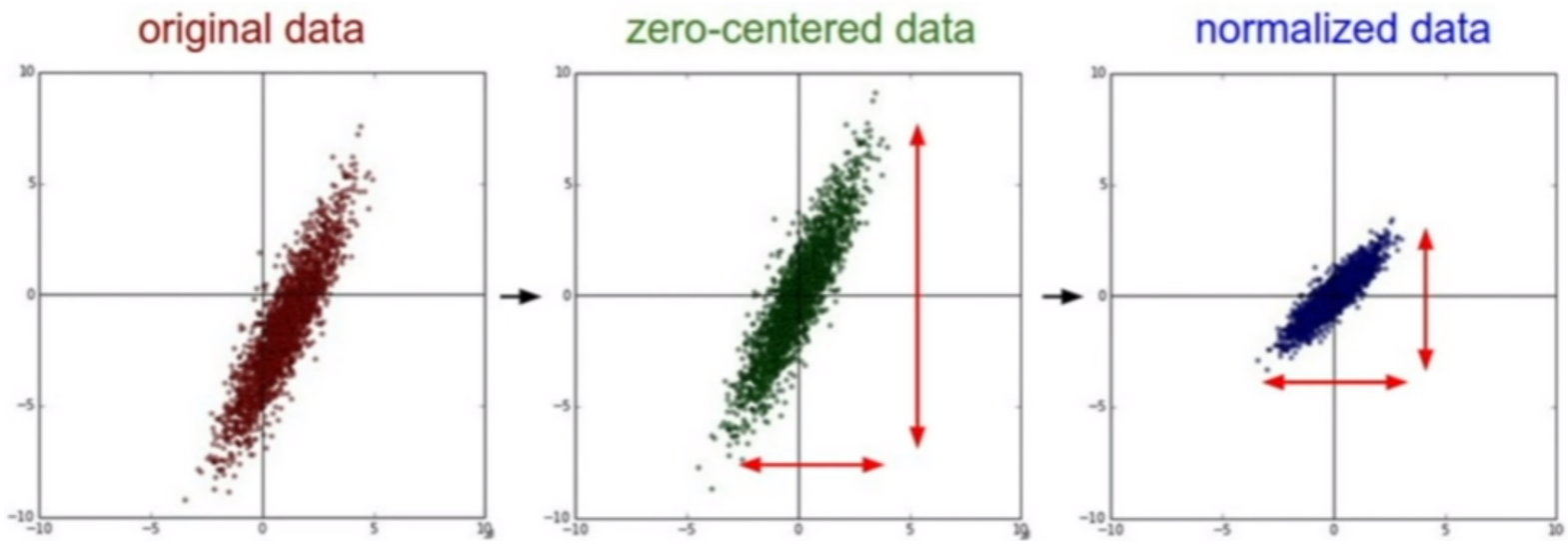
1. JL transform is data oblivious
2. JL transform is randomized
3. JL transform preserves pairwise distances between compressed points
4. Coordinates in new space are meaningless
5. Number of dimensions in new space depends on number of data points
   - $k = \Omega(\log n) \implies (1 \pm \epsilon)$ guarantee

1. PCA is data dependent
2. PCA is deterministic
3. PCA attempts to keep compressed points almost the same as original
4. Coordinates in new space are meaningful
5. Number of dimensions in new space depends on variation in the data
   - Can give very good results even for k 2 or 3

PCA: Preprocessing

Important pre-processing steps for PCA to get that

- All coordinates have same scale and unit (or all be unitless)
- The mean (centroid) of data is $\mathbf{0}$ (all coordinates have mean 0)
- All coordinates have variance 1 ▷ Optional
- Achieved by z-score normalizing each coordinate
- z-score normalization of u_i is done by $u'_{ji} = \dfrac{u_{ji} - \bar{u}_i}{\sigma_i}$, where $\bar{u}_i$ is the mean and σ_i is the std-dev of the variable u_i



Covariance and Correlation

Covariance helps understanding the relationships between variables

For a dataset $X \subset \mathbb{R}^m$, with $|X| = n$ (n points of dimensions m)

- **Covariance** between variables $\mathbf{u}_i$ and $\mathbf{u}_j$ with means $\overline{u_i}$ and $\overline{v_i}$ is

$$COV(\mathbf{u}_i, \mathbf{u}_j) = COV(i,j) = \frac{1}{n}\sum_{\ell=1}^{n}(x_{\ell i} - \overline{u_i})(x_{\ell j} - \overline{u_j})$$

- $COV(\mathbf{u}_i, \mathbf{u}_j) < 0 \implies$ inverse proportionality
- $COV(\mathbf{u}_i, \mathbf{u}_j) > 0 \implies$ direct proportionality
- $COV(\mathbf{u}_i, \mathbf{u}_j) = 0 \implies$ no linear relation

Covariance and Correlation

Correlation

- Covariance value depends on magnitude and scale of the variables
- Correlation quantifies how strongly they are linearly related

$$\rho_{ij} = corr(\mathbf{u}_i, \mathbf{u}_j) = \frac{COV(\mathbf{u}_i, \mathbf{u}_j)}{\sigma_{u_i}.\sigma_{u_j}}$$

- $\rho_{ij} \in [-1, 1]$
- It is not affected by changes in scale of $\mathbf{u}_i$ and $\mathbf{u}_j$
  - $\rho_{ij} = -1 \implies$ perfect negative linear association
  - $\rho_{ij} = 1 \implies$ perfect positive linear association
  - $\rho_{ij} = 0 \implies$ no linear association
- For standardized data (mean 0 and variance 1) $\rho_{ij} = COV(\mathbf{u}_i, \mathbf{u}_j)$

Variance-Covariance Matrix

For a dataset $X \subset \mathbb{R}^m$, with $|X| = n$ (n points of dimensions m)

Variance-Covariance Matrix a $m \times m$ matrix C or Σ, $C_{ij} = COV(\mathbf{u}_i, \mathbf{u}_j)$

$$X$$

m features/dimensions

$$\begin{array}{c} n \text{ points} \\ \end{array} \begin{pmatrix} \mathbf{u_1} & \mathbf{u_2} & \cdots & & \mathbf{u_m} \\ x_{11} & x_{12} & \cdots & & x_{1m} \\ x_{21} & x_{22} & \cdots & & x_{2m} \\ \vdots & \vdots & \cdots & & \vdots \\ \vdots & \vdots & \cdots & & \vdots \\ \vdots & \vdots & \cdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & & x_{nm} \end{pmatrix}$$

$$C \ or \ \Sigma$$

m

$$m \begin{pmatrix} Cov(u_1,u_1) & Cov(u_1,u_2) & \cdots & Cov(u_1,u_m) \\ Cov(u_2,u_1) & Cov(u_2,u_2) & \cdots & Cov(u_2,u_m) \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ Cov(u_m,u_1) & Cov(u_m,u_2) & \cdots & Cov(u_m,u_m) \end{pmatrix}$$

C is a symmetric matrix. For standardized dataset

- Entries on the principal diagonal (the variances) are 1
- Off diagonal entries are the pairwise correlations i.e. $C = X^T X$

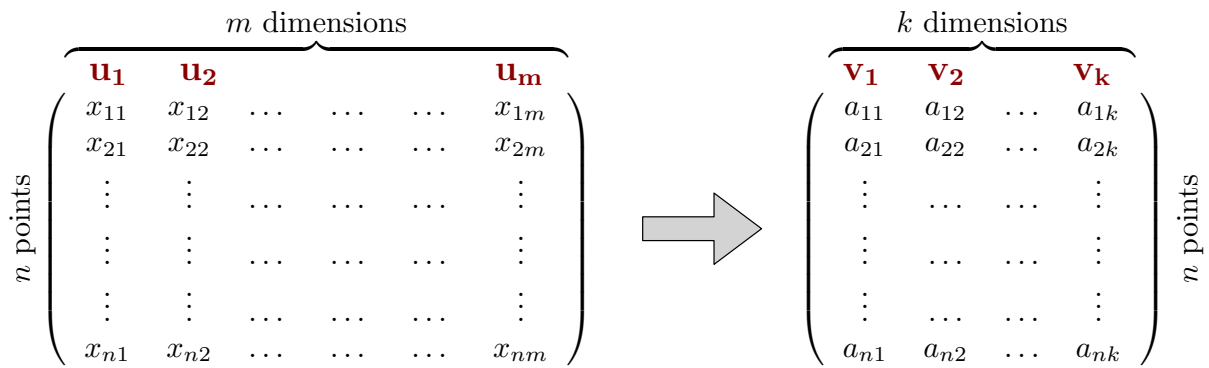
PCA: Setup

Data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$ viewed as rows of a $n \times m$ matrix
Original variables are $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_m}$ (standard bases vectors of $\mathbb{R}^m$)
$\mathbf{x}_i \in X$ is linear combination of $\mathbf{u_1}, \ldots, \mathbf{u_m}$ ($x_{i1}, \ldots, x_{im}$ are coefficients)

$$\mathbf{x}_i = x_{i1}\mathbf{u_1} + x_{i2}\mathbf{u_2} + \ldots + x_{im}\mathbf{u_m}$$

Find a k-d representation $X' = \{\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_n\}$ for X
Project X onto a k-d subspace spanned by bases vectors $\mathbf{v_1}, \ldots, \mathbf{v_k} \subset \mathbb{R}^m$
$\mathbf{x}'_i$ is a linear combination of $\mathbf{v_1}, \ldots, \mathbf{v_k}$

$$\mathbf{x}'_i = a_{i1}\mathbf{v_1} + a_{i2}\mathbf{v_2} + \ldots + a_{ik}\mathbf{v_k}, \qquad a_{ij} = \langle x_i, \mathbf{v_j} \rangle = x_i \cdot \mathbf{v_j}$$



PCA: Setup

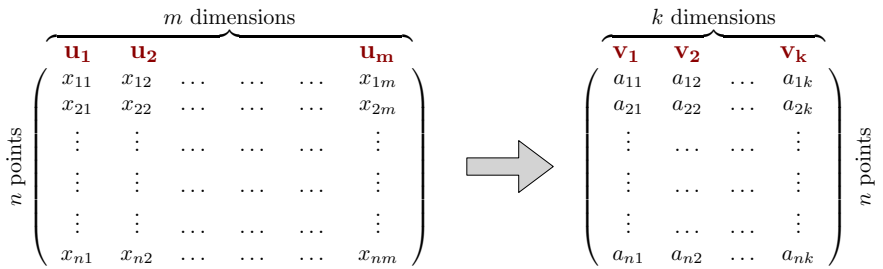
Given X as a $n \times m$ matrix (rows are $\mathbf{x}_i$'s, columns are dimensions)

Original variables are $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_m}$ (standard bases vectors of $\mathbb{R}^m$)

$$\mathbf{x}_i = x_{i1}\mathbf{u_1} + x_{i2}\mathbf{u_2} + \ldots + x_{in}\mathbf{u_n}$$

Project X onto a k-d subspace spanned by bases vectors $\mathbf{v_1}, \ldots, \mathbf{v_k} \subset \mathbb{R}^m$

$$\mathbf{x}'_i = a_{i1}\mathbf{v_1} + a_{i2}\mathbf{v_2} + \ldots + a_{ik}\mathbf{v_k}, \qquad a_{ij} = \langle x_i, \mathbf{v_j} \rangle = x_i \cdot \mathbf{v_j}$$

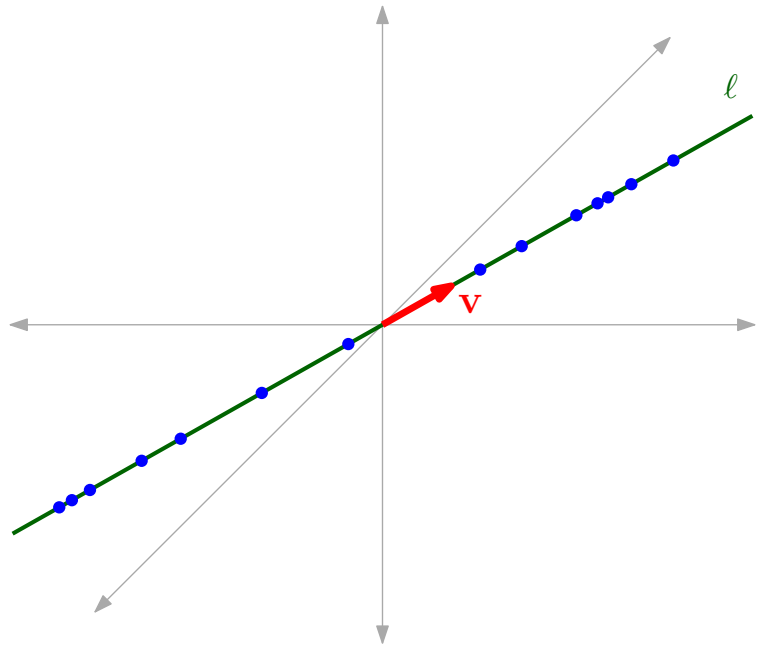


PCA Goals:

- Find the best vectors $\mathbf{v_1}, \ldots, \mathbf{v_k} \subset \mathbb{R}^m$
- Avoid redundancy $\Longrightarrow$ $\mathbf{v_1}, \ldots, \mathbf{v_k}$ are orthonormal
- Need to formulate the objective (goodness measure)

Principal Component Analysis

As a warm-up exercise, suppose the m-d data lies on a line

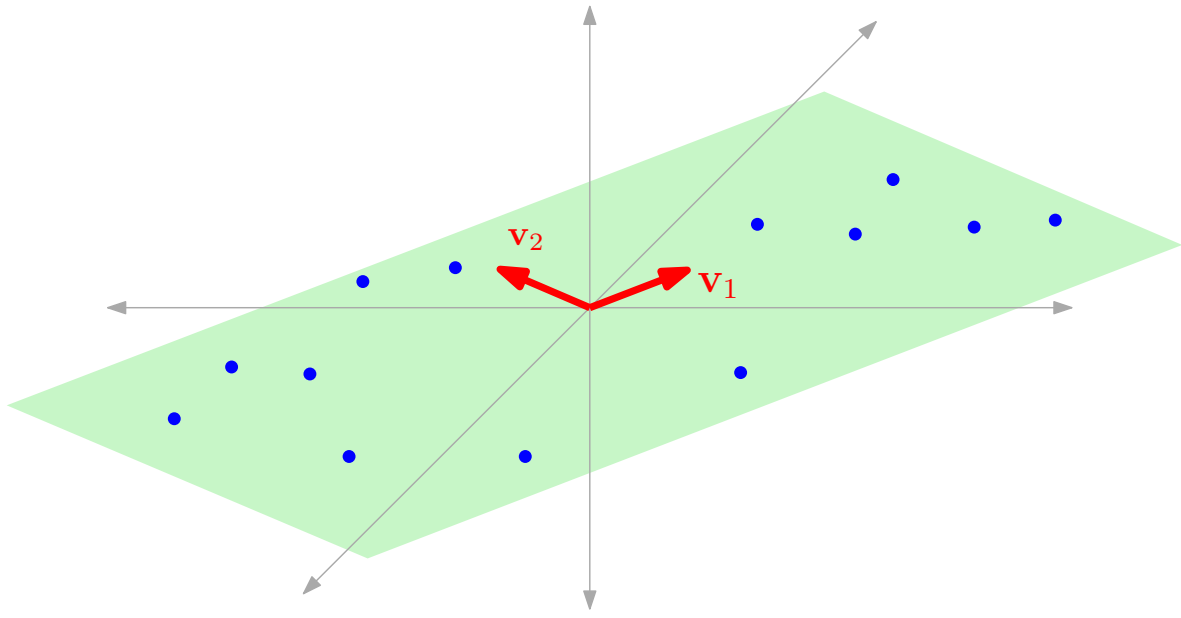


- Let $\mathbf{v}$ be the unit vector in direction of ℓ
- For $\mathbf{x_i} \in X$, let $f(\mathbf{x_i}) := \mathbf{v} \cdot \mathbf{x}$
- In this case, since $\mathbf{v} \cdot \mathbf{x}_i = \mathbf{x}_i$ (as $\mathbf{x}_i$ lies on ℓ), we get

$$\forall i, j \quad \|f(x_i) - f(x_j)\| = \|\mathbf{v} \cdot \mathbf{x}_i - \mathbf{v} \cdot \mathbf{x_j}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$$

Principal Component Analysis

If the m-d data lies on a k-d plane with orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$

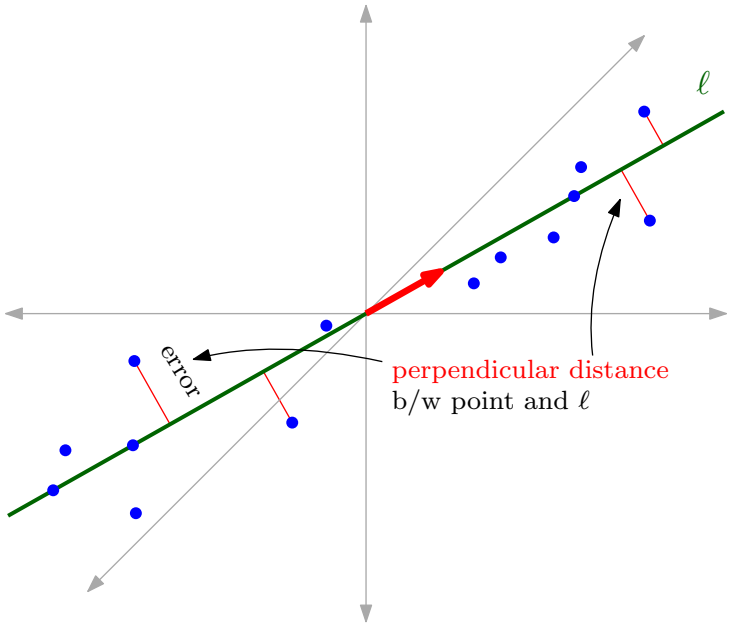


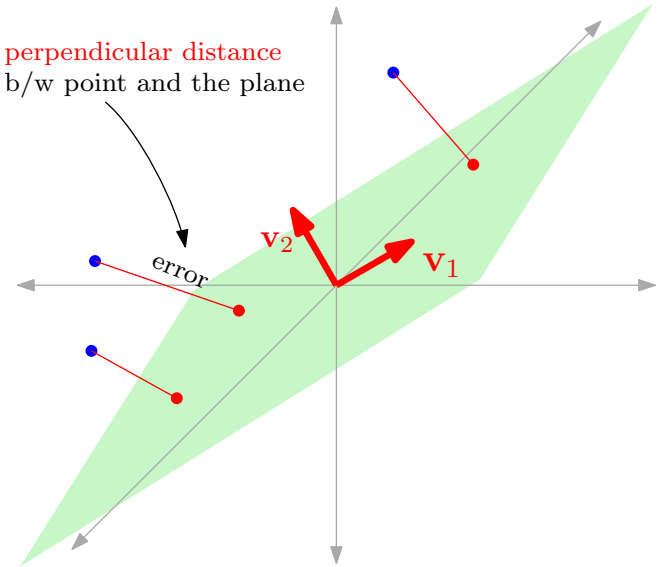
- Let $\mathbf{V}$ be the matrix with $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ as columns
- For $\mathbf{x_i} \in X$, let $f(\mathbf{x_i}) := \mathbf{x}_i^T \mathbf{V}$, we get

$$\forall i, j \quad \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\| = \|\mathbf{x}_i^T \mathbf{V} - \mathbf{x_j}^T \mathbf{V}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$$

In above cases, we get 0 error (no-distortion) dimensionality reduction

Principal Component Analysis





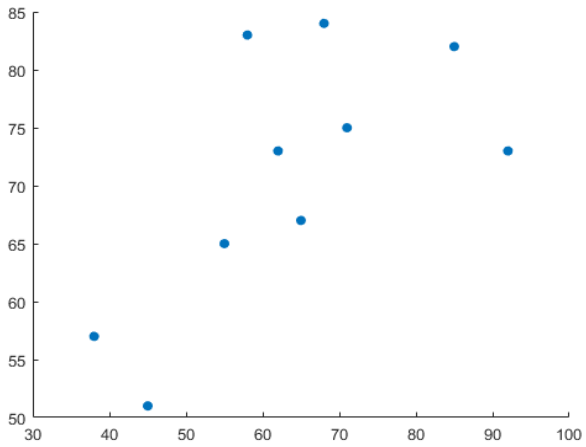
PCA finds the low dimensional space to which the data is close

- Similar to (multiple) linear regression, but
  1. Error there is vertical distance from subspace not perpendicular distance
  2. There no mention of $\mathbf{v}_1, \ldots$ and their orthonormality
- PCA finds orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ spanning the subspace to which perpendicular distances from points in X are minimum

PCA: Example with $k = 1$

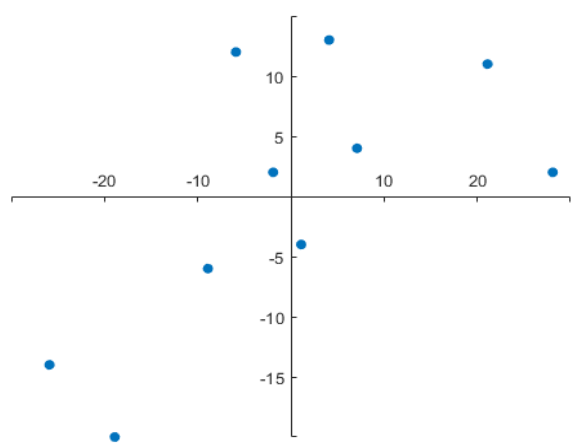
Represent 10 students records in 2 courses by one number

	$\mathbf{c}_1$	$\mathbf{c}_2$
s_1	92	73
s_2	85	82
s_3	71	75
s_4	68	84
s_5	65	67
s_6	62	73
s_7	58	83
s_8	45	51
s_9	38	57
s_{10}	55	65

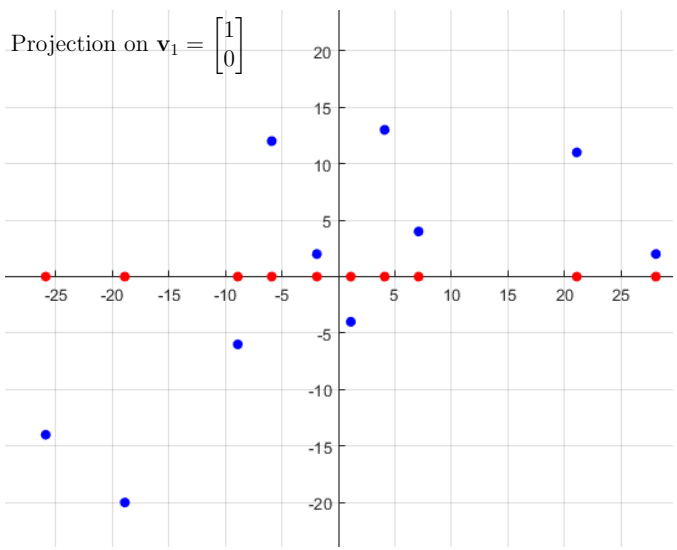


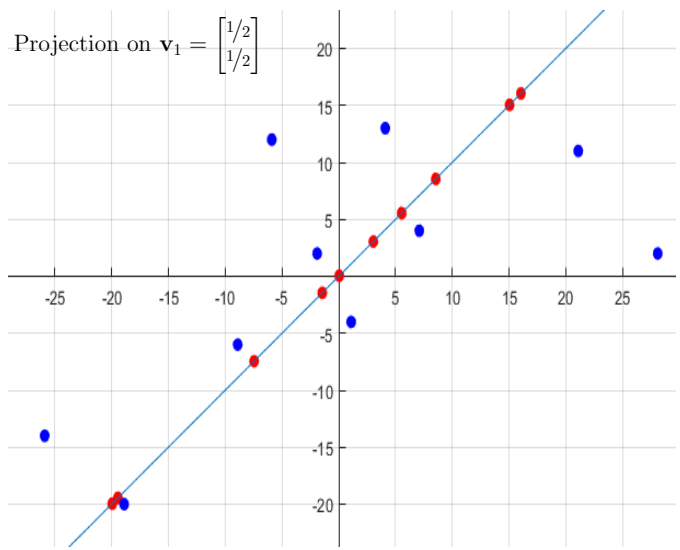
Raw Data

	$\mathbf{c}_1$	$\mathbf{c}_2$
s_1	28.1	2
s_2	21.1	11
s_3	7.1	4
s_4	4.1	13
s_5	1.1	-4
s_6	-1.9	2
s_7	-5.9	12
s_8	-18.9	-20
s_9	-25.9	-14
s_{10}	-8.9	-6



Zero-Centered Data

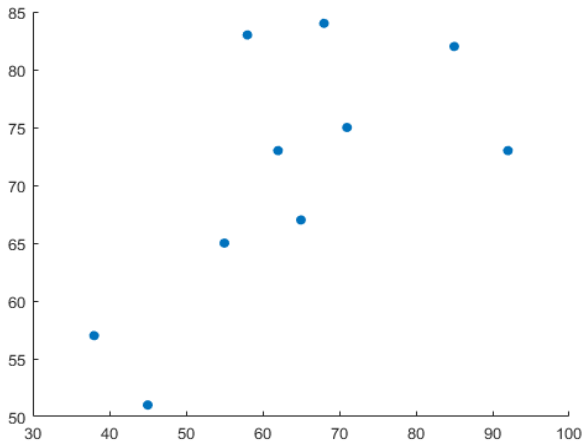




PCA: Example with $k = 1$

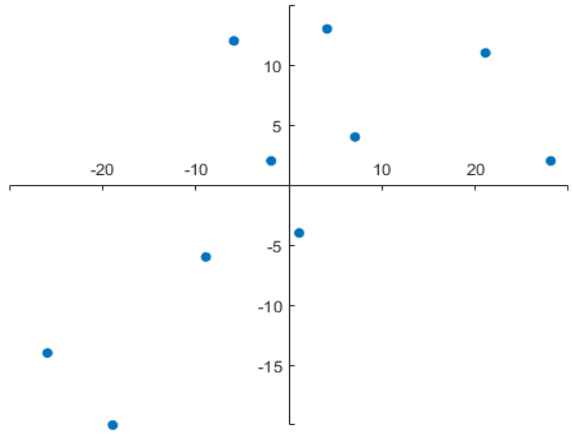
Represent 10 students records in 2 courses by one number

	$\mathbf{c}_1$	$\mathbf{c}_2$
s_1	92	73
s_2	85	82
s_3	71	75
s_4	68	84
s_5	65	67
s_6	62	73
s_7	58	83
s_8	45	51
s_9	38	57
s_{10}	55	65

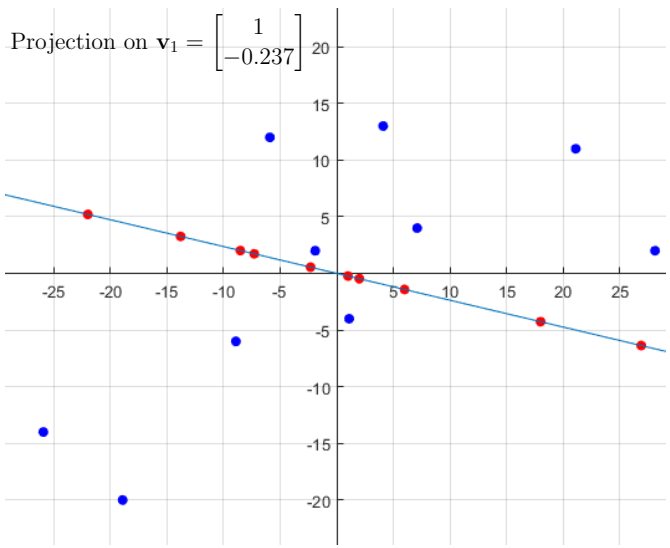


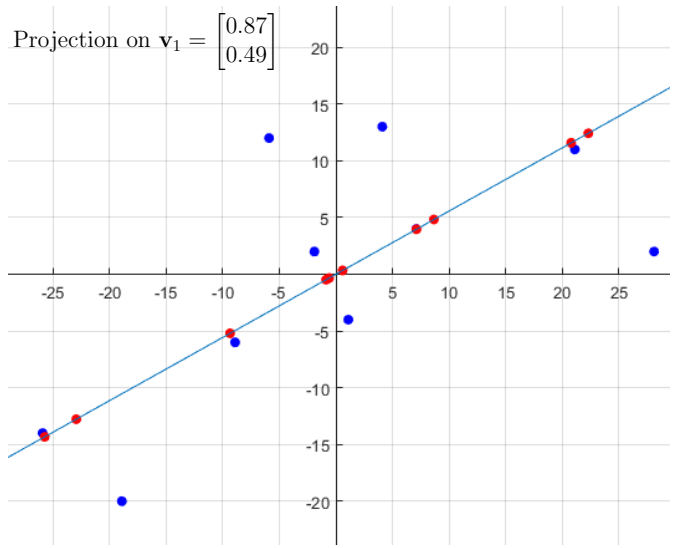
Raw Data

	$\mathbf{c}_1$	$\mathbf{c}_2$
s_1	28.1	2
s_2	21.1	11
s_3	7.1	4
s_4	4.1	13
s_5	1.1	-4
s_6	-1.9	2
s_7	-5.9	12
s_8	-18.9	-20
s_9	-25.9	-14
s_{10}	-8.9	-6



Zero-Centered Data





PCA: A Bigger Example

Record of 16 students in 4 courses (original and zero-centered)

	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$
s_1	95	89	70	64
s_2	91	91	71	70
s_3	79	77	65	58
s_4	76	74	68	69
s_5	76	69	65	64
s_6	78	68	65	64
s_7	79	70	47	42
s_8	62	61	47	46
s_9	68	63	88	88
s_{10}	68	67	90	89
s_{11}	66	63	82	75
s_{12}	66	67	78	70
s_{13}	68	63	75	72
s_{14}	64	63	76	70
s_{15}	53	46	79	72
s_{16}	43	42	61	60

	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$
s_1	24.3	21.9	−0.4	−3.1
s_2	20.3	23.9	0.6	2.9
s_3	8.3	9.9	−5.4	−9.1
s_4	5.3	6.9	−2.4	1.9
s_5	5.3	1.9	−5.4	−3.1
s_6	7.3	0.9	−5.4	−3.1
s_7	8.3	2.9	−23.4	−25.1
s_8	−8.8	−6.1	−23.4	−21.1
s_9	−2.8	−4.1	17.6	20.9
s_{10}	−2.8	−0.1	19.6	21.9
s_{11}	−4.8	−4.1	11.6	7.9
s_{12}	−4.8	−0.1	7.6	2.9
s_{13}	−2.8	−4.1	4.6	4.9
s_{14}	−6.8	−4.1	5.6	2.9
s_{15}	−17.8	−21.1	8.6	4.9
s_{16}	−27.8	−25.1	−9.4	−7.1

PCA: A Bigger Example

Data porjected on $\mathbf{v_1} = \begin{bmatrix} 0.6 & 0.6 & -0.4 & -0.4 \end{bmatrix}^T$

	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$
s_1	24.3	21.9	−0.4	−3.1
s_2	20.3	23.9	0.6	2.9
s_3	8.3	9.9	−5.4	−9.1
s_4	5.3	6.9	−2.4	1.9
s_5	5.3	1.9	−5.4	−3.1
s_6	7.3	0.9	−5.4	−3.1
s_7	8.3	2.9	−23.4	−25.1
s_8	−8.8	−6.1	−23.4	−21.1
s_9	−2.8	−4.1	17.6	20.9
s_{10}	−2.8	−0.1	19.6	21.9
s_{11}	−4.8	−4.1	11.6	7.9
s_{12}	−4.8	−0.1	7.6	2.9
s_{13}	−2.8	−4.1	4.6	4.9
s_{14}	−6.8	−4.1	5.6	2.9
s_{15}	−17.8	−21.1	8.6	4.9
s_{16}	−27.8	−25.1	−9.4	−7.1

$a_{i1} = \langle s_i, \mathbf{v_1} \rangle$
28.7
24.7
16.3
7.4
7.6
8.2
25.5
8.4
−18.9
−17.8
−12.8
−7
−7.7
−9.7
−28.1
−24.9

$s_i' = \langle s_i, \mathbf{v_1} \rangle \mathbf{v_1}$			
17.5	16.5	−11.2	−11.1
15	14.2	−9.6	−9.5
9.9	9.4	−6.3	−6.3
4.5	4.2	−2.9	−2.8
4.6	4.4	−3	−2.9
5	4.7	−3.2	−3.2
15.5	14.6	−9.9	−9.9
5.1	4.8	−3.3	−3.3
−11.5	−10.8	7.4	7.3
−10.8	−10.2	6.9	6.9
−7.8	−7.3	5	4.9
−4.3	−4	2.7	2.7
−4.7	−4.4	3	3
−5.9	−5.6	3.8	3.8
−17.1	−16.1	10.9	10.9
−15.1	−14.3	9.7	9.6

- Compare s_i' with s_i
- Coordinates with big values are still bigger
- Coordinates with smaller values are somewhat smaller
- We saved 75% of storage. For s_i only need to save a_{i1}

PCA: A Bigger Example

Projection on plane spanned by $\mathbf{v_1} = \begin{bmatrix} 0.6 & 0.6 & -0.4 & -0.4 \end{bmatrix}^T$ and $\mathbf{v_2} = \begin{bmatrix} 0.4 & 0.4 & 0.6 & 0.6 \end{bmatrix}^T$

	c_1	c_2	c_3	c_4
s_1	24.3	21.9	−0.4	−3.1
s_2	20.3	23.9	0.6	2.9
s_3	8.3	9.9	−5.4	−9.1
s_4	5.3	6.9	−2.4	1.9
s_5	5.3	1.9	−5.4	−3.1
s_6	7.3	0.9	−5.4	−3.1
s_7	8.3	2.9	−23.4	−25.1
s_8	−8.8	−6.1	−23.4	−21.1
s_9	−2.8	−4.1	17.6	20.9
s_{10}	−2.8	−0.1	19.6	21.9
s_{11}	−4.8	−4.1	11.6	7.9
s_{12}	−4.8	−0.1	7.6	2.9
s_{13}	−2.8	−4.1	4.6	4.9
s_{14}	−6.8	−4.1	5.6	2.9
s_{15}	−17.8	−21.1	8.6	4.9
s_{16}	−27.8	−25.1	−9.4	−7.1

a_{i1}	a_{i2}
28.7	15.8
24.7	19.3
16.3	−1.5
7.4	4.5
7.6	−2.3
8.2	−1.9
25.5	−24.4
8.4	−32
−18.9	20.1
−17.8	23.5
−12.8	8.1
−7	4.4
−7.7	3
−9.7	0.9
−28.1	−7.1
−24.9	−30.2

$s'_i = a_{i1}\mathbf{v_1} + a_{i2}\mathbf{v_2}$			
23.4	22.8	−1.8	−1.7
22.3	21.9	1.8	1.9
9.4	8.8	−7.2	−7.2
6.2	6	−0.2	−0.2
3.8	3.4	−4.3	−4.3
4.3	4	−4.3	−4.3
6.4	4.9	−24.3	−24.3
−6.9	−8	−22.2	−22.2
−4	−2.8	19.2	19.2
−2	−0.8	20.8	20.8
−4.7	−4.1	9.8	9.7
−2.6	−2.3	5.3	5.3
−3.6	−3.2	4.7	4.7
−5.6	−5.2	4.3	4.3
−19.8	−19	6.7	6.7
−26.5	−26.4	−8.2	−8.3

- Compare s'_i with s_i
- In general s'_i are closer to s_i
- We saved 50% of storage. For s_i only need to save a_{i1} and a_{i2}

PCA A Bigger Example

Projection on hyperplane spanned by $\mathbf{v_1} = \begin{bmatrix} 0.6 & 0.6 & -0.4 & -0.4 \end{bmatrix}^T$, $\mathbf{v_2} = \begin{bmatrix} 0.4 & 0.4 & 0.6 & 0.6 \end{bmatrix}^T$, and $\mathbf{v_3} = \begin{bmatrix} -0.7 & 0.7 & 0.1 & -0.1 \end{bmatrix}^T$

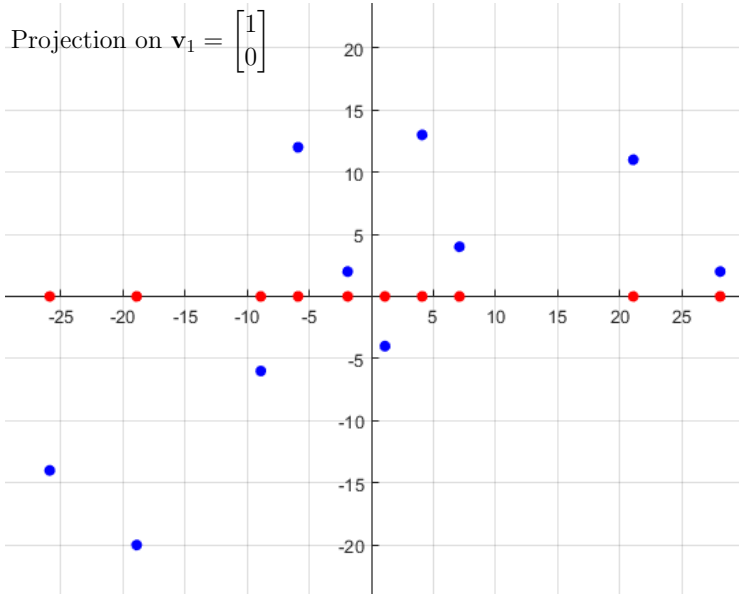
	c_1	c_2	c_3	c_4
s_1	24.3	21.9	−0.4	−3.1
s_2	20.3	23.9	0.6	2.9
s_3	8.3	9.9	−5.4	−9.1
s_4	5.3	6.9	−2.4	1.9
s_5	5.3	1.9	−5.4	−3.1
s_6	7.3	0.9	−5.4	−3.1
s_7	8.3	2.9	−23.4	−25.1
s_8	−8.8	−6.1	−23.4	−21.1
s_9	−2.8	−4.1	17.6	20.9
s_{10}	−2.8	−0.1	19.6	21.9
s_{11}	−4.8	−4.1	11.6	7.9
s_{12}	−4.8	−0.1	7.6	2.9
s_{13}	−2.8	−4.1	4.6	4.9
s_{14}	−6.8	−4.1	5.6	2.9
s_{15}	−17.8	−21.1	8.6	4.9
s_{16}	−27.8	−25.1	−9.4	−7.1

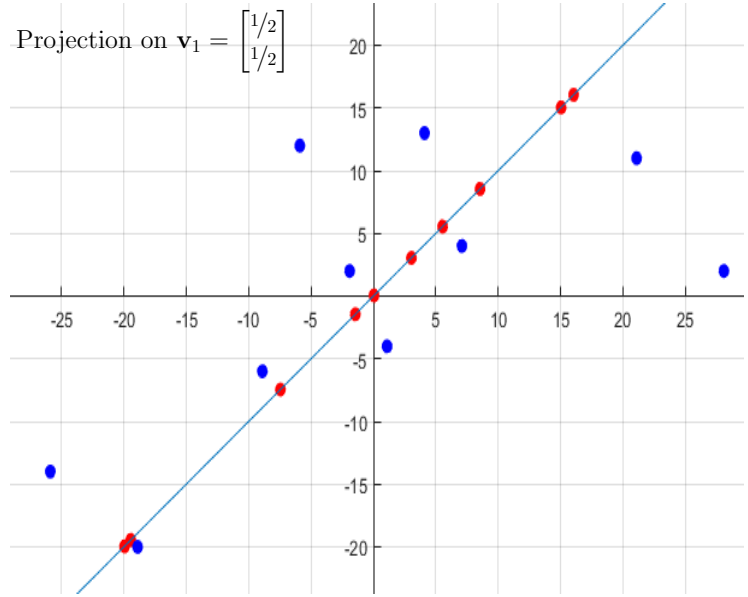
	$a_{i1}\mathbf{v_1} + a_{i2}\mathbf{v_2} + a_{i3}\mathbf{v_3}$			
s_1'	24	22.2	−1.9	−1.6
s_2'	20.5	23.7	2	1.5
s_3'	8	10.2	−7.1	−7.5
s_4'	5.6	6.6	−0.2	−0.3
s_5'	5.4	1.8	−4.5	−4
s_6'	7.4	0.8	−4.7	−3.8
s_7'	8.1	3.1	−24.5	−24
s_8'	−8.5	−6.3	−22	−22.5
s_9'	−2.5	−4.3	19	19.5
s_{10}'	−2.6	−0.3	20.8	20.7
s_{11}'	−5	−3.8	9.8	9.7
s_{12}'	−5	0.2	5.6	4.9
s_{13}'	−2.7	−4.1	4.6	4.9
s_{14}'	−6.9	−3.9	4.5	4
s_{15}'	−18.1	−20.7	6.5	7
s_{16}'	−27.5	−25.3	−8	−8.5

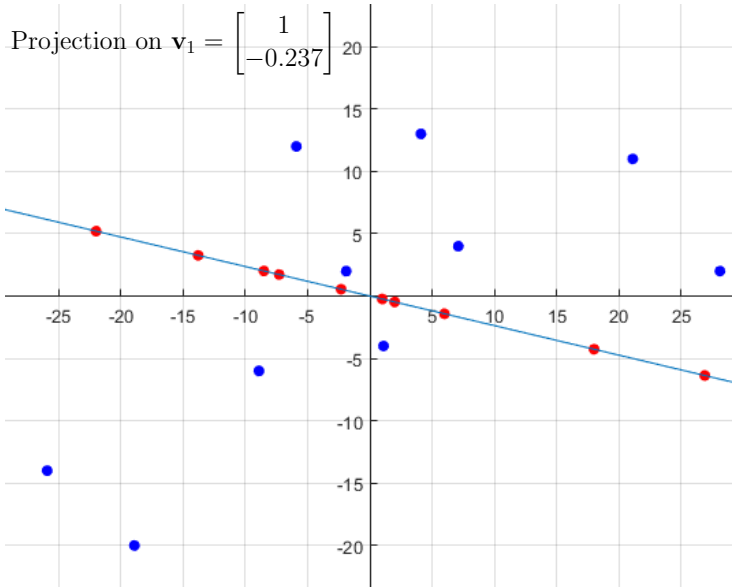
- Compare s_i' with s_i
- Each s_i' is almost the same as s_i
- We saved 25% of storage. For s_i only need to save a_{i1}, a_{i2} and a_{i3}

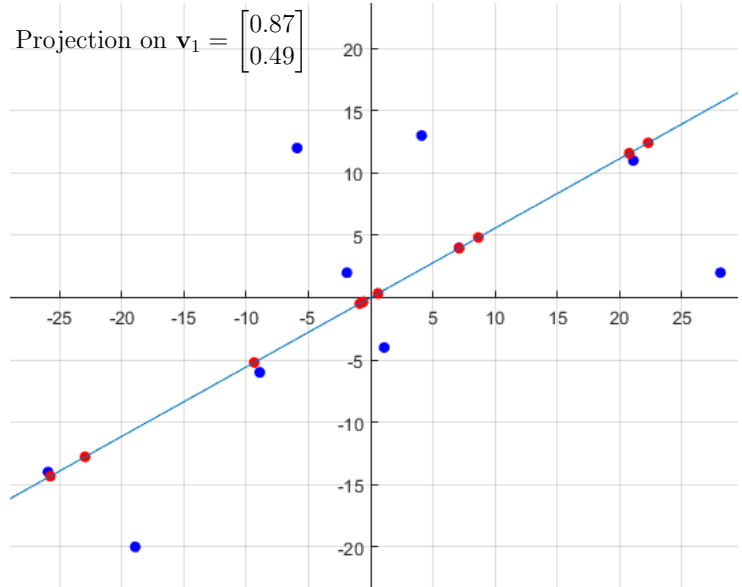
PCA Objective

Which vectors to project on ?







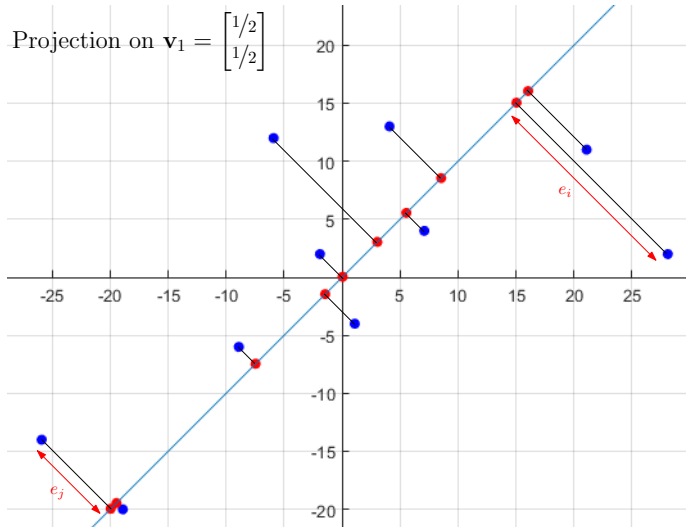


PCA Objective

	$\mathbf{c}_1$	$\mathbf{c}_2$
s_1	28.1	2
s_2	21.1	11
s_3	7.1	4
s_4	4.1	13
s_5	1.1	−4
s_6	−1.9	2
s_7	−5.9	12
s_8	−18.9	−20
s_9	−25.9	−14
s_{10}	−8.9	−6

	$a_{i1}\mathbf{v_1}$	
s_1'	15	15
s_2'	16	16
s_3'	5.5	5.5
s_4'	8.5	8.5
s_5'	−1.4	−1.4
s_6'	0	0
s_7'	3	3
s_8'	−19.4	−19.4
s_9'	−19.9	−19.9
s_{10}'	−7.4	−7.4

	$\|\|s_i - s_i'\|\|$
e_1	18.5
e_2	7.1
e_3	2.2
e_4	6.3
e_5	3.6
e_6	2.8
e_7	12.7
e_8	0.8
e_9	8.4
e_{10}	2.1



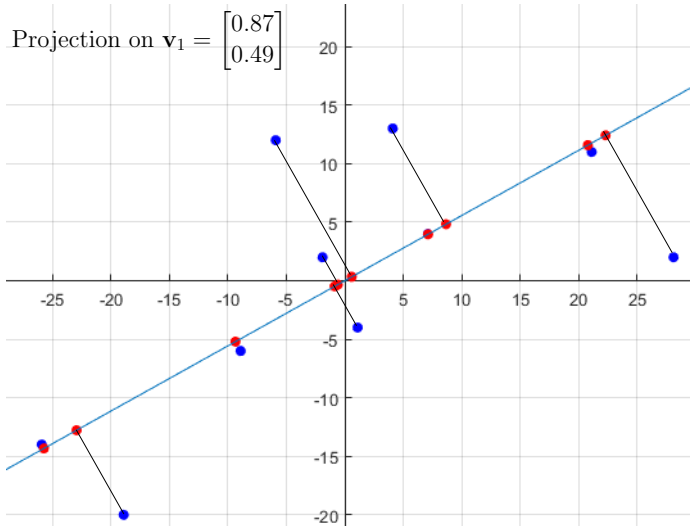
Total Error (sum)
= 64.3467

PCA Objective

	$\mathbf{c}_1$	$\mathbf{c}_2$
s_1	28.1	2
s_2	21.1	11
s_3	7.1	4
s_4	4.1	13
s_5	1.1	−4
s_6	−1.9	2
s_7	−5.9	12
s_8	−18.9	−20
s_9	−25.9	−14
s_{10}	−8.9	−6

	$a_{i1}\mathbf{v_1}$	
s_1'	22.3	12.4
s_2'	20.8	11.6
s_3'	7.1	4
s_4'	8.7	4.8
s_5'	−0.9	−0.5
s_6'	−0.6	−0.3
s_7'	0.6	0.3
s_8'	−22.9	−12.8
s_9'	−25.7	−14.3
s_{10}'	−9.3	−5.2

	$\|\|s_i - s_i'\|\|$
e_1	11.9
e_2	0.7
e_3	0
e_4	9.4
e_5	4
e_6	2.7
e_7	13.4
e_8	8.3
e_9	0.4
e_{10}	0.9



Total Error (sum)
= 51.6030

PCA Objective ($k = 1$)

Find a vector to project the data onto that results in minimum reconstruction error or minimum information loss

Given $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$,

Find a unit vector $\mathbf{v_1} \in \mathbb{R}^m$ and get $X' = \{\mathbf{x}'_i = \langle \mathbf{x}_i, \mathbf{v}_1 \rangle \mathbf{v}_1\}$ that minimizes the total reconstruction error

$$\sum_{\mathbf{x}_i \in X} e_i := \sum_{\mathbf{x}_i \in X} \|\mathbf{x}_i - \mathbf{x}'_i\| = \sum_{\mathbf{x}_i \in X} (\text{distance b/w } \mathbf{x}_i \text{ and line spanned by } \mathbf{v}_1)$$

$\|\mathbf{x}_i - \mathbf{x}'_i\|$ is the perpendicular distance between $\mathbf{x}_i$ and the line spanned by $\mathbf{v}_1$ or between $\mathbf{x}_i$ and its projection on $\mathbf{v}_1$

Squared reconstruction error $\because$

- The algebra is easier and has a
- nice connection with variance in data

$$\underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\min} \sum_{\mathbf{x}_i \in X} \|\mathbf{x}_i - \mathbf{x}'_i\|^2 = \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\min} \sum_{\mathbf{x}_i \in X} (\text{dist b/w } \mathbf{x}_i \text{ and line spanned by } \mathbf{v}_1)^2$$

PCA Objective ($k = 1$)

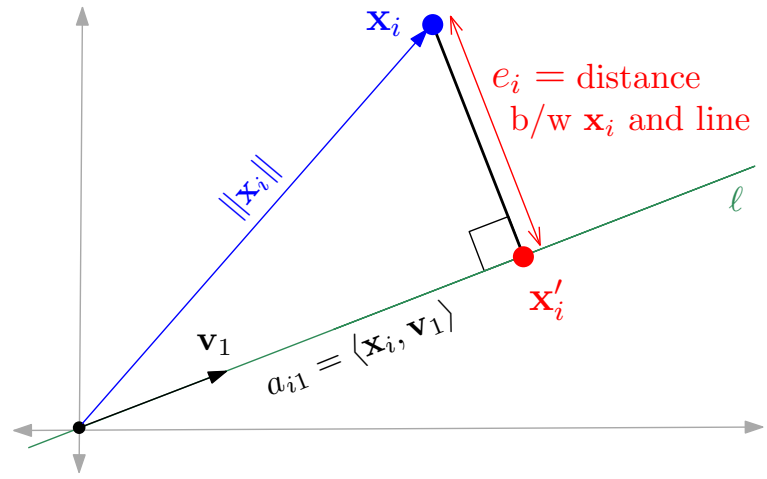
$$\arg\min_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} \|\mathbf{x}_i - \mathbf{x}_i'\|^2 = \arg\min_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} (\text{dist b/w } \mathbf{x}_i \text{ and line spanned by } \mathbf{v}_1)^2$$

For $\mathbf{x}_i$ and it's projection on $\mathbf{v}_1$

$$(\text{dist b/w } \mathbf{x}_i \text{ and span of } \mathbf{v}_1)^2$$

$$=$$

$$\|\mathbf{x}_i\|^2 - \langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2$$



Phythogorus theorem

$\|\mathbf{x}_i\|^2$ is constant $\implies$ minimizing LHS is maximizing $\langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2$

Thus the objective function of PCA (with $k = 1$) is

$$\arg\max_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} \langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2 = \arg\max_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} a_{i1}^2$$

PCA Objective ($k = 1$)

The objective function of PCA (with $k = 1$) is

$$\underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\min} \sum_{\mathbf{x}_i \in X} \|\mathbf{x}_i - \mathbf{x}_i'\|^2 = \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\min} \sum_{\mathbf{x}_i \in X} (\text{dist b/w } \mathbf{x}_i \text{ and line spanned by } \mathbf{v}_1)^2$$

$$\text{Equivalently,} \quad \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max} \sum_{\mathbf{x}_i \in X} \langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2 = \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max} \sum_{\mathbf{x}_i \in X} a_{i1}^2$$

X is zero-centered $\implies$ $\sum_{i=1}^{n} \mathbf{x}_{ij}^2 = \sigma_j^2$, variance in jth coordinate of X

By linearity of dot-product ($\mathbf{p} \cdot (\mathbf{q} + \mathbf{r}) = \mathbf{p} \cdot \mathbf{q} + \mathbf{p} \cdot \mathbf{r}$), we get

$$\sum_{i=1}^{n} a_{i1} = \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{v}_1 \rangle = \langle \sum_{i=1}^{n} \mathbf{x}_i, \mathbf{v}_1 \rangle = 0$$

$\because$ first term in dot product is 0

The projections a_{i1}'s are zero-centered and $\sum_{\mathbf{x}_i \in X} a_{i1}^2$ is variance in a_{i1}'s

PCA Objective ($k = 1$)

The objective function of PCA (with $k = 1$) is

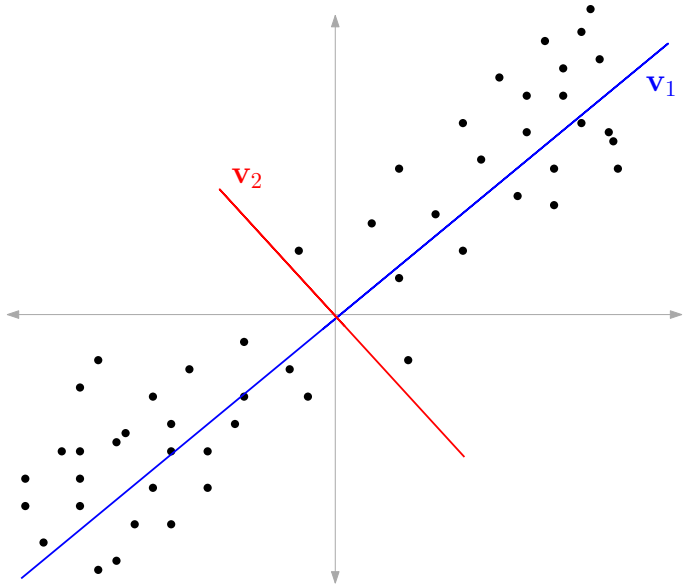
$$\arg\min_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} \|\mathbf{x}_i - \mathbf{x}_i'\|^2 = \arg\min_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} (\text{dist b/w } \mathbf{x}_i \text{ and line spanned by } \mathbf{v}_1)^2$$

$$\text{Equivalently,} \quad \arg\max_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} \langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2 = \arg\max_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} a_{i1}^2$$

Objective of PCA ($k = 1$): Find direction of the most variance of X

This data has much higher variance in the direction of $\mathbf{v_1}$ than in the direction of $\mathbf{v_2}$

Thus PCA seeks $\mathbf{v_1}$



PCA Objective ($k > 1$)

Project the dataset onto a k-dimensional hyperplane S so the sum of squared distances of points to S is minimum

$$\underset{k\text{-d hyperplane } S}{\arg\min} \sum_{\mathbf{x}_i \in X} \|\mathbf{x}_i - \mathbf{x}_i'\|^2 := \underset{k\text{-d hyperplane } S}{\arg\min} \sum_{\mathbf{x}_i \in X} (\text{distance b/w } \mathbf{x}_i \text{ and } S)^2$$

$\mathbf{x}_i'$ is the projection of $\mathbf{x}_i \in X$ on S

$$\text{Equivalently, } \underset{k\text{-d hyperplane } S}{\arg\max} \sum_{\mathbf{x}_i \in X} (\text{length of projection of } \mathbf{x}_i \text{ on } S)^2$$

Represent S by orthonormal bases $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$
Keeps algebra simple, length of projections of x is easy to compute

$$\left(\text{length of projection of } x_i \text{ on } S = \mathit{span}(\mathbf{v_1}, \ldots, \mathbf{v_k})\right)^2 = \sum_{j=1}^{k} \langle x_i, \mathbf{v_j} \rangle^2$$

PCA Objective ($k > 1$)

$$\underset{\substack{\mathbf{v}_1,\ldots,\mathbf{v}_k \\ \|\mathbf{v}_p\|=1 \\ \mathbf{v}_p \perp \mathbf{v}_q}}{\arg\max} \sum_{i=1}^{n} \underbrace{\sum_{j=1}^{k} \langle \mathbf{x}_i, \mathbf{v}_j \rangle^2}_{\text{squared projection length on } span(\mathbf{v}_1,\ldots,\mathbf{v}_k)}$$

The vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ maximizing this objective are called the *top k principal components of X*

Problem: Principal Component Analysis

Given $X \subset \mathbb{R}^m$, $|X| = n$ and an integer $k \geq 1$, find vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ to maximize the above objective and project X onto $\mathbf{v}_1, \ldots, \mathbf{v}_k$.

PCA Linear Algebraic Formulation ($k = 1$)

The objective function of PCA (with $k = 1$) is

$$\underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\min} \sum_{\mathbf{x}_i \in X} \|\mathbf{x}_i - \mathbf{x}_i'\|^2 = \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\min} \sum_{\mathbf{x}_i \in X} (\text{dist. b/w } \mathbf{x}_i \text{ and line spanned by } \mathbf{v}_1)^2$$

$$\text{Equivalently,} \quad \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max} \sum_{\mathbf{x}_i \in X} \langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2 = \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max} \sum_{\mathbf{x}_i \in X} a_{i1}^2$$

Objective of PCA ($k = 1$): Find direction of the most variance of X

The projection of X on the unit vector $\mathbf{v}_1$ is $X\mathbf{v}_1 = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{v}_1 \rangle \\ \langle \mathbf{x}_2, \mathbf{v}_1 \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{v}_1 \rangle \end{bmatrix}$

We want to maximize the sum of squares of this column vector

$$\sum_{\mathbf{x}_i \in X} \langle \mathbf{x}_i, \mathbf{v_1} \rangle^2 = (X\mathbf{v}_1)^T X\mathbf{v}_1 = \mathbf{v}_1^T X^T X \mathbf{v}_1$$

PCA Linear Algebraic Formulation ($k = 1$)

PCA Objective ($k = 1$) $\triangleright$ $C = X^T X$ covariance matrix

$$\arg\max_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \sum_{\mathbf{x}_i \in X} \langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2 := \arg\max_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \mathbf{v}_1^T X^T X \mathbf{v}_1 := \arg\max_{\mathbf{v}_1, \|\mathbf{v}_1\|=1} \mathbf{v}_1^T C \mathbf{v}_1$$

How to find the vector $\mathbf{v}_1$?

We first discuss the special case when covariance matrix is diagonal to get an understanding

PCA ($k = 1$): Diagonal Covariance Matrix

PCA ($k = 1$) Find the vector $\mathbf{v}_1$

$$\underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max} \sum_{\mathbf{x}_i \in X} \langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2 := \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max}\ \mathbf{v}_1^T X^T X \mathbf{v}_1 := \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max}\ \mathbf{v}_1^T C \mathbf{v}_1$$

Special case: C is diagonal ▷ All correlations $= 0$ (Not realistic)

$$C = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_m \end{bmatrix}$$

▷ C is just a scaling linear transform

Assume $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$ ▷ $C_{ij} \geq 0$

Optimal solution $\mathbf{v}_1$ is standard basis vector of $\mathbb{R}^m$ $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T$

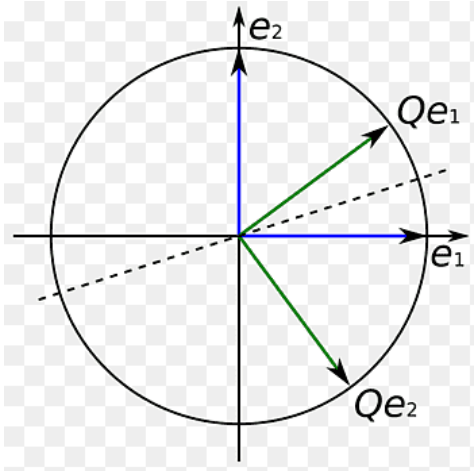
For any vector $\mathbf{u}\mathbb{R}^m$, $\|u\| = 1$ $\sum^m \lambda_i u_i^2 \leq \lambda_1 = \sum^m \lambda_i e_{1i}^2$

Orthogonal Matrices

A real square matrix whose columns and rows are orthonormal vectors
length of columns $= 1$ and columns pairs are orthogonal (dot-product $= 0$)

Properties of an orthogonal matrix Q

- $Q^{\mathrm{T}}Q = QQ^{\mathrm{T}} = I$
- $Q^{-1} = Q^{T}$ ▷ Q is necessarily invertible
- Q preserves vector length i.e. $\|Q\mathbf{v}\| = \|\mathbf{v}\|$ (unitary transformation)
- Q only achieves a rotation, reflection or permutation of coordinates



Eigen Decomposition of C

For every real symmetric matrix the eigenvalues are real and the eigenvectors can be chosen real and orthonormal

- Let $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ be the matrix with columns eigenvectors of C
- For $1 \leq i \leq m$, $C\,\mathbf{q}_i = \lambda_i\,\mathbf{q}_i$
- Q is an orhtogonal matrix

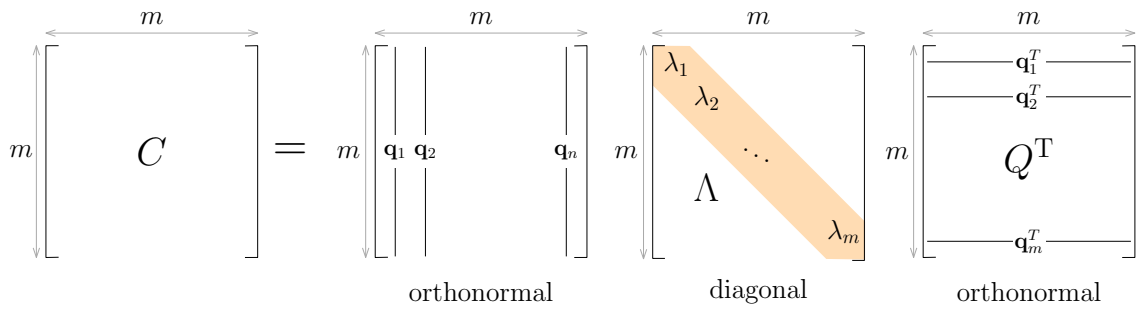
$$Q = \begin{bmatrix} | & | & & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \ldots & \mathbf{q}_m \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{bmatrix}$$

$$C = Q\Lambda Q^{-1}$$

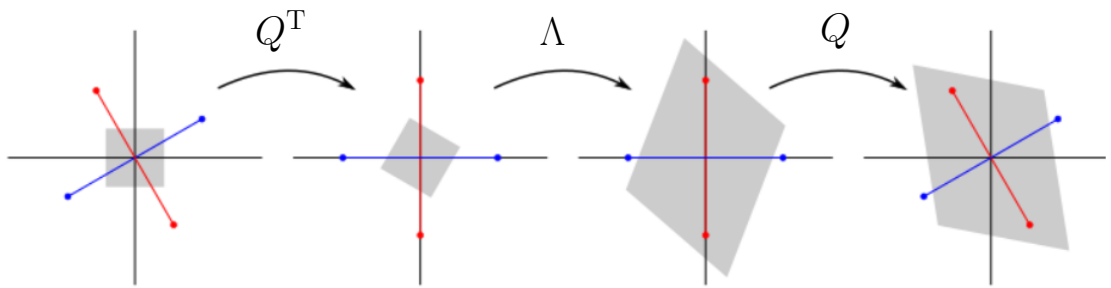
Eigen Decomposition of C

- Let $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ be the matrix with columns eigenvectors of C
- For $1 \leq i \leq m$, $C\,\mathbf{q}_i = \lambda_i\,\mathbf{q}_i$
- Q is an orhtogonal matrix

$$C = Q\Lambda Q^{-1}$$



So square symmetric matrices are rotation, scaling and rotation



Eigen Decomposition of C

PCA ($k = 1$): Find the vector $\mathbf{v}_1$

$$\underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max} \sum_{\mathbf{x}_i \in X} \langle \mathbf{x}_i, \mathbf{v}_1 \rangle^2 := \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max}\ \mathbf{v}_1^T X^T X \mathbf{v}_1 := \underset{\mathbf{v}_1, \|\mathbf{v}_1\|=1}{\arg\max}\ \mathbf{v}_1^T C \mathbf{v}_1$$

$$C = Q \Lambda Q^{-1}$$

$\mathbf{e}_1$ is the direction of maximum stretch under Λ
The direction $\mathbf{v}_1$ of max stretch under $C = Q\Lambda Q^T$ is such that $Q^T \mathbf{v}_1 = \mathbf{e}_1$
$\mathbf{v}_1$ get stretched the most under ΛQ^T and Q does not stretch or shrink it

$$Q^T \mathbf{v}_1 = \mathbf{e}_1 \implies \mathbf{v}_1 = (Q^T)^{-1} \mathbf{e}_1 = Q \mathbf{e}_1$$

$\mathbf{v}_1 = Q\mathbf{e}_1$ is the first column of Q or the leading eigenvector of C
To get the top k principal components we use the first k leading eigenvectors of C. This is very easy to see, again first in the case when C is a diagonal matrix.

Eigen Decomposition of C

PCA ($k > 1$): Find the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$

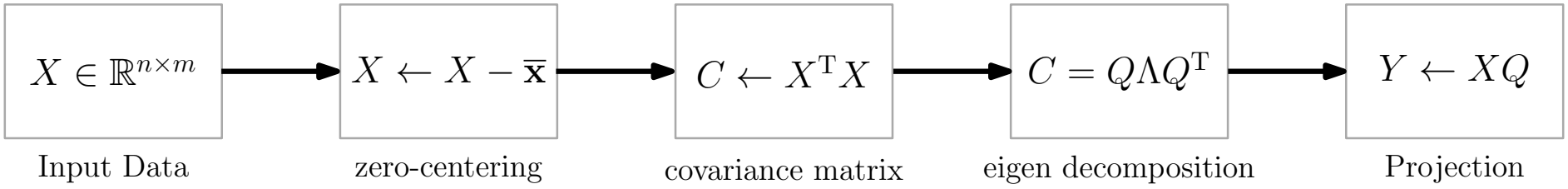
$$\underset{\substack{\mathbf{v}_1,\ldots,\mathbf{v}_k \\ \|\mathbf{v}_p\| = 1 \\ \mathbf{v}_p \perp \mathbf{v}_q}}{\arg\max} \sum_{i=1}^{n} \underbrace{\sum_{j=1}^{k} \langle \mathbf{x}_i, \mathbf{v}_j \rangle^2}_{\text{squared projection length on } span(\mathbf{v}_1, \ldots, \mathbf{v}_k)}$$

$$C = Q \Lambda Q^{-1}$$

$\mathbf{v}_1, \ldots, \mathbf{v}_k$ are the first k leading eigenvectors of C

This is easy to see, again first work it out in the case when C is diagonal

PCA - Algorithm



- Zero centering the data takes $O(nm)$ (input scan)
- Covariance matrix: $O(m^2 n)$ – $O(m^2)$ values each takes $O(n)$ ops
- Eigen decomposition takes $O(m^3)$ time for a $m \times m$ matrix
- Transformation and dimensionality reduction require $n \times m \times k$ time
- Total runtime is $O(nm^2 + m^3)$.

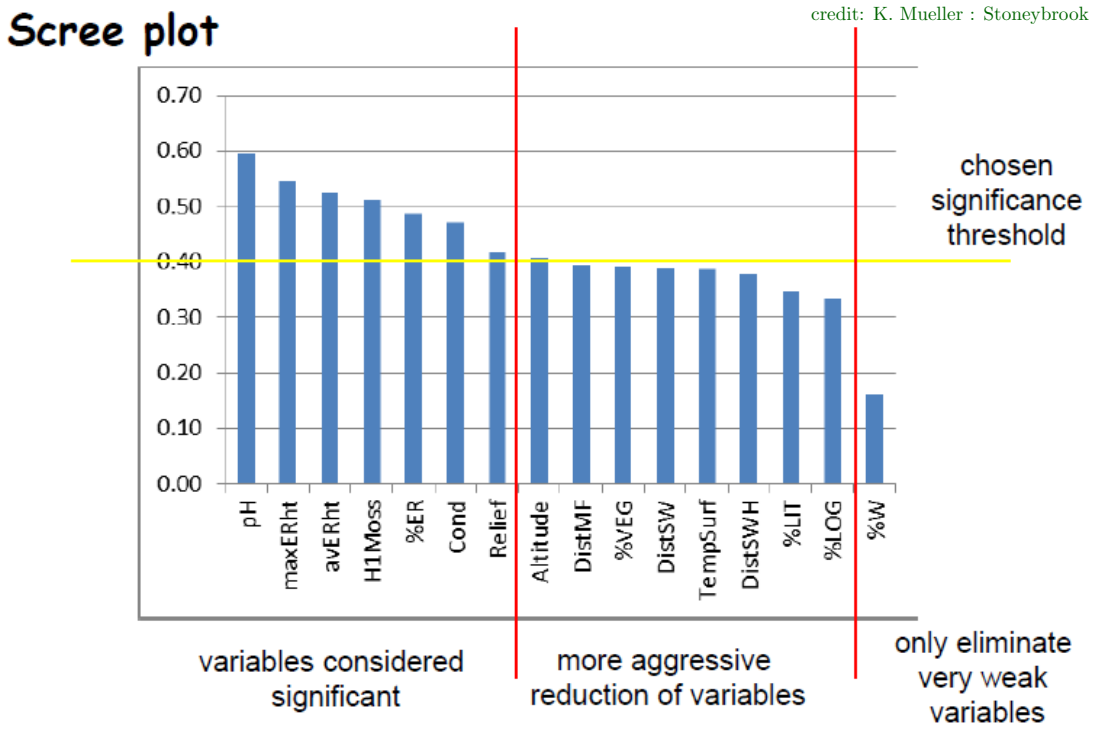
For large datasets computing C is computationally infeasible

Use SVD method to compute PCA to avoid computing C

Top eigenvectors of C can be computed using power iteration method

PCA - Number of components

Depends on the task at hand – Same question as number of clusters
Variance explained by $\mathbf{q}_i$ is equal to λ_i
Select k such that $\lambda_{k+1}, \ldots, \lambda_m$ are small

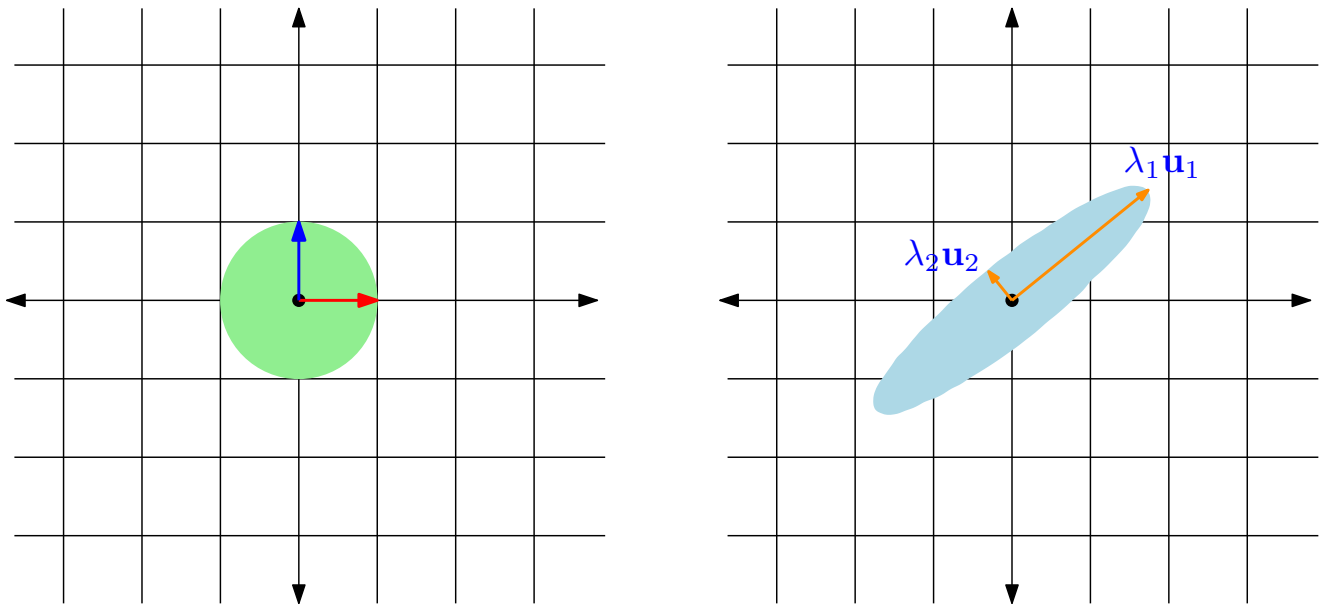


Or select k such that $\dfrac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geq (1-\epsilon)$ for $1 < \epsilon < 1$

Power Iteration Method

- The top eigenvector of A is the direction of max stretch
- $\lambda_1, \ldots, \lambda_m$ eigenvalues of $A \iff \lambda_1^r, \ldots, \lambda_m^r$ eigenvalues of A^r

Picture the transformation of the unit circle by A



The longest axis (major axis in $2d$) corresponds to the top eigenvector

The ellipse corresponding to A^k will be very long and very thin difference between axes will be amplified (λ_1^k vs λ_2^k)

Power Iteration Method

Power iteration method uses this intuition

- For a random unit vector $\mathbf{v}$ $A(A(A \ldots A(A(\mathbf{v}))))$ will be almost entirely in the direction of top eigenvector
- For large k A^k maps almost all unit vectors close to longest ellipse axis

---

Algorithm Power Iteration to compute top eigenvector of $A = X^T X$

$\mathbf{v}_0 \leftarrow$ RANDOM-UNIT-VECTOR() ▷ Generate random direction

$i \leftarrow 1$

while stopping criteria is not met **do**

  $\mathbf{v}_i \leftarrow A\mathbf{v}_{i-1}$

  $\mathbf{v}_i \leftarrow \mathbf{v}_i / \|\mathbf{v}_i\|$ ▷ Normalize to get unit vector

  $i \leftarrow i + 1$

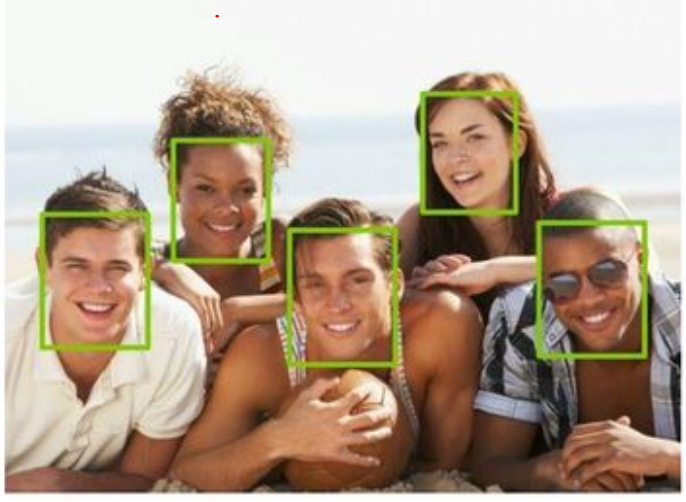
- Stop: when $\|\mathbf{v}_i\| - \|\mathbf{v}_{i-1}\| < \epsilon$
- Runtime depends on spectral gap λ_2/λ_1
- Can use repeated squaring to compute $A^k\mathbf{v}$

Power Iteration Method

- To compute second eigenvector after the first, $\mathbf{v}_1$
- Project A onto $\mathbf{v}_1$ and subtract it out
- The residual matrix $A' \leftarrow A - A\mathbf{v}_1\mathbf{v}_1^T$
- Row i of A' is $[\mathbf{a}_i - \langle \mathbf{a}_i, \mathbf{v} - 1 \rangle \mathbf{v}_i$
- The top eigenvector of A' is second leading eigenvector of A
- For the bottom eigenvector (e.g. of the Laplacian matrix, that we use for spectral clustering), use the inverse power iteration method
- This follows from the fact that eigenvalues of A^k are $\lambda_1^k, \lambda_2^k, \ldots,$
- For $k = -1$, eigen values of the inverse A^{-1} of A are $1/\lambda_1, 1/\lambda_2, \ldots,.$
- $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \implies 1/\lambda_1 \leq 1/\lambda_2 \leq \ldots \leq 1/\lambda_n$
- Thus, the power method on A^{-1} yields the smallest eigenpair
- With some linear algebra computing the inverse can be avoided
- To compute all eigenvectors the algorithm is called the QR algorithm

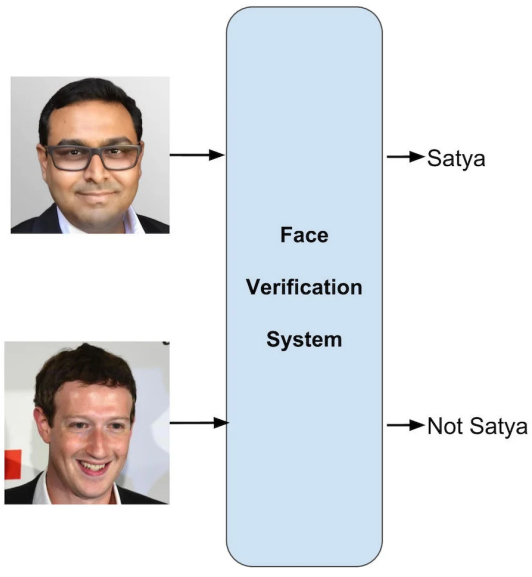
PCA Case Study: Eigenfaces

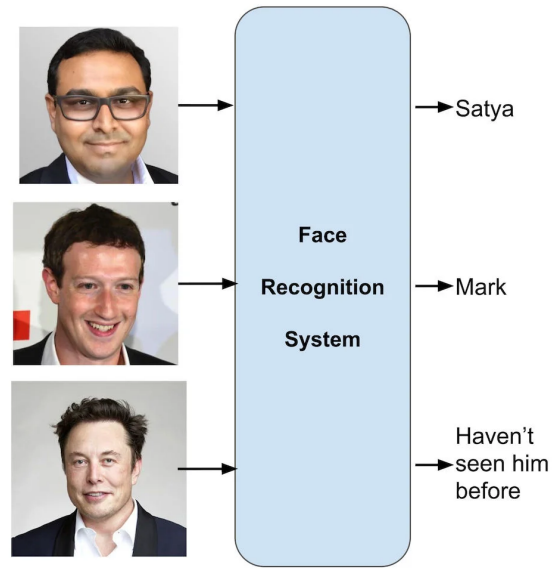
Classic application of PCA is image compression and face recognition



Detection finds the faces in images

J. Niebles & R. Krishna @ Stanford





source: learnopencv.com

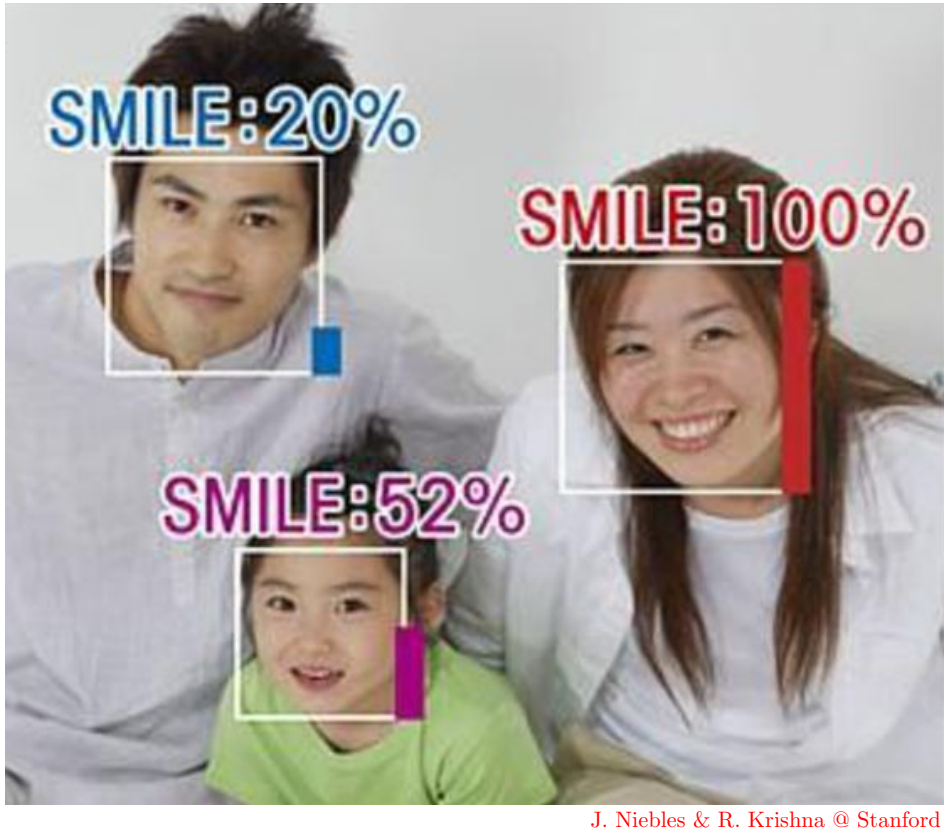
Face Detection and Recognition Applications

Surveillance

GILLES SABRIETHE NEW YORK TIMES/REDUX

Face Detection and Recognition Applications

Emotion and Expression Detection



J. Niebles & R. Krishna @ Stanford

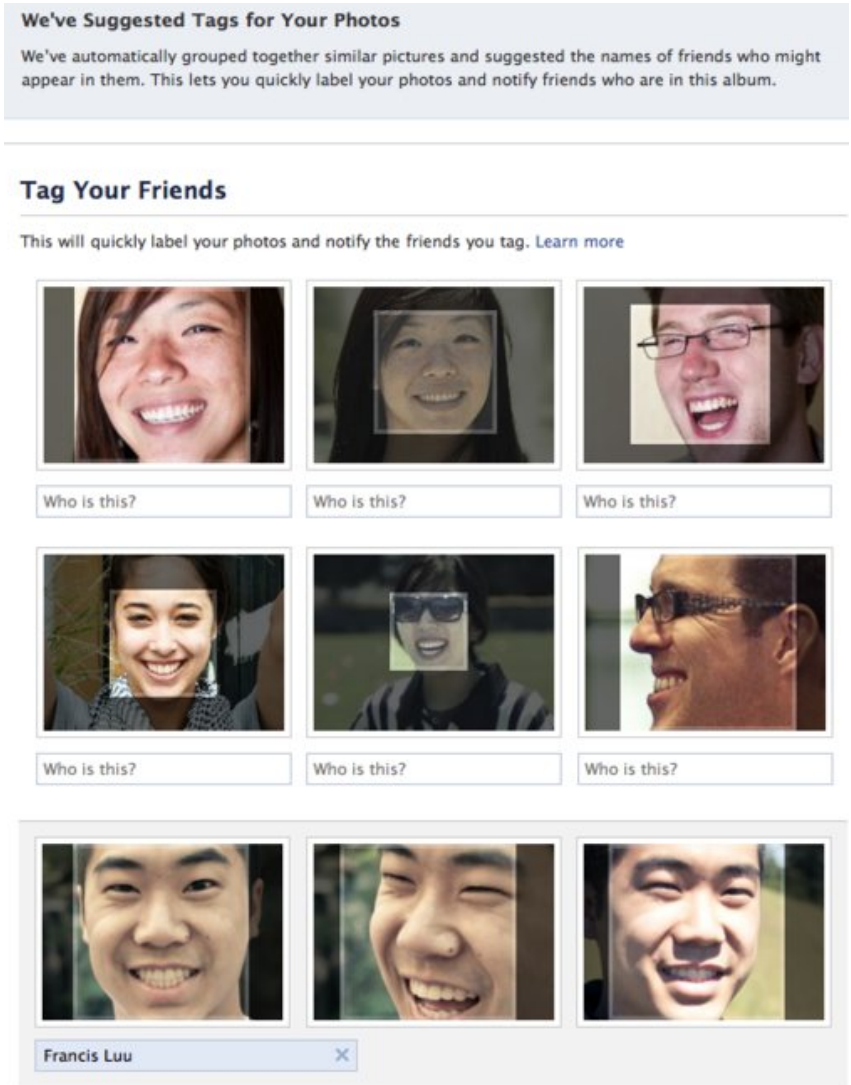
Face Detection and Recognition Applications

Photo Album Organization

J. Niebles & R. Krishna @ Stanford

Face Detection and Recognition Applications

Facebook Auto Tag Suggestions



PCA Case Study: Eigenfaces

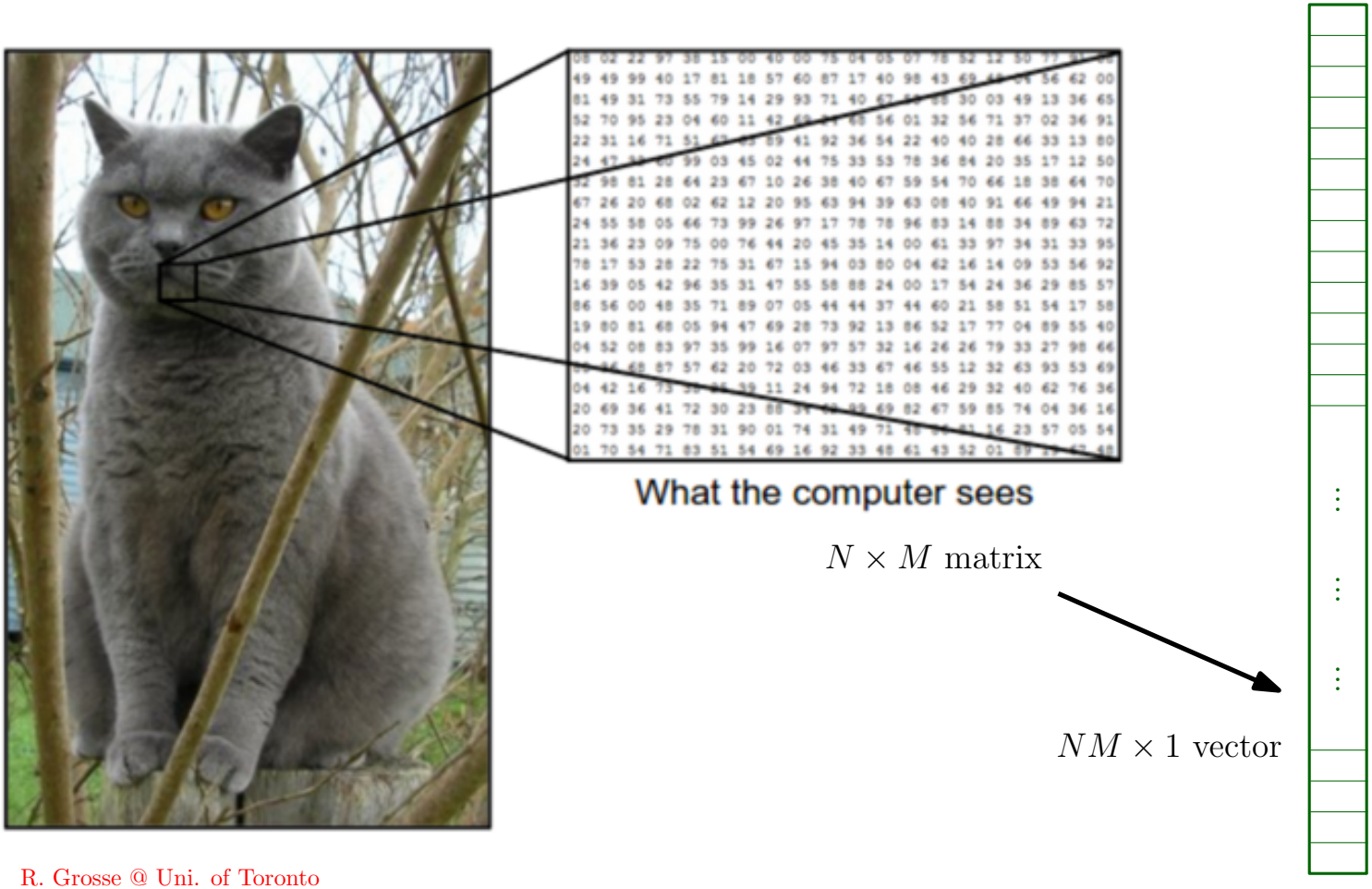
Input Images

Dataset For each face there should be a few training examples

All faces should be centered

PCA Case Study: Eigenfaces

Represent images by vectors



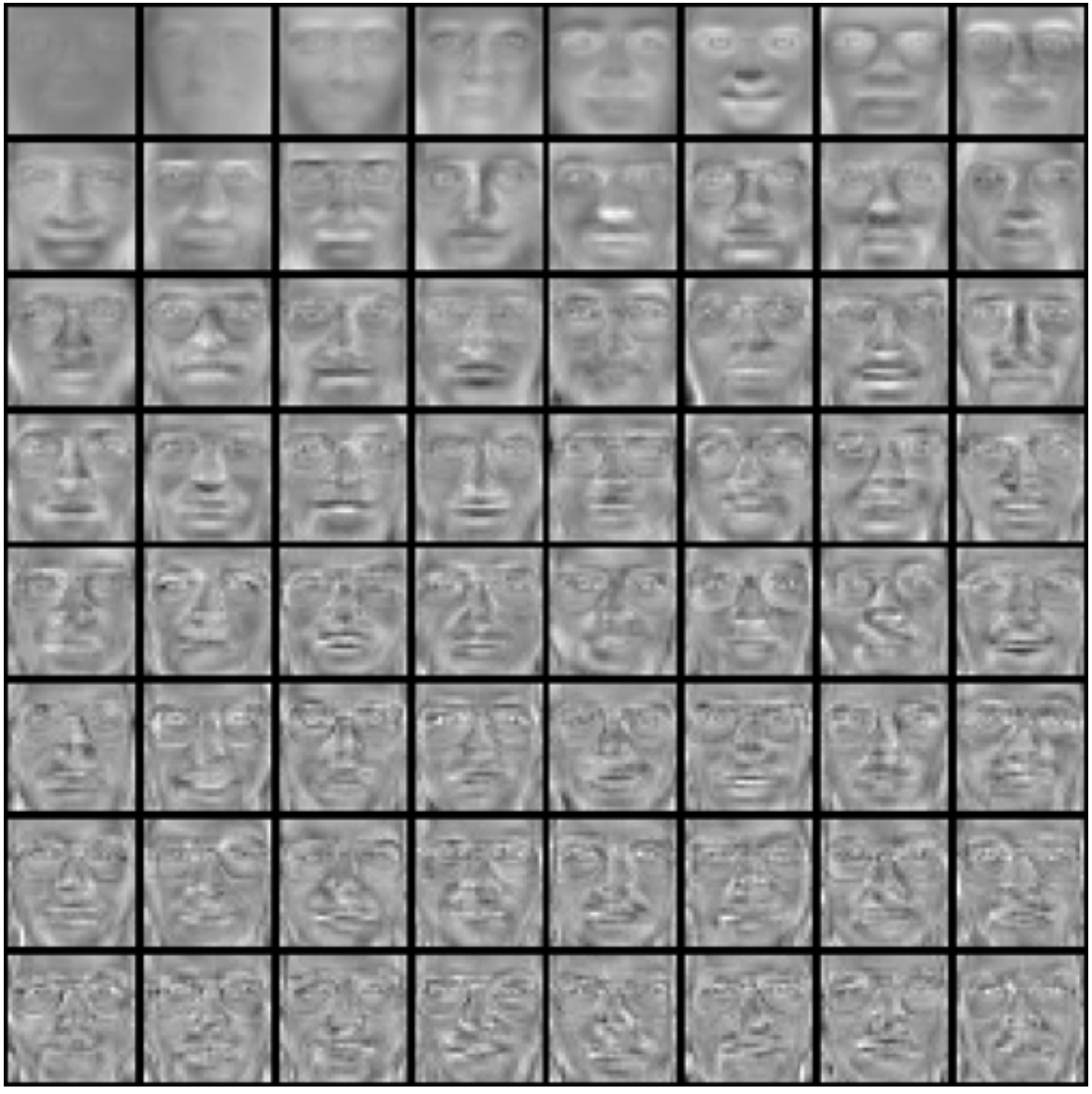
PCA Case Study: Eigenfaces

Mean Face

Mean face $\bar{\mathbf{x}}$

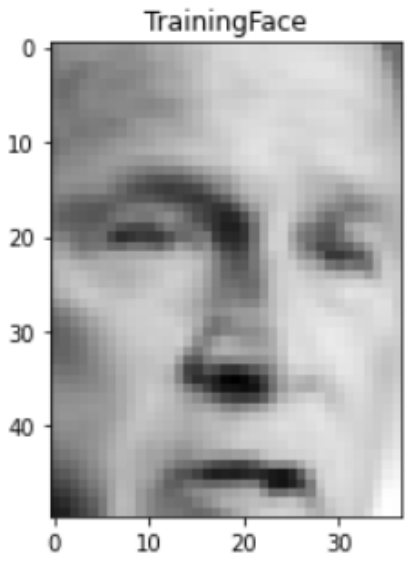
PCA Case Study: Eigenfaces

Top eigenvectors: $u_1, \ldots, u_k$ (visualized as images - eigenfaces)

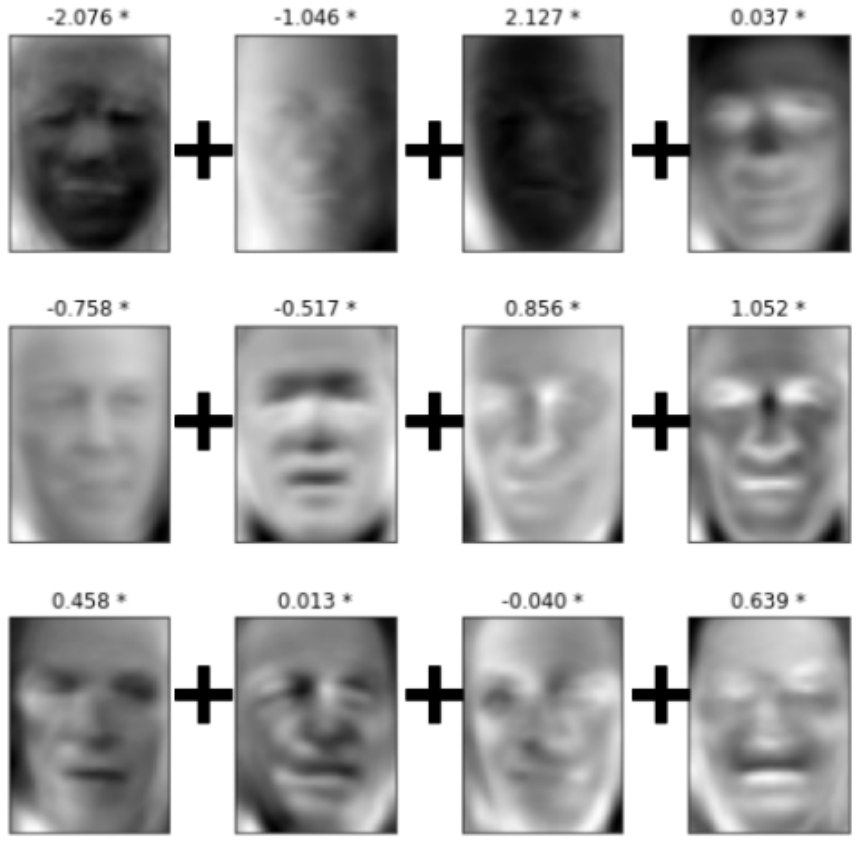


PCA Case Study: Eigenfaces

Represent a face as a linear combination of top k eigenfaces



=



PCA Case Study: Eigenfaces

Effect of number of principal components on reconstruction

$K=5$ $K=10$ $K=20$ $K=30$ $K=50$ $K=100$ $K=200$ $K=300$ $K=400$ $K=500$ $K=600$

PCA Case Study: Eigenfaces

Face Recognition:

- Subtract the mean face from the given (test) face
- Project onto the same k principal components
- Use k-nearest neighbors (by the new representations) and make a prediction based on that

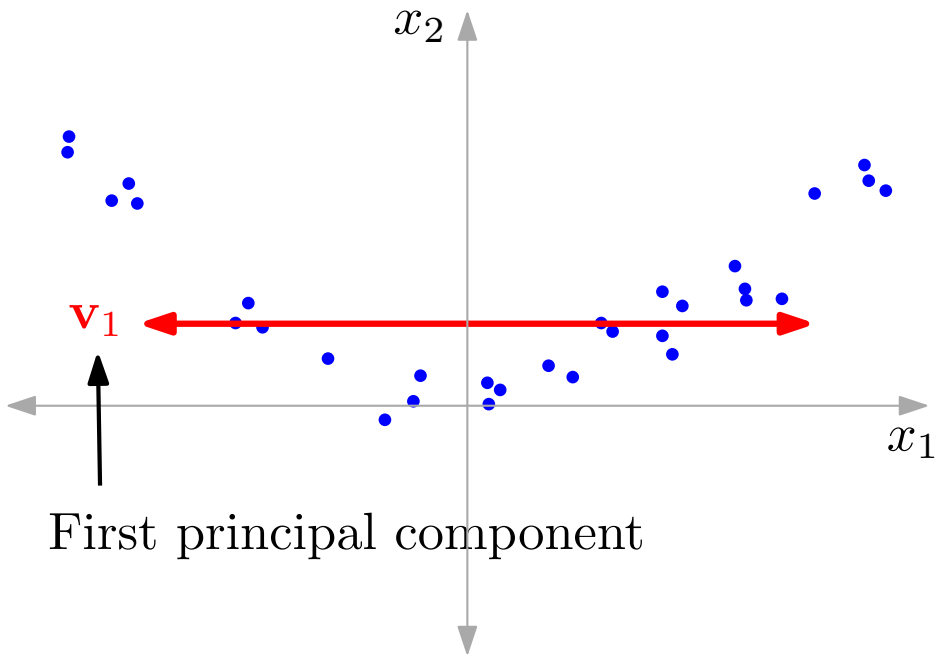
PCA Case Study: Eigenfaces

Face Detection:

- For a region R of the image, project R onto the principal components
- If the ℓ_2 distance of the new representation of R with R is not significant, then R is a face

PCA: Limitations

PCA does not capture non-linear relationships between attributes

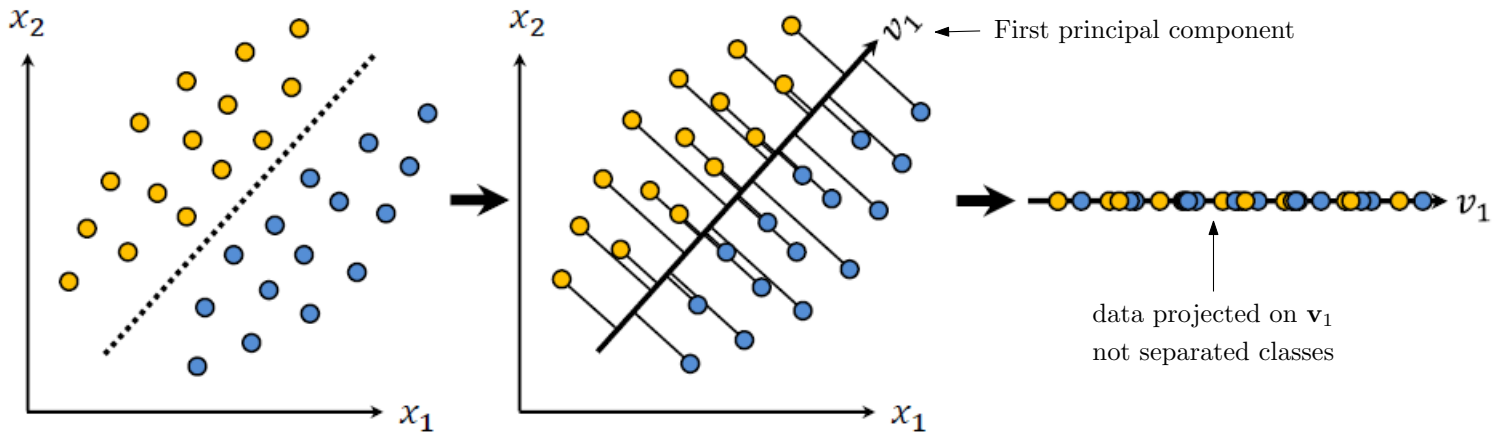


PCA: Limitations

PCA does not take into account any class labels (a completely unsupervised approach)

It does not necessarily help separate data based on classes

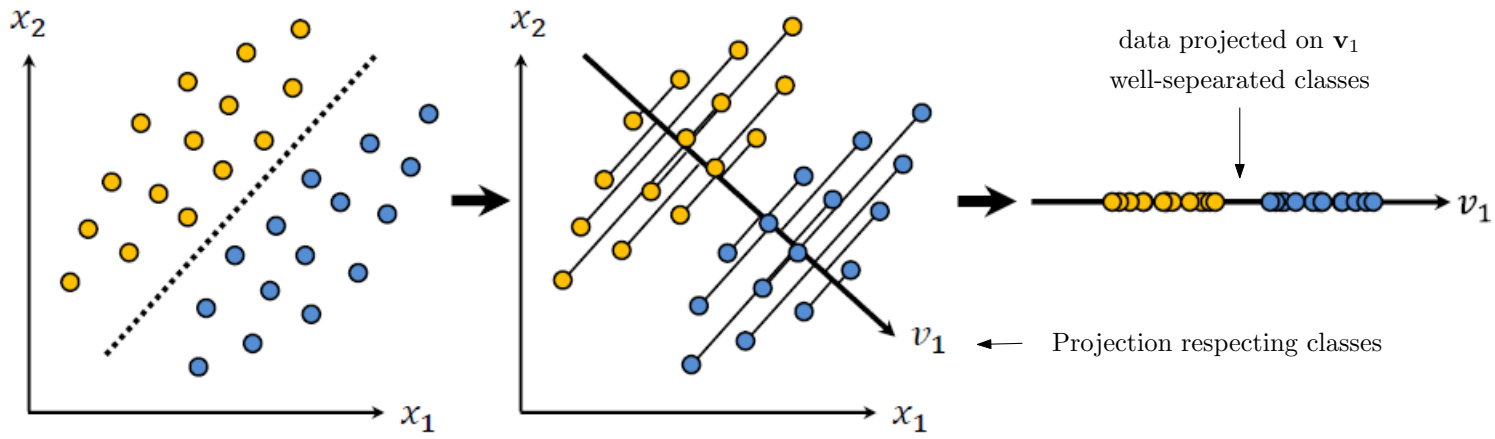
reduced dimensional data may not lead to better classification



Linear Discriminant Analysis

Linear Discriminant Analysis (LDA):

Seeks a projection that best discriminates the data



Linear Discriminant Analysis

Other dimensionality reduction methods that we will not study

Linear Methods include

- **Factor Analysis**
- **Independent Component Analysis:** Seeks a projection that preserves as much information in the data as possible

Non-linear methods include

- **Laplacian Eigenmaps**
- **ISOMAP**
- **Local Linear Embedding** Embedding to low dimensional manifolds