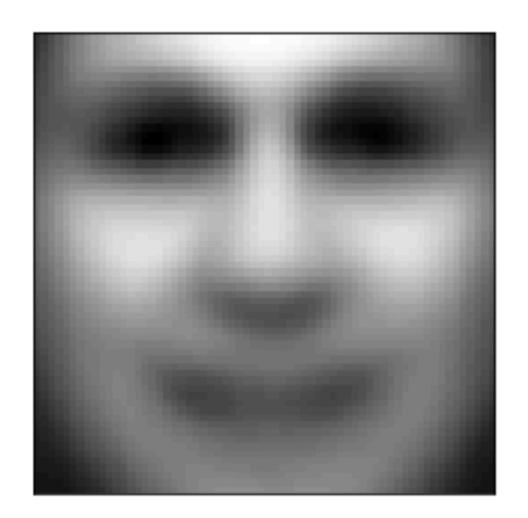
#### Random sample of 500 faces

	500 randomly sampled faces																		
9	25	T	U	8		•		1	<b>E</b>	<u>a</u>	4				<b>*</b>	0		1	0
•	•		Ì	•	1	•	Ð	1	Ø.	6	6	ø	•	**	0			1	
			5		1	1		1	重			<b>U</b>	•	1				•	
	0		•	0			25	Ø		1	9	•	0	•	1	3	•	•	
٨	A∰.	9	13	0	Ð	8	1	1	意		•	9	虁	1	3			Ď.	0
	堂	<b>Ø</b> .	2	•	1	•	1	9	重	*	1		1	0	è	0	0	1	1
		0	*	10	1	(4)	-	40h	•			1	意		Ó		1	T.	2
•	(Car		遵			9	-	1	色。	1	1	9		225	9		Ĭ,	(1)	
	1	9	1	1	Di.	1		Ð	Ö				e	•			1		TO YES
	ė			9	<b></b>	M	1	N.			Ď.	Ž		9	0	1		1	
S		Ú	<b>*</b>	•	2	23	1			9		J.	0	1		(9)	9	W.	
			9	Ø.	1	0	•	1		Ŏ.	9		ø,		•		1		0
ğ	(1)	0	1	100 m	0	180	D		9	8					35	THE STREET		9	1
	•	T	Ø.		•		1	T		1	藝		•		T	9			2
1		<b>O</b>		3	5	1		9	0	M.	0	連			Ð		1	1	1
	9	2	ē.			9		9	9	T		1	1	1		5	(E)		9
9	0	0	3	ė		0	<b></b>	置、	e	0	T		1				9		9
0					意			1	1			ø	Ð	<b>1</b> 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	9	Ð	6	THE	
Ð	1			9	<b>6</b>		Ch	ė.		1			1			<b>Q</b> .			
3.	1	1			營	3	0						•		9	9		Ø	
	Ď	1		1	1	9			5	ŧ	<b>.</b>	0	25	1	Đ.	9	D.	0	9
20	意	0	1	1		0	Ø.	包	T		9	<b>*</b>	<u> </u>	OF THE	查	1	6	曹	(0)
	•	重	•	O	躗	1	Ø.	A.L	0	1	8	1	ě.	<u>.</u>	<b>E</b>	1	1	1	<b>.</b>
	画	Ō	1	0	9	A.	Ú	1	1	1	8		•	重			<b>2</b> .	9	Ð
	()			- C	1		Đ,		Ú	E.		Ó	1		0	Bo	1	9	D.

## Eigenface algorithm: PCA on face images

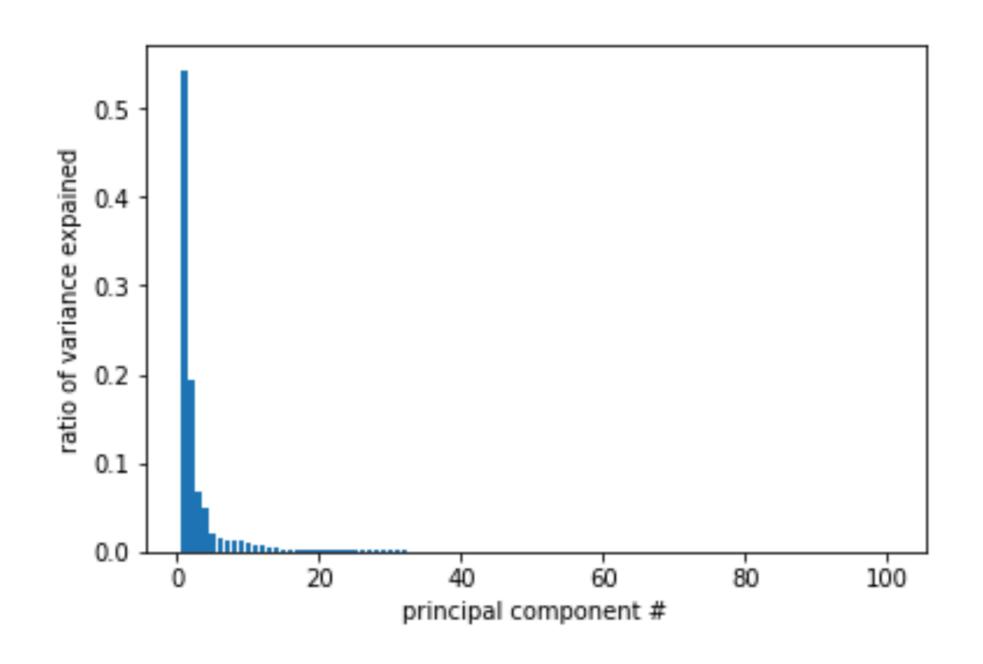
- 1. Load in grayscale images, all the with same width and height:  $I_1, I_2, \ldots, I_N$ .
- 2. Collapse each 2D image into 1D vectors  $\vec{x}_i$  (e.g. 16x16 2D image  $\Rightarrow$  256 1D vector). So, number of samples N = number of images. Variables are each of the pixels (e.g. if length( $\vec{x}_i$ ) is 256. M = 256). Like usual, A = [ $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_M$ ] (rows: images/samples, cols: 1D pixel value variables)
- 3. Center the images (subtract grand mean image):  $A_c = A \vec{\mu}$ , where  $\vec{\mu}$  is the column means of A (i.e. the mean pixel value at the same position across all images in the dataset).
- 4. Compute covariance matrix  $\Sigma$  then recover eigenvalues and eigenvectors.
- 5. Project images onto top k of principal components.

## Grand mean of 500 faces $(\vec{\mu})$

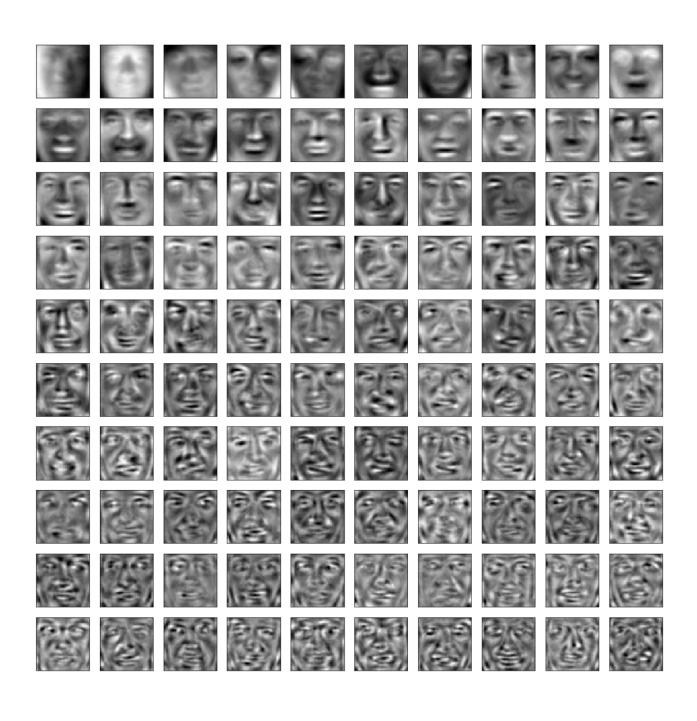


Because  $\vec{\mu}$  is 1D vector, I had to **reshape** it into a 2D image format (e.g. 256 1D vector -> 16x16 2D image)

## Variance explained by top eigenvalues/PCs



#### Face principal components: top k eVecs (Reshaped 1D -> 2D)



### Project one face image onto top K PCs



## Facial recognition using Eigenface algorithm

- 1. Do PCA on faces in your "known face database", get PC vectors (2 slides back).
- 2. Project **query** image you want to recognize into PCA space, keeping top k PCs (1 slide back, except no need to reconstruct data space). shape=(1, k)
- 3. Re-project images in your known face database PCA space one-by-one (1 slide back, except no need to reconstruct data space). shape=(1, k)
- 4. Compute the distance (scalar) between projected query image and current project database image (both (1, k) vectors).
- 5. If match is close enough (distance < threshold), you recognize the face! Otherwise repeat step 3.
- 6. If you exhaust all images in known faces dataset, you do not recognize the query face.

# Application: PCA on handwritten digits (Optdigits dataset)