Eigenface & Fishface for face detection

Contents

- Explore dataset
- Apply PCA
- Combine PCA & IDA (Fisher face)

Data set for PCA, from E-class

Dataset 1

- Training set: 20
- Test set: 40

Dataset 2

- Training set: 187
- Closed test set: 90
- Open test set: 200

Environment setup

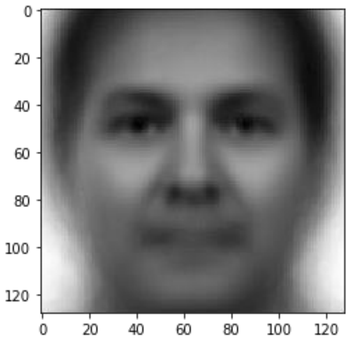
- Python in Win10
- Opencv 2
- Sklearn
    - PCA & IDA (Linear Discriminant Analysis)
- Jupyter notebook

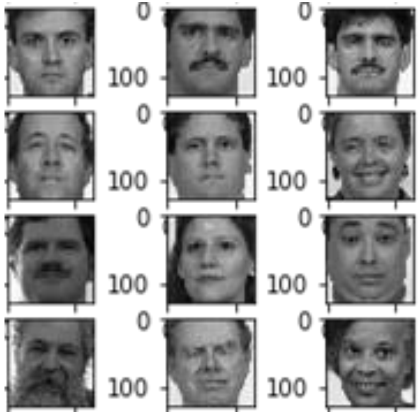
Data preprocessing

Step1: Read training images and test images

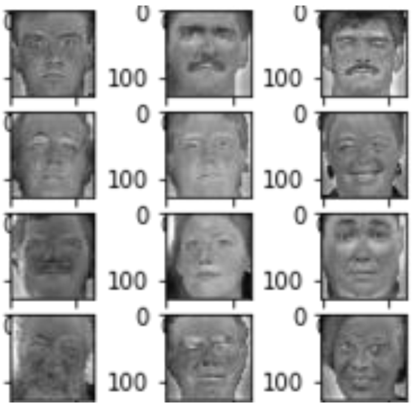
- Training set shape: (187, 16384)
- Close test set shape: (90, 16384)
- Open test set shape: (200, 16384)

Step2: Get mean face





Before minus mean face



After minus mean face

Singular Value Decomposition (SVD)

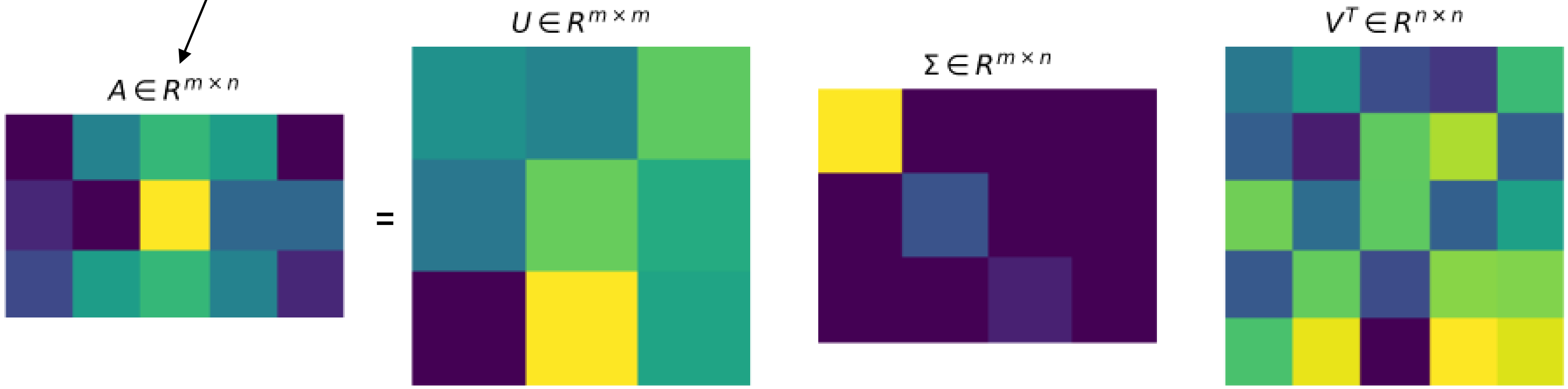
$$A_{mxn} = U_{mxm} \, S_{mxn} \, V^T{}_{nxn}$$

Where

$$U^TU = I_{mxm}$$

$$V^TV = I_{nxn}$$ (i.e. U and V are orthogonal)

Training faces



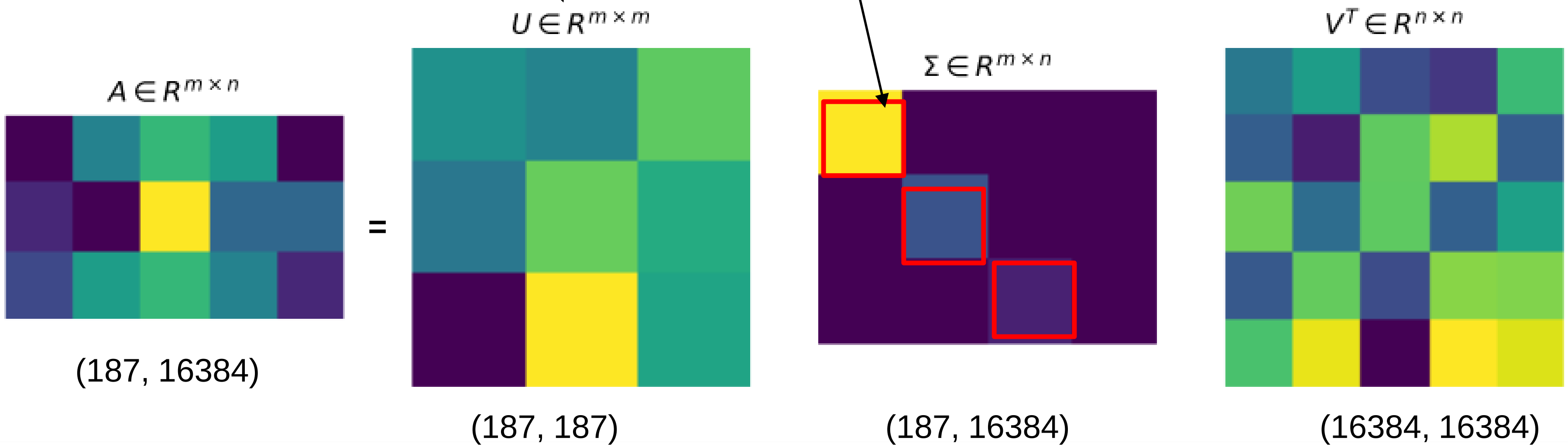
Singular Value Decomposition (SVD)

X_test_new = X_test • V^T

To transform test faces to new space

Training faces in new space

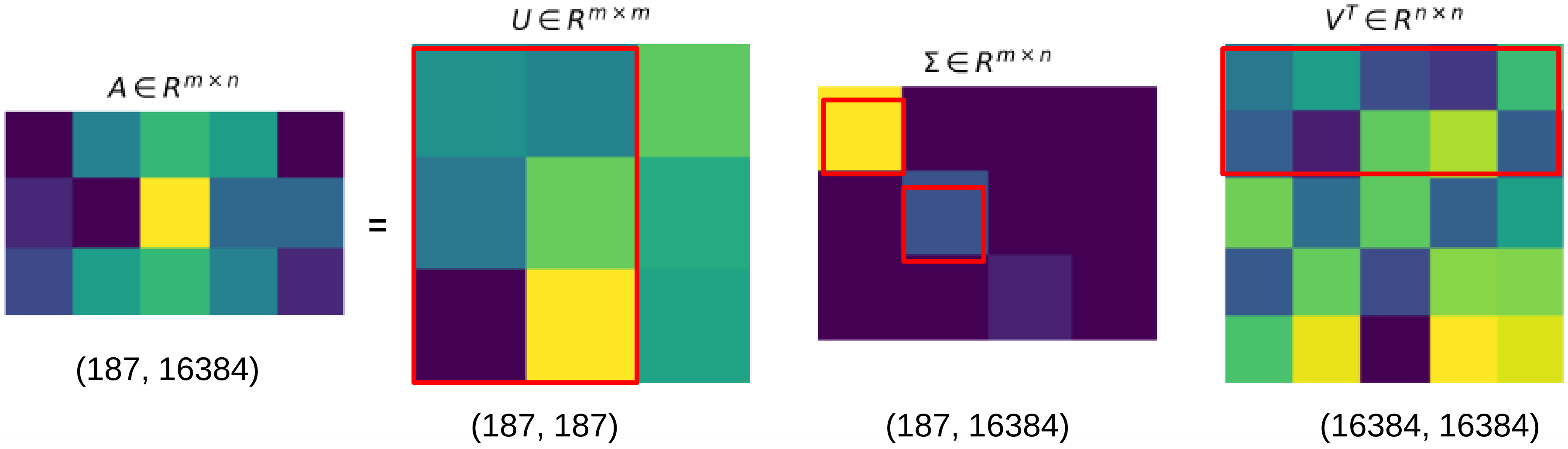
To be our eigenvalues



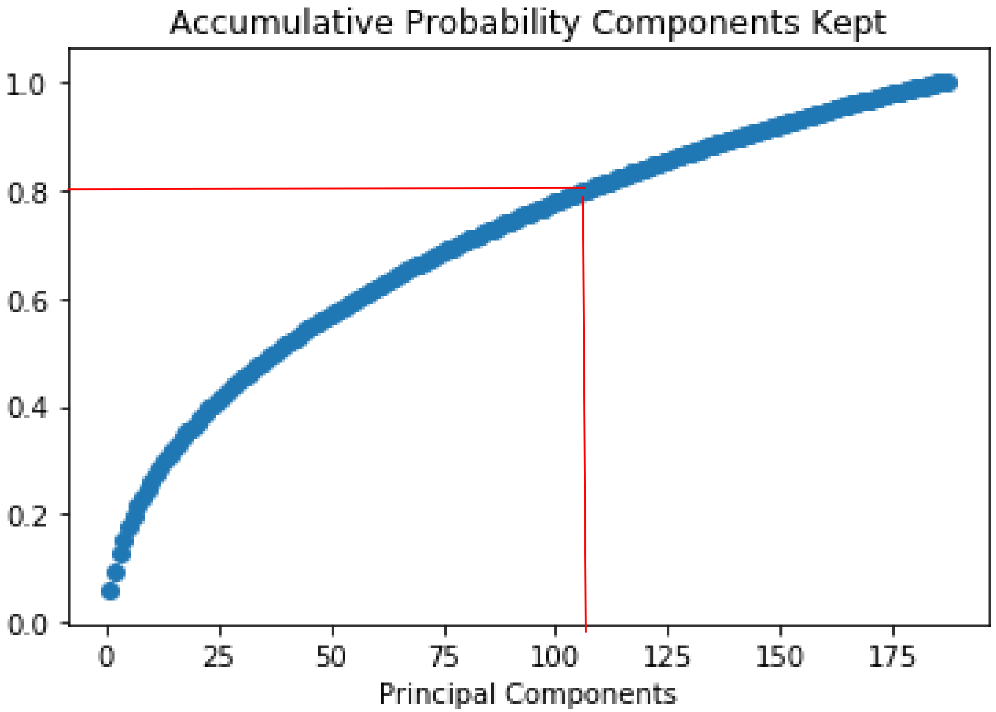
Singular Value Decomposition (SVD)

If choose two principle components.

X_test_new = X_test • V^T



Principle components



Principle Component and Eigenfaces

100%

94%

84%

70 %

57 %

51 %

32 %.

26 %

12 %

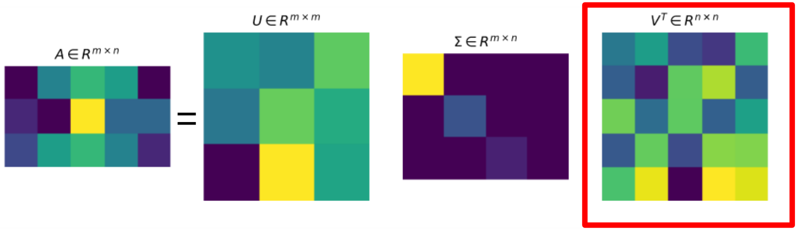
9 %

5.9 %

Principle Component and Eigenfaces

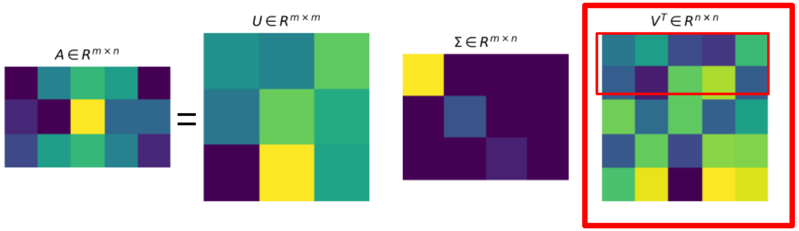
How to get these Eigenfaces?

Get Eigenfaces



(16384, 16384)

Get Eigenfaces

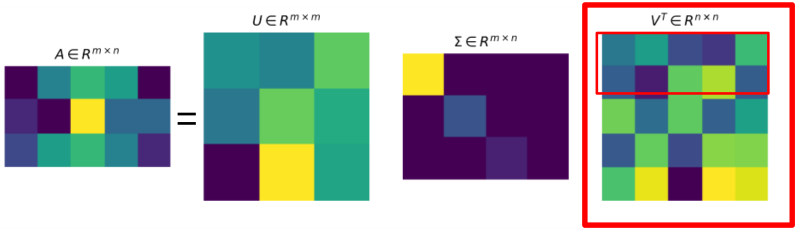


(16384, 16384)

Principle component

$V_{principle}$ (100, 16384)

Get Eigenfaces



(16384, 16384)

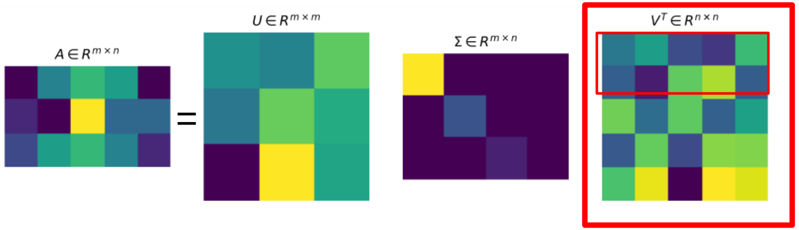
Principle component

$V_{principle}$ (100, 16384)

Transform test faces to new space

X_test_new =	**X_test**	$\cdot$	$V_{principle}^T$
(90,100)	(90,16384)		(16384,100)

Get Eigenfaces



(16384, 16384)

Principle component

$V_{principle}$ (100, 16384)

Transform test faces to new space

X_test_new =	**X_test**	$\cdot\ V_{principle}^T$
(90,100)	(90,16384)	(16384,100)

Get Eigenfaces

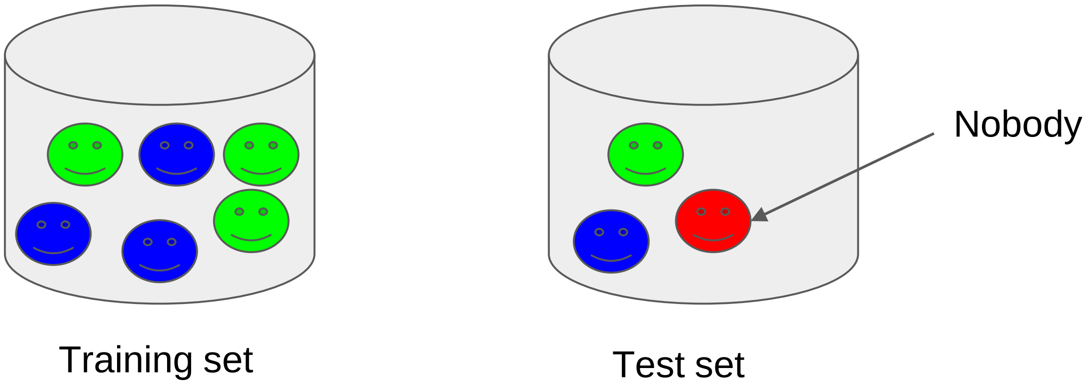
Eigenface	= X_test_new	$\cdot\ V_{principle}$
(90,16384)	(90,100)	(100,16384)

A test face

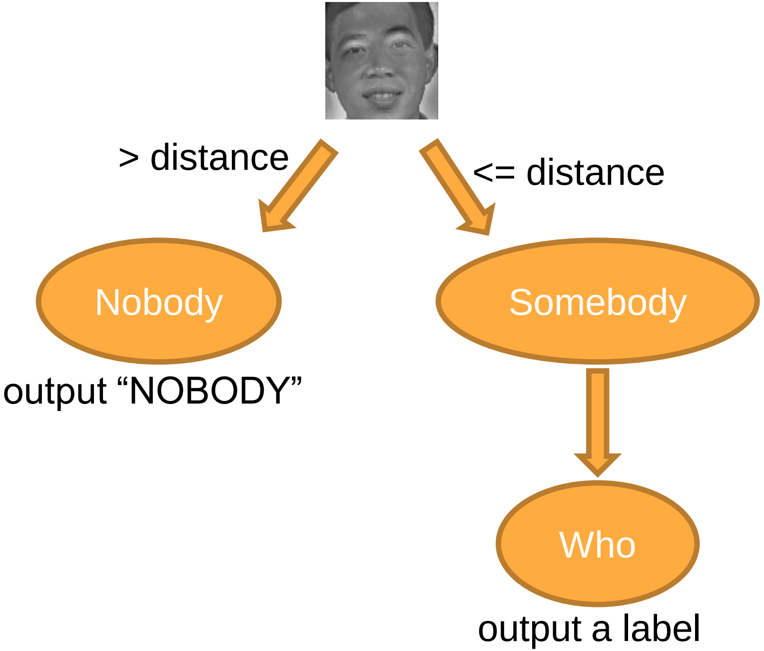
An eigenface

PCA Processing

- **Issue**: some faces (people) in **test set** do not exist in **training set**.



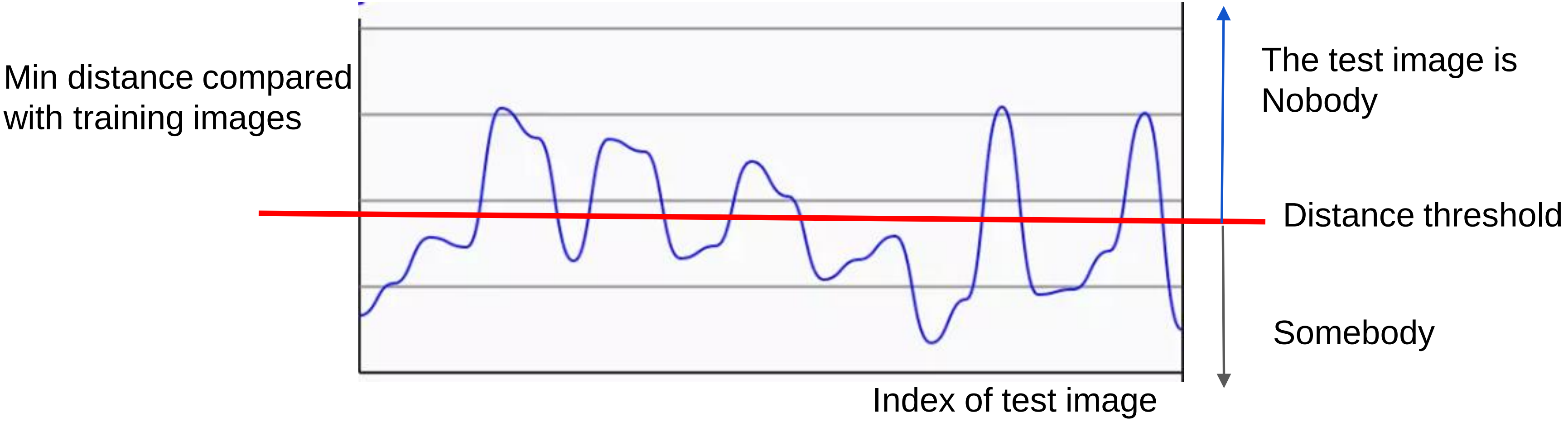
Prediction Strategy



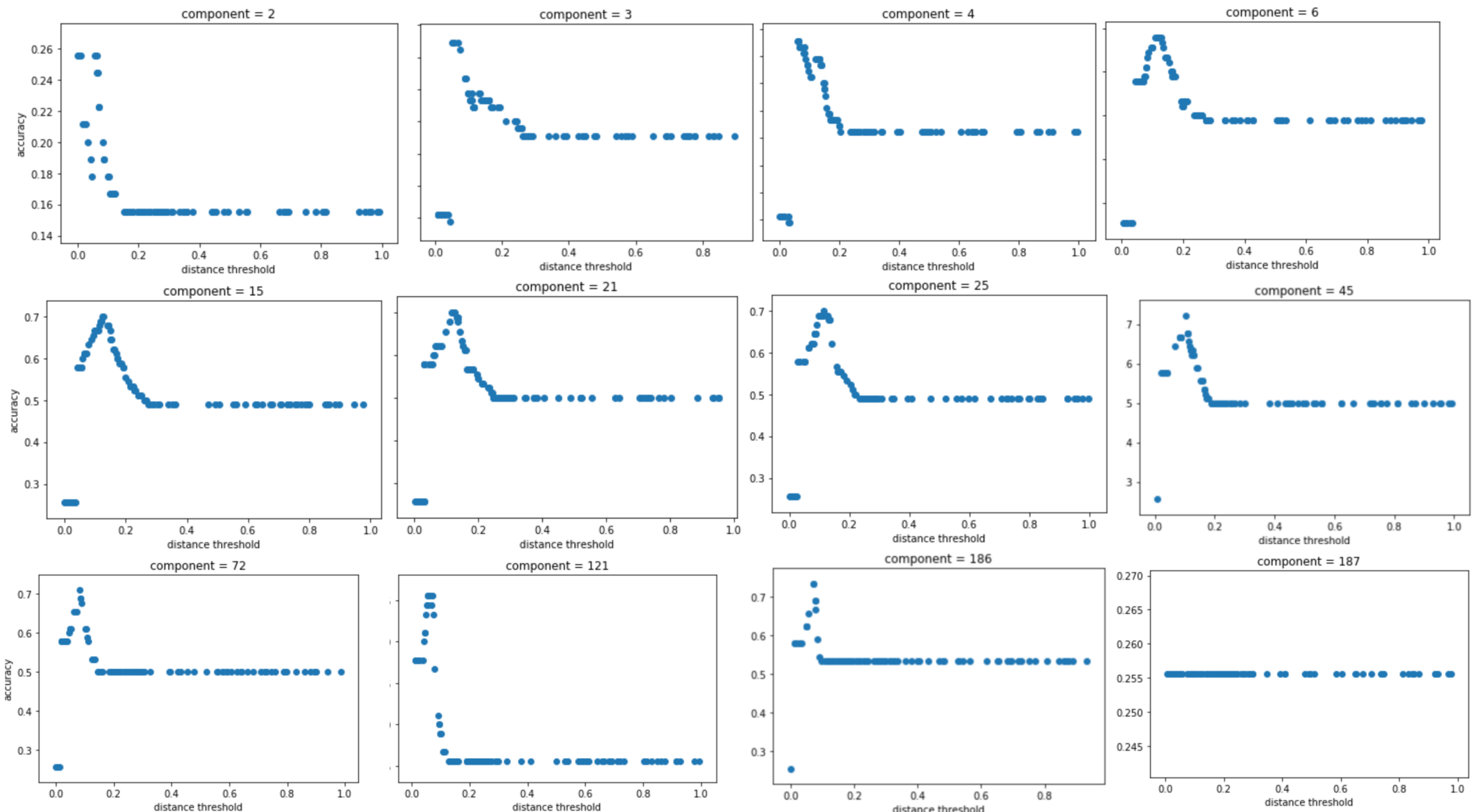
Step 7: Find a distance threshold

Euclidean Distance

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

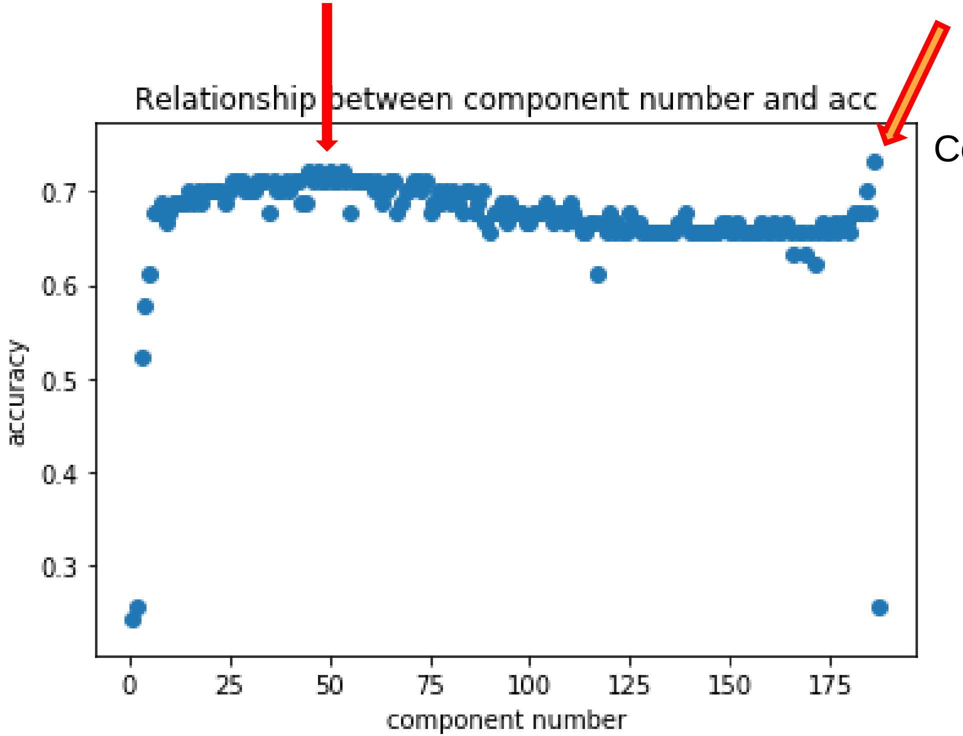


Best component number and distance threshold



Component Number and Accuracy

Component=45, acc = 0.722



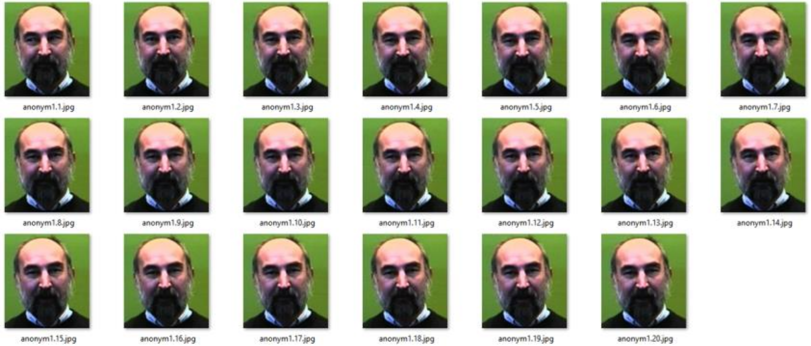
Component =186, acc = 0.733

Prediction results

		accuracy	Prediction time all faces / second	Prediction time per face / second	NOBODY
Dataset1	Test set (40)	90%	0.0156	0.000391	35.0 %
Dataset2	Close test set (90)	73.3 %	0.085	0.000943	25%
	Open test set (200)	**100 %**	0.2078	0.001039	**100%**

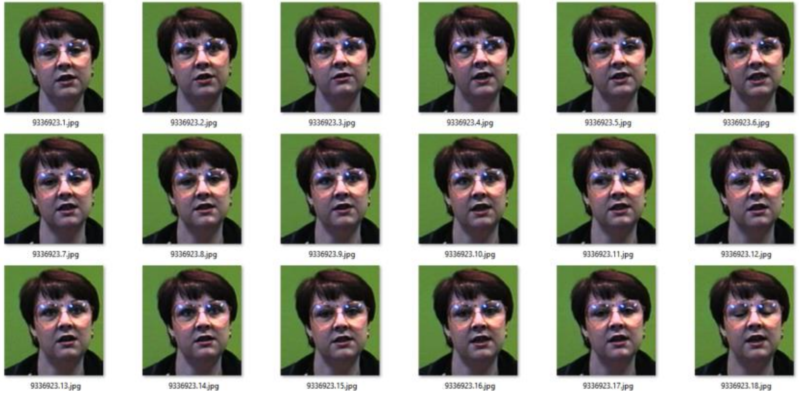
Fisher Face (PCA & LDA)

Why do not use the previous dataset?



New Data set

- 152 people totally (38 Nobody)
- **> 10 faces / person**
- Training set: 2120 faces
- Test set: 306 Faces



https://cswww.essex.ac.uk/mv/allfaces/faces94.html

How to combine PCA and LDA?

Combine PCA & IDA

Stratege:

```
pca = PCA(n_components=n_component_pca)
pca.fit(self.X_train)
```

```
lda = LinearDiscriminantAnalysis(n_components=n_component_lda)
self.train_transformed = lda.fit_transform(pca.transform(self.X_train), self.y_train)
self.test_transformed = lda.transform(pca.transform(self.X_test))
```

Combine PCA & IDA

Stratege:

```
pca = PCA(n_components=n_component_pca)
pca.fit(self.X_train)

lda = LinearDiscriminantAnalysis(n_components=n_component_lda)
self.train_transformed = lda.fit_transform(pca.transform(self.X_train), self.y_train)
self.test_transformed = lda.transform(pca.transform(self.X_test))
```

Combine PCA & IDA

Stratege:

```
pca = PCA(n_components=n_component_pca)
pca.fit(self.X_train)

lda = LinearDiscriminantAnalysis(n_components=n_component_lda)
self.train_transformed = lda.fit_transform(pca.transform(self.X_train), self.y_train)
self.test_transformed = lda.transform(pca.transform(self.X_test))
```

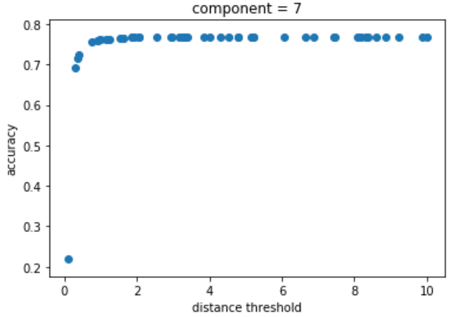
Combine PCA & IDA

Stratege:

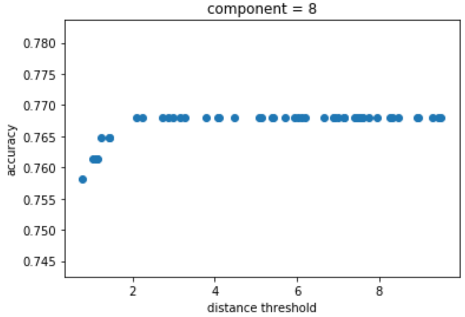
```
pca = PCA(n_components=n_component_pca)
pca.fit(self.X_train)

lda = LinearDiscriminantAnalysis(n_components=n_component_lda)
self.train_transformed = lda.fit_transform(pca.transform(self.X_train), self.y_train)
self.test_transformed = lda.transform(pca.transform(self.X_test))
```

In new space

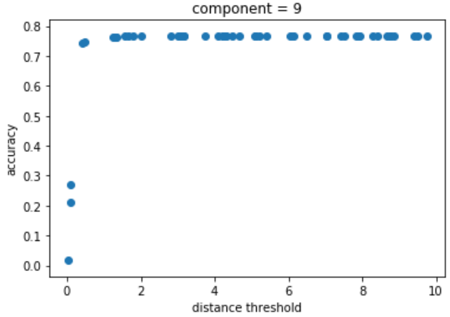
Find the best hyperparameter.

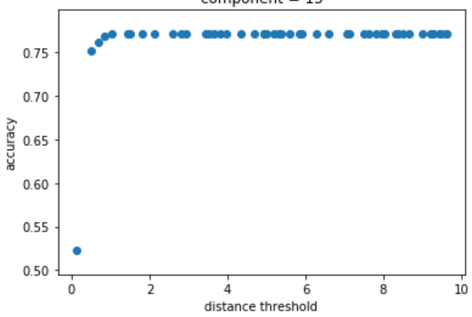


2.104522505274958 0.7679738562091504

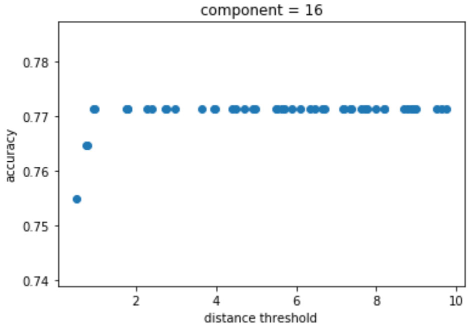


1.5804497128468065 0.7679738562091504

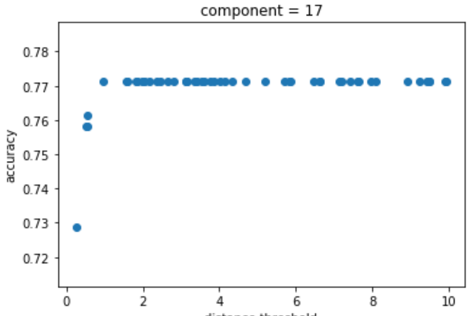


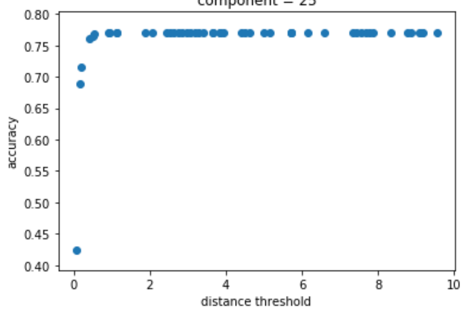


0.9551473875690175 0.7712418300653595

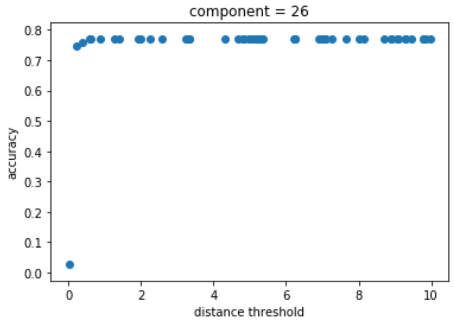


0.9810070455690301 0.7712418300653595

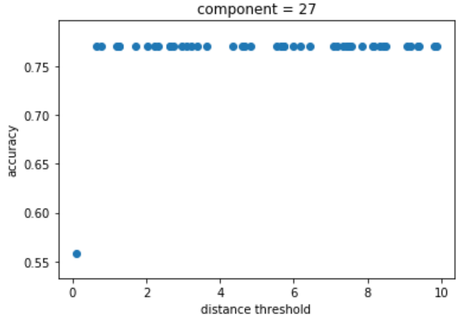




0.5927895765791502 0.7712418300653595



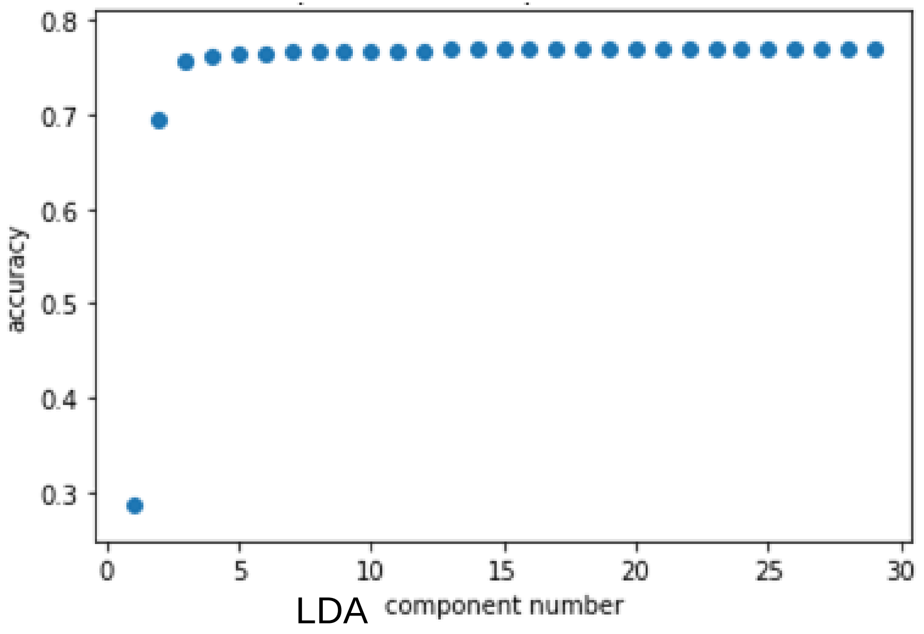
0.6401646844310227 0.7712418300653595



Result of PCA & LDA

PCA **30** component, keep **80%** face information.

LDA 13 component, 77.12% accuracy



Prediction results (PCA & LDA)

	accuracy	Prediction time per face / second	NOBODY
test set (306)	77.12%	0.008311	25%

Training data: 2120

Conclusion

	Method		accuracy	Prediction time all faces / second	Prediction time per face / second	NOBODY
Dataset1	PCA	Test set (40)	**90%**	0.0156	0.000391	35.0 %
Dataset2	PCA	Close test set (90)	73.3 %	0.085	0.000943	25%
	PCA	Open test set (200)	**100 %**	0.2078	0.001039	**100%**
Dataset3	PCA&LDA	Test set(306)	77.12%	2.5433	0.008311	25%

Thank you!