CS6375: Machine Learning

Gautam Kunapuli

Principal Component Analysis

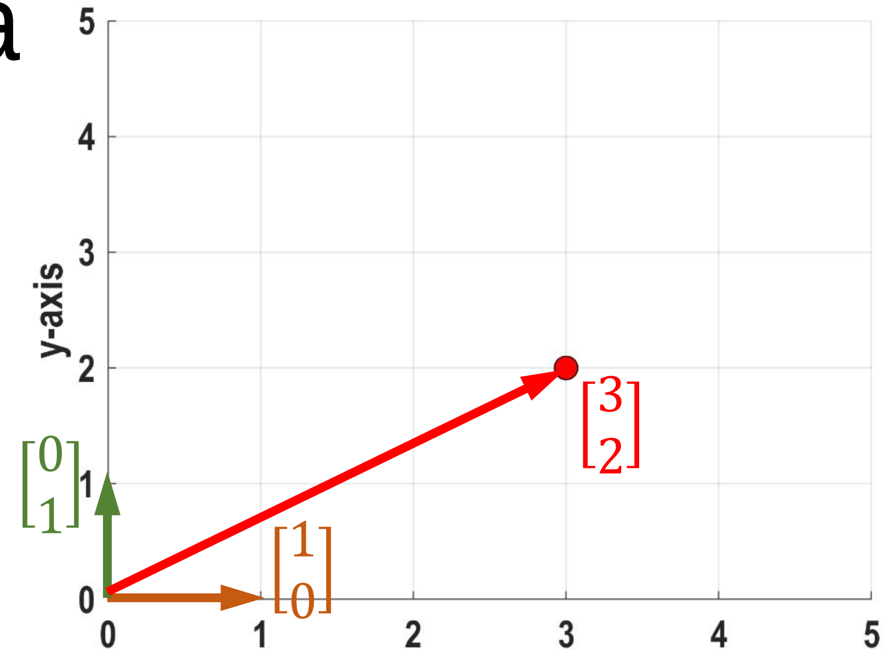
THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science

A Review of Linear Algebra

Every point in space can be expressed as a **linear combination** of **standard basis (or natural basis)** vectors

$$\begin{bmatrix}3\\2\end{bmatrix} = 3 \cdot \begin{bmatrix}1\\0\end{bmatrix} + 2 \cdot \begin{bmatrix}0\\1\end{bmatrix}$$

The **components of the vector** tell you how far along each direction of the basis you must travel to describe your point.

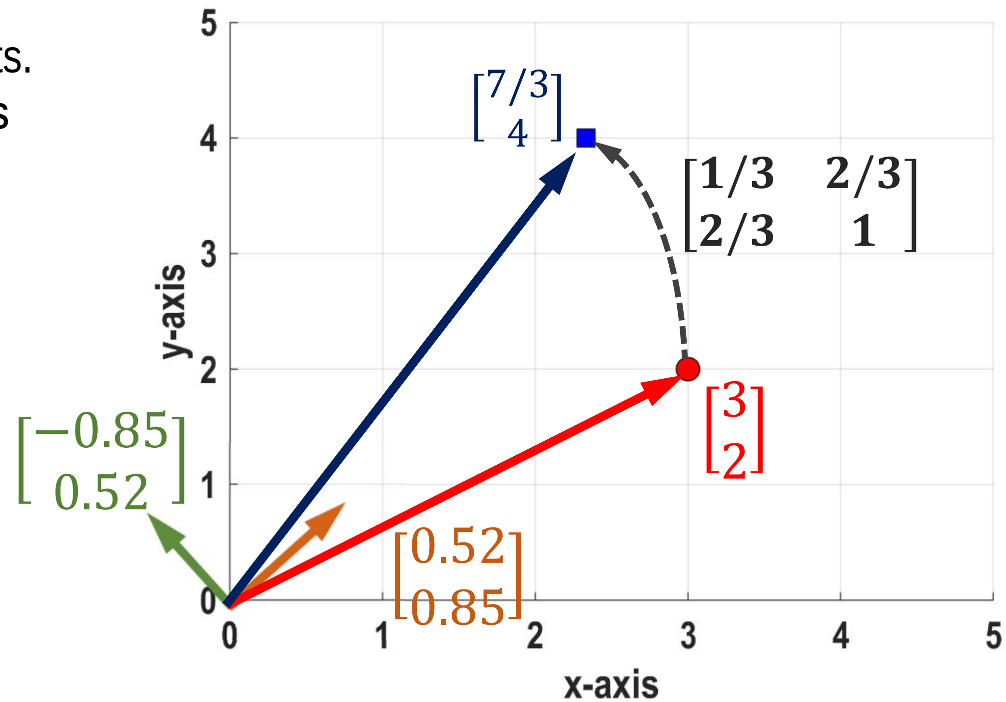


A matrix can be used to transform (rotate and scale) points. This corresponds to a change of basis. The **eigenvectors** describe the new basis of the transformation matrix. For instance, data points transformed by a matrix A

$$\begin{bmatrix}\mathbf{1/3} & \mathbf{2/3}\\ \mathbf{2/3} & \mathbf{1}\end{bmatrix}$$

can be described in terms of its eigenvectors

$$\begin{bmatrix}7/3\\4\end{bmatrix} = 3.28 \cdot \begin{bmatrix}0.52\\0.85\end{bmatrix} - 1.5 \cdot \begin{bmatrix}-0.85\\0.52\end{bmatrix}$$

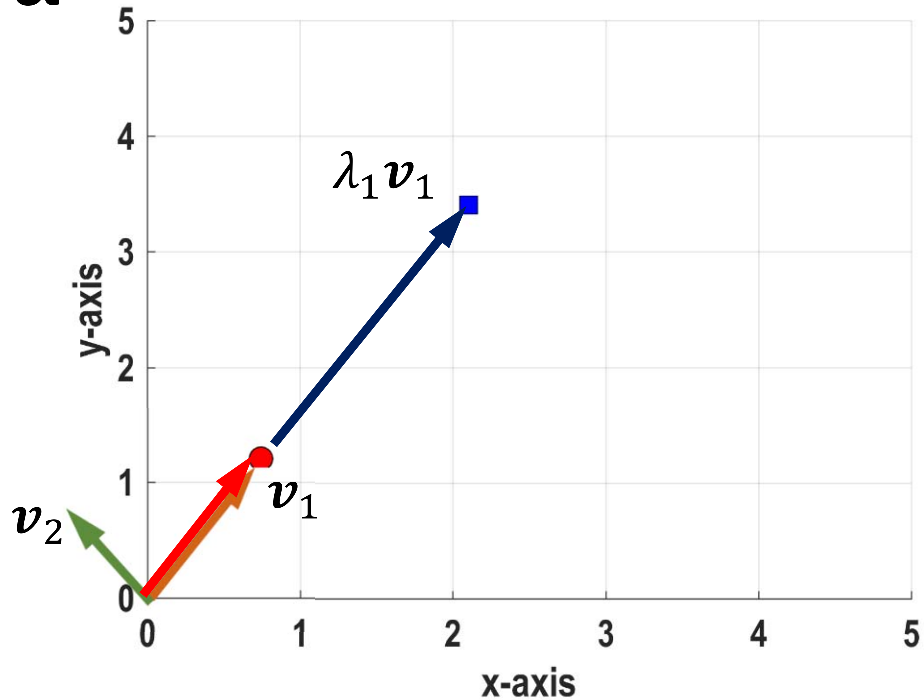


A Review of Linear Algebra

What happens when we apply the transformation to the eigenvectors themselves?

The **directions of eigenvectors** themselves remain **unchanged** under the transformation! They only get **rescaled**; the amount of rescaling is captured by the **eigenvalue.**

$$A\boldsymbol{v}_1 = \lambda_1 \boldsymbol{v}_1$$
$$A\boldsymbol{v}_2 = \lambda_2 \boldsymbol{v}_2$$



In matrix form:

$$A \underbrace{\begin{bmatrix} | & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 \\ | & | \end{bmatrix}}_{V} = \underbrace{\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}}_{\Lambda} \underbrace{\begin{bmatrix} | & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 \\ | & | \end{bmatrix}}_{V}$$

The prefix eigen- is adopted from the German word *eigen* for "proper“ or "characteristic". Eigenvalues and eigenvectors have a wide range of applications, for example in stability analysis, vibration analysis, atomic orbitals, **facial recognition**, and matrix diagonalization.

The eigenvectors are orthonormal, that is, they have magnitude 1 and are **perpendicular to each other**; which is written as $V^T V = I$ (and thus, $V^T = V^{-1}$ for an orthonormal matrix). So we have

$$A = V^T \Lambda V$$

This is known as the **eigen-decomposition** of a matrix.

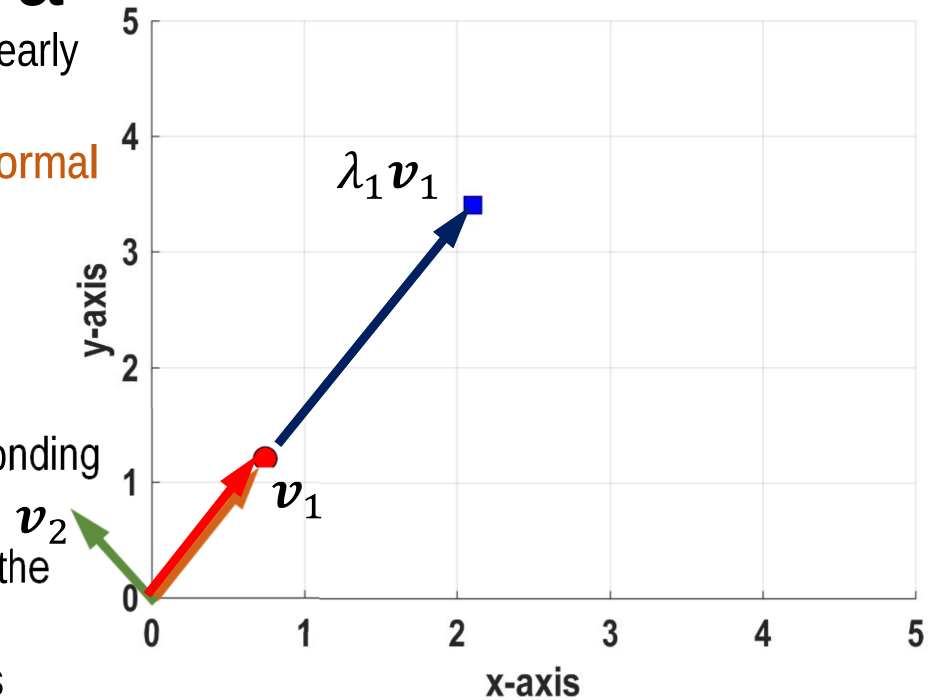
Eigenvalues and eigenvectors visualized:
http://setosa.io/ev/eigenvectors-and-eigenvalues/

A Review of Linear Algebra

If the transformation $A \in \mathbb{R}^{d \times d}$ is symmetric, then it has d linearly independent eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$ corresponding to d real eigenvalues; moreover, it has n linearly independent orthonormal eigenvectors

- $\boldsymbol{v}_i^T \boldsymbol{v}_j = 0, \forall\, i \neq j$
- $\boldsymbol{v}_i^T \boldsymbol{v}_i = 1, \forall\, i$

- There can be zero, negative or multiple eigenvalues corresponding to a matrix.
- The orthonormal eigenvectors form a basis of $\mathbb{R}^n$ (similar to the standard coordinate axes)
- A symmetric matrix is positive definite if and only if **all of its eigenvalues are positive**



Examples:

- The 2 x 2 identity $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ has all eigenvalues equal to 1 (positive definite) with orthonormal eigenvectors $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- The matrix $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ has eigenvalues 0 and 2 with orthonormal eigenvectors $\begin{bmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ and $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$
- The matrix $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ has eigenvalues 1 and 3 with orthonormal eigenvectors $\begin{bmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ and $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$

Principal Component Analysis: Intuition

Any point $\boldsymbol{x} \in \mathbb{R}^d$ can be written using the eigenvector basis of a (symmetric) matrix

$$\boldsymbol{x} = \sum_{i=1}^{d} c_i \boldsymbol{v}_i$$

- the weight c_i (also, co-ordinate) is the projection of $\boldsymbol{x}$ along the line given by the eigenvector $c_i = \boldsymbol{v}_i^T \boldsymbol{x}$
- Transformations using a matrix can be written as $A\boldsymbol{x} = V^T \Lambda V \boldsymbol{x}$

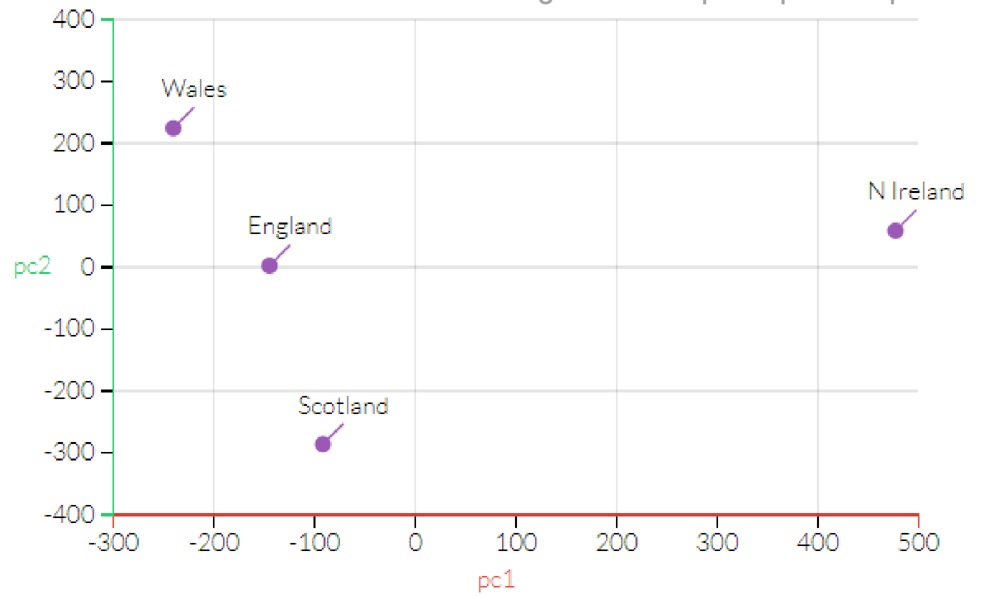
Intuition: Can we use **fewer eigen-vectors** to obtain a low-dimensional representation that approximates the transformed data point well-enough to be useful?

original data: 17 features/dimensions

	England	N Ireland	Scotland	Wales
Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227
Cereals	1472	1494	1462	1582
Cheese	105	66	103	103
Confectionery	54	41	62	64
Fats and oils	193	209	184	235
Fish	147	93	122	160
Fresh fruit	1102	674	957	1137
Fresh potatoes	720	1033	566	874
Fresh Veg	253	143	171	265
Other meat	685	586	750	803
Other Veg	488	355	418	570
Processed potatoes	198	187	220	203
Processed Veg	360	334	337	365
Soft drinks	1374	1506	1572	1256
Sugars	156	139	147	175

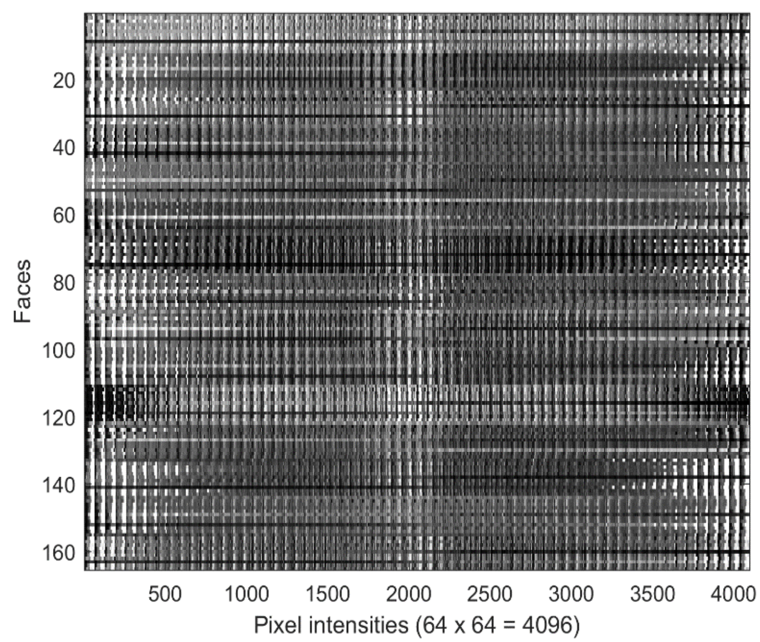
Note that in this example, contrary to common convention, features are rows and training examples are columns.

transformed data: 2 dimensions using the first 2 principal components



Example: Face Recognition

Example: Develop a model to **quickly** and **efficiently identify people** from photographs, videos etc. in a robust manner (that is, stable and reliable under changing facial expressions, orientations, lighting conditions)



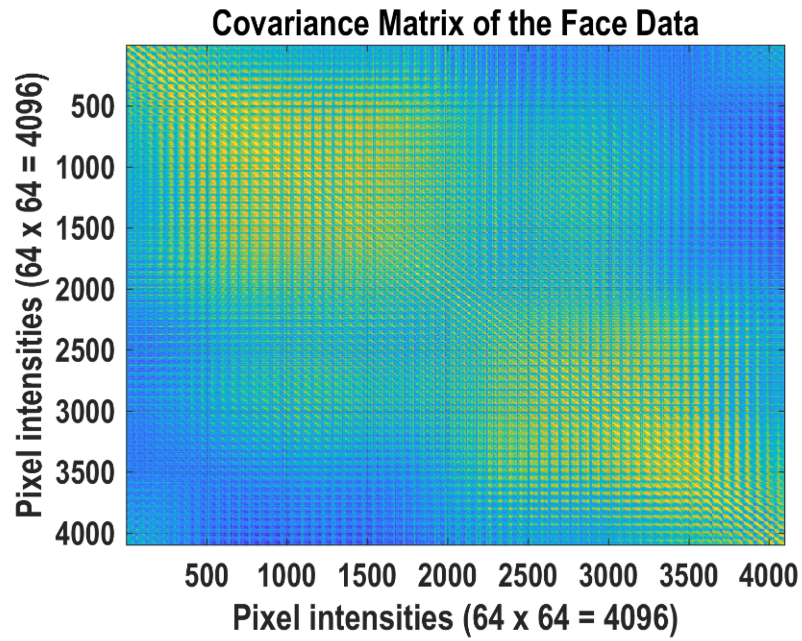
Let's suppose that our data is a collection of images of the faces of individuals

- The goal is, given the "training data" of n images, to **correctly match new images** to the training data
- Each image is an $s \times s$ array of pixels: $\boldsymbol{x}_i \in \mathbb{R}^d, d = s^2$
- As with digit recognition, construct the matrix $X \in \mathbb{R}^{n \times d}$, whose i-th row is the i-th **vectorized image**
    - **pre-process** to subtract the mean from each image

Principal Component Analysis

- Can be used to reduce the dimensionality of the data while still maintaining a good approximation of the sample mean and variance
- Can also be used for **selecting good features** that are combinations of the input features
- Unsupervised – just finds a good representation of the data in terms of combinations of the input features

The covariance matrix of the data $X^T X$ is 4096 x 4096, as each image has 4096 features! Can we represent each face using **significantly fewer features** than 4096?



covariance matrix is symmetric, positive semi-definite; this means all the eigen-values will be positive or zero

Principal Component Analysis identifies the principal components in the sample covariance matrix of the data, $X^T X$

(*note that since our data is #examples (n) x features (d), the covariance matrix will be $d \times d$*)

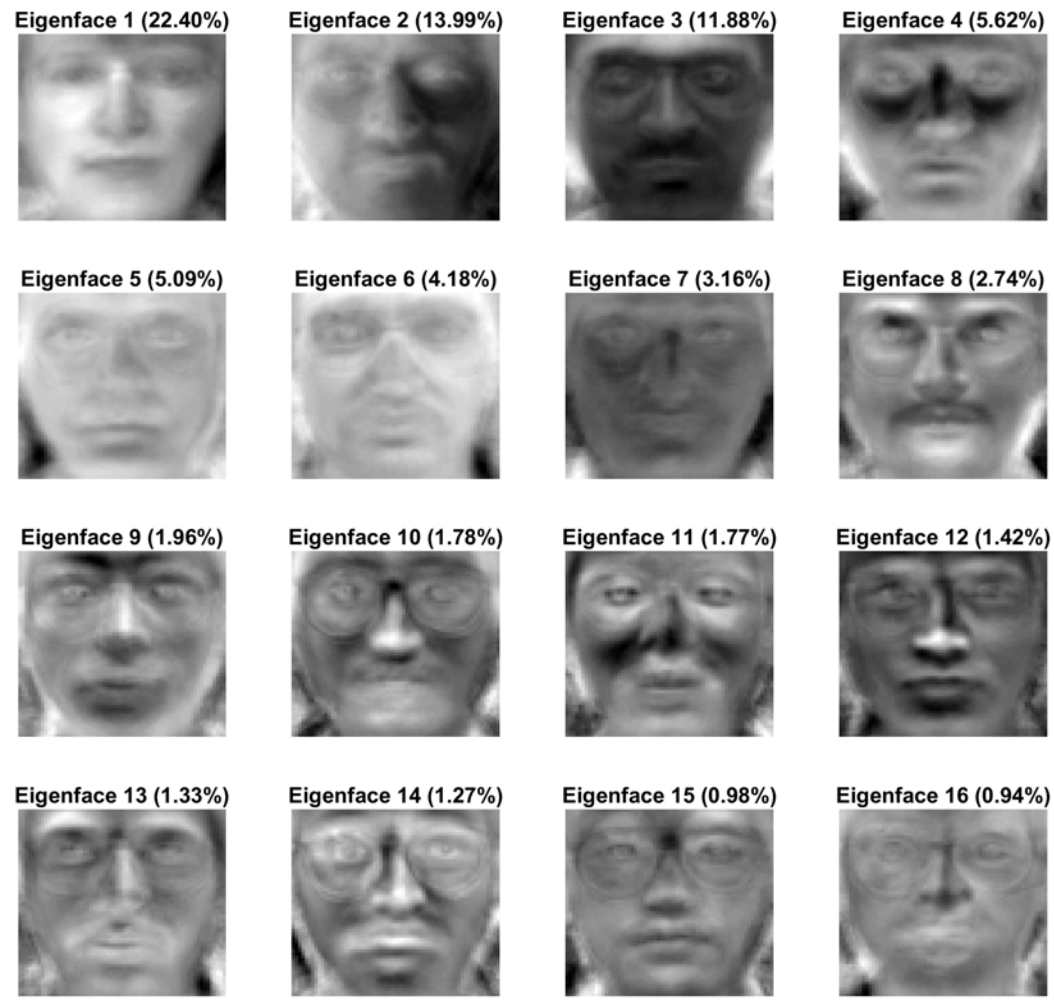
- PCA finds a set of **orthogonal vectors** that best explain the variance of the sample covariance matrix
- These are exactly the **eigenvectors** of the covariance matrix $X^T X$
- We can discard the eigenvectors corresponding to small magnitude eigenvalues to yield an approximation
- **Simple algorithm** to describe, MATLAB and other programming languages have built in **support** for eigenvector/eigenvalue computations

Principal Component Analysis: Training

PCA Training

Given: training data $X \in \mathbb{R}^{n\times d}$

- pre-process and center the training data
- Compute the eigenvalues and eigenvectors of the covariance matrix $[V, \Lambda] = eig(X^T X)$
- Save the top k eigenvectors (columns of V) as $V_k \in \mathbb{R}^{d\times k}$



Principal Component Analysis identifies the principal components in the covariance matrix of the face data

- in face recognition, the eigenvectors are called **eigenfaces**; *as there are 4096 features, there are 4096 eigenfaces*
- in this example, the first $k = 16$ eigenvectors capture 80.5% of the total variance (sum of all the eigenvalues)
  - in practice, we compute the cumulative sum of the eigenvalues and choose k such that we reach a satisfactory approximation threshold (typically, 90% of the variance)

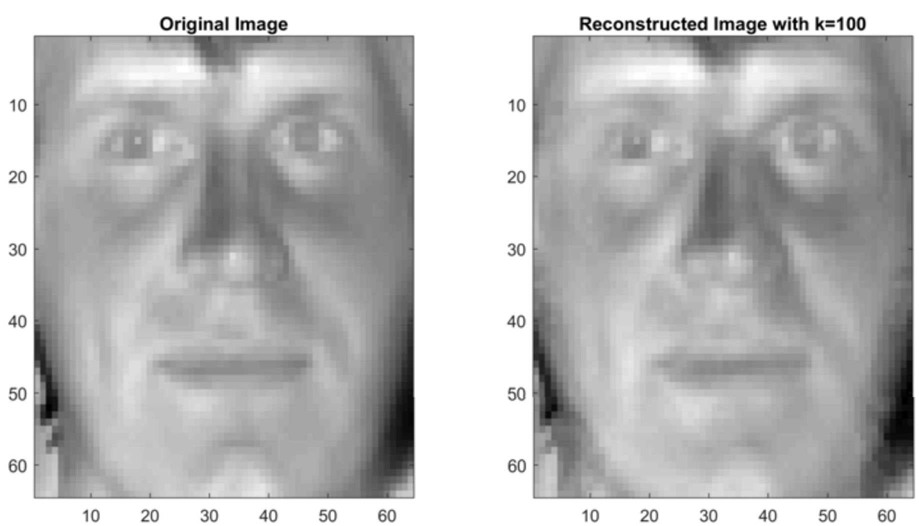
Principal Component Analysis: Prediction

PCA Testing

Given: test example $\boldsymbol{x}_{\text{test}} \in \mathbb{R}^{d\times 1}$

- pre-process and center the test example
- compute the projection of $\boldsymbol{x}_{\text{test}}$ onto each of the k eigen-vectors: $\boldsymbol{c}_{\text{test}} = V_k^T \boldsymbol{x}_{\text{test}}$, where $\boldsymbol{c}_{\text{test}} \in \mathbb{R}^{k\times 1}$
- determine if the input image is close to one of the faces in the data set

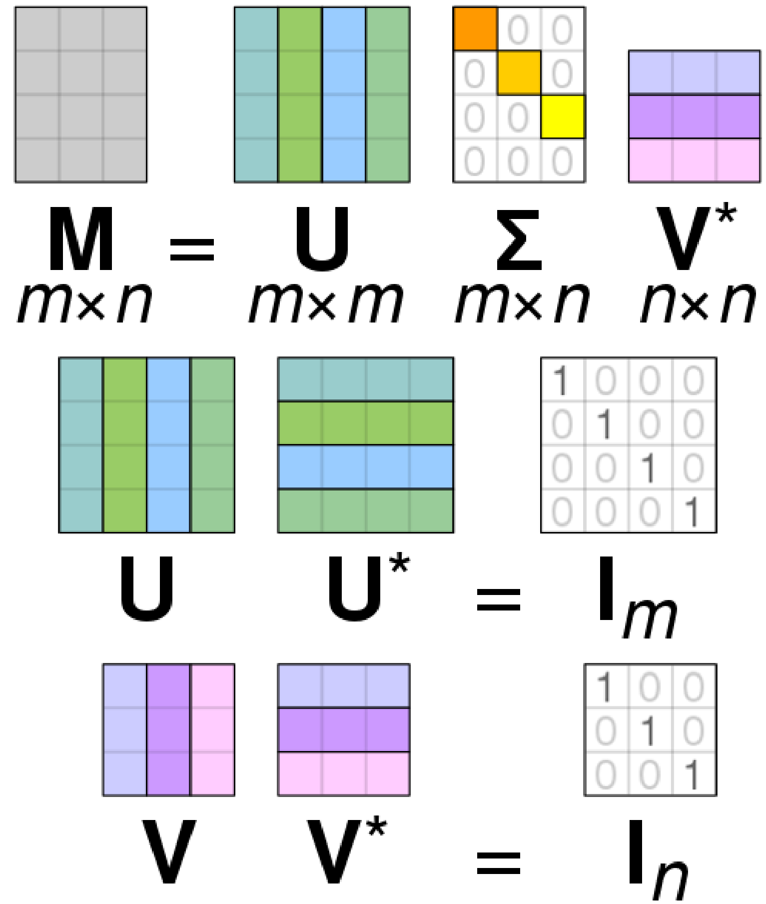
Each new example can now be represented using k dimensions, by ***projecting it onto the top k eigen-basis****. This means that instead of $d = 4096$ features, PCA now allows us to use $k = 16$ features!*



Using more eigenvectors improves the accuracy of reconstruction, but also increases the complexity of representation and decreases the efficiency of computation. Here, the choice of $k = 100$ is still several orders of magnitude smaller than the original dimension, $d = 4096$.

PCA in Practice

Forming the sample covariance matrix $X^T X$ can **require a lot of memory** (especially if $n \gg d$)
- higher resolution images (256 x 256) say, require that we construct a 65536 x 65536 covariance matrix
- Need a faster way to compute this without forming the covariance matrix explicitly
- Typical approach: use the singular value decomposition



Relationship between the **eigenvalue decomposition** and the **singular value decomposition**:
- every matrix $X \in \mathbb{R}^{n\times d}$ admits a **decomposition** of the form $X = U\Sigma V^T$
  - where $U \in \mathbb{R}^{n\times n}$ is an orthonormal matrix, $\Sigma \in \mathbb{R}^{n\times d}$ is a non-negative diagonal matrix, and $V \in \mathbb{R}^{d\times d}$ is an orthonormal matrix
  - the σ_{ii} entries of the diagonal matrix Σ are called the singular values
- $X^T X = V\Sigma^T U^T U \Sigma V^T = V(\Sigma^T\Sigma)V^T$; eigenvalues are squares of singular values; right singular vectors are eigenvectors!

PCA in Practice

While PCA is an **unsupervised** method, it is commonly used as a pre-processing/dimensionality reduction step for supervised classification problems

- PCA **does not take labels into account** to determine a low-dimensional projection subspace
- this means that if two classes both share a direction of maximum variance, projection into PCA space will make them **inseperable**!

Approaches such as **Linear Discriminant Analysis** handle this drawback by using other criteria to identify a low-dimensional subspace