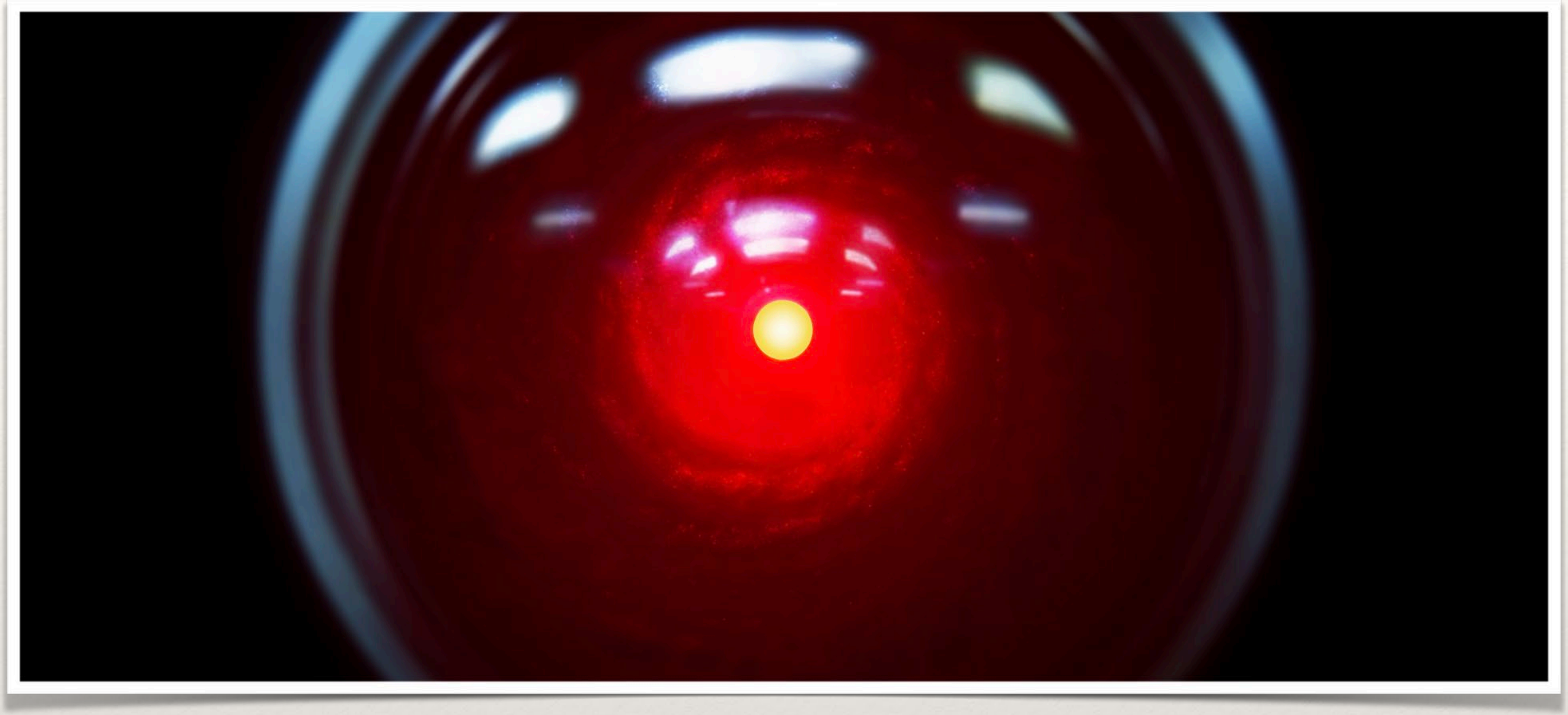


Computer Vision

Covariance and Principal Components

Hansung Kim
h.kim@soton.ac.uk

Example

Face dataset

Variance and Covariance

Random Variables and Expected Values

- Mathematicians talk variance (and covariance) in terms of *random variables* and *expected values*
- The expected value (denoted E[X]) is the most likely value a random variable will take.
  - For this course we'll **assume** that the values an element of a feature can take are all equally likely
    - The expected value is thus just the **mean value**

Variance

$$\sigma^2(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

Variance (σ^2) is the mean squared difference from the mean (μ).
It's a measure of how spread-out the data is.

technically it's $E[(X - E[X])^2]$

Covariance

$$\sigma(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x - \mu_x)(y - \mu_y)$$

Covariance ($\sigma(x,y)$) measures how two variables change together

technically it's E[(x - E[x])(y-E[y])]

Covariance

$$\sigma(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x - \mu_x)(y - \mu_y)$$

The variance is the covariance when the two variables are the same ($\sigma(x,x)=\sigma^2(x)$)

Covariance

$$\sigma(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x-\mu_x)(y-\mu_y)$$

A covariance of 0 means the variables are uncorrelated (independent).

Covariance is related to Correlation

$$\boxed{\rho(x,y) = \frac{\sigma(x,y)}{\sqrt{\sigma^2(x)}\sqrt{\sigma^2(y)}}}$$

Covariance Matrix

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma(X_1, X_1) & \sigma(X_1, X_2) & \cdots & \sigma(X_1, X_n) \\ \sigma(X_2, X_1) & \sigma(X_2, X_2) & \cdots & \sigma(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(X_n, X_1) & \sigma(X_n, X_2) & \cdots & \sigma(X_n, X_n) \end{bmatrix}$$

A covariance matrix encodes how all possible pairs of dimensions in an *n*-dimensional dataset vary together

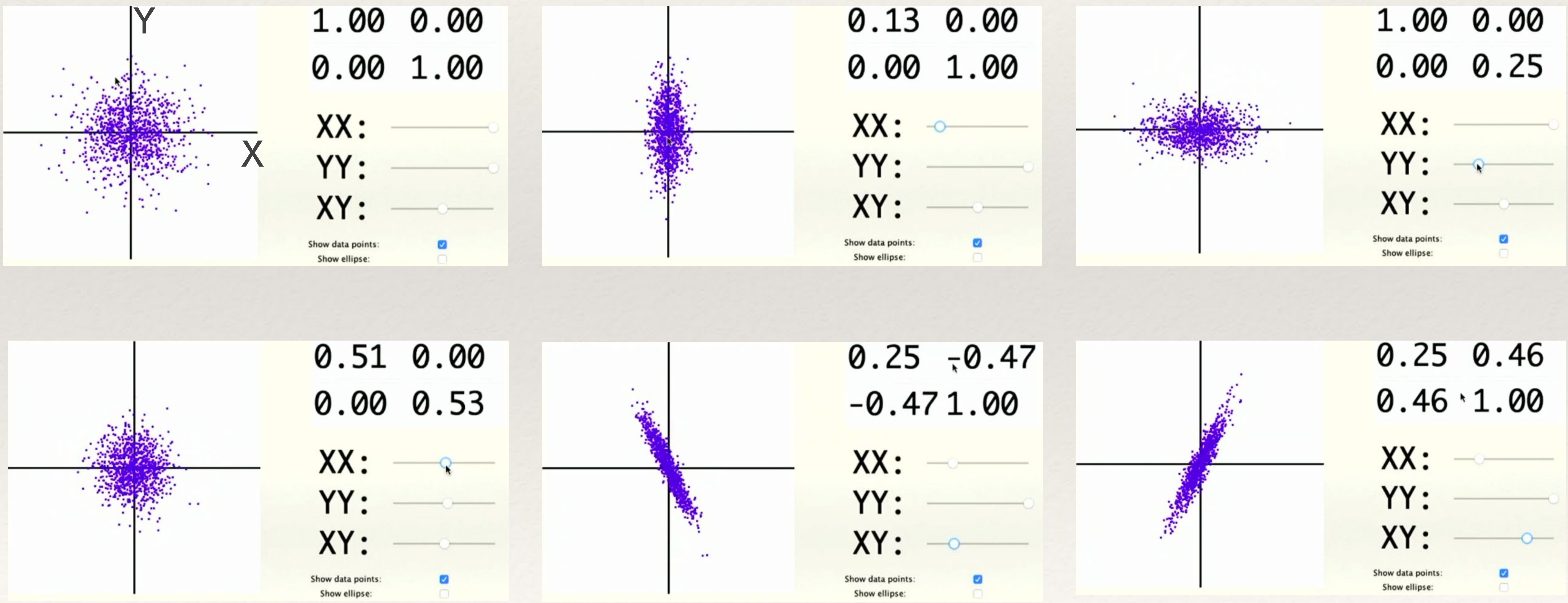
Covariance Matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma(X_1, X_1) & \sigma(X_1, X_2) & \cdots & \sigma(X_1, X_n) \\ \sigma(X_2, X_1) & \sigma(X_2, X_2) & \cdots & \sigma(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(X_n, X_1) & \sigma(X_n, X_2) & \cdots & \sigma(X_n, X_n) \end{bmatrix}$$

The covariance matrix is a **square symmetric matrix**

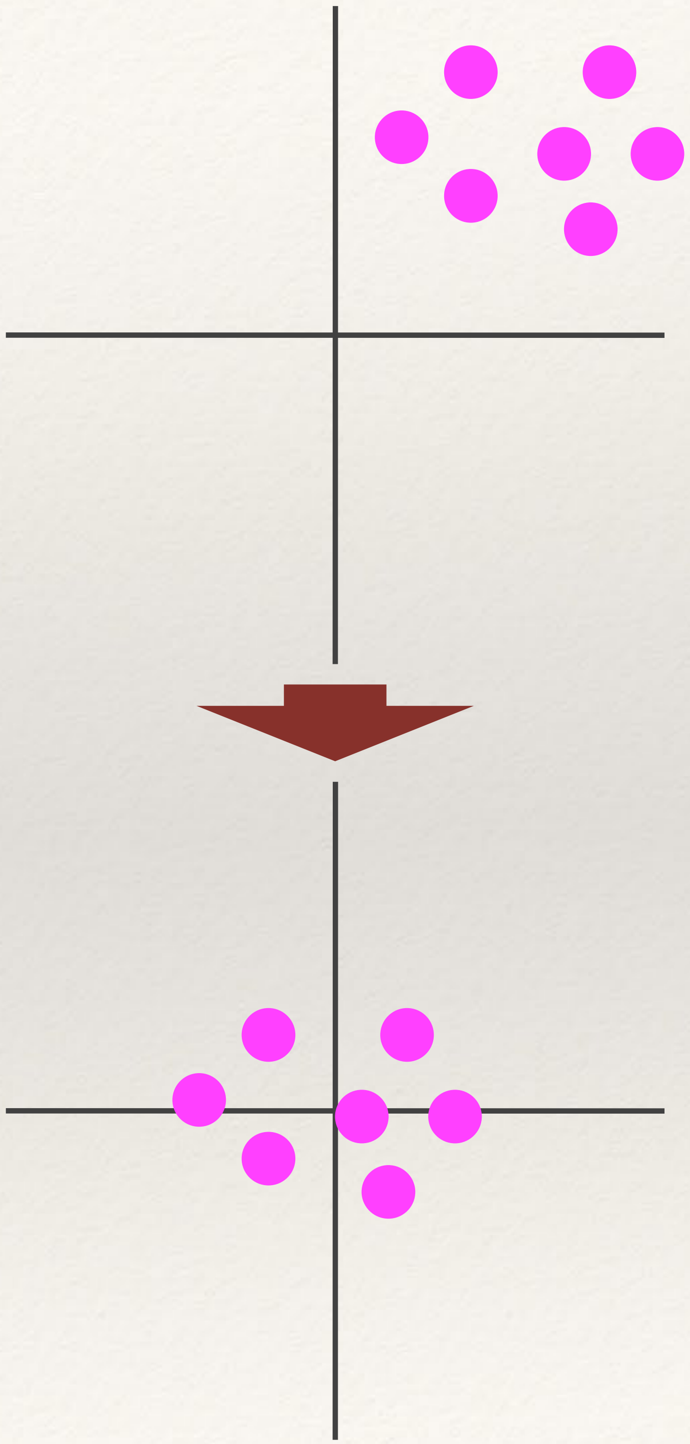
2D Covariance

Geometric interpretation of $\Sigma = \begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{bmatrix}$



Mean Centring

- Mean Centring is the process of computing the mean (across each dimension independently) of a set of vectors, and then subtracting the mean vector from every vector in the set.
  - All the vectors will be translated so their average position is the origin



Covariance matrix again

$$
Z = \begin{bmatrix} v_{11} & v_{12} & v_{13} & \dots \\ v_{21} & v_{22} & v_{23} & \dots \\ v_{31} & v_{32} & v_{33} & \dots \\ v_{41} & v_{42} & v_{43} & \dots \end{bmatrix}
$$

Each row is a **mean centred** featurevector

To the **mean of each column**

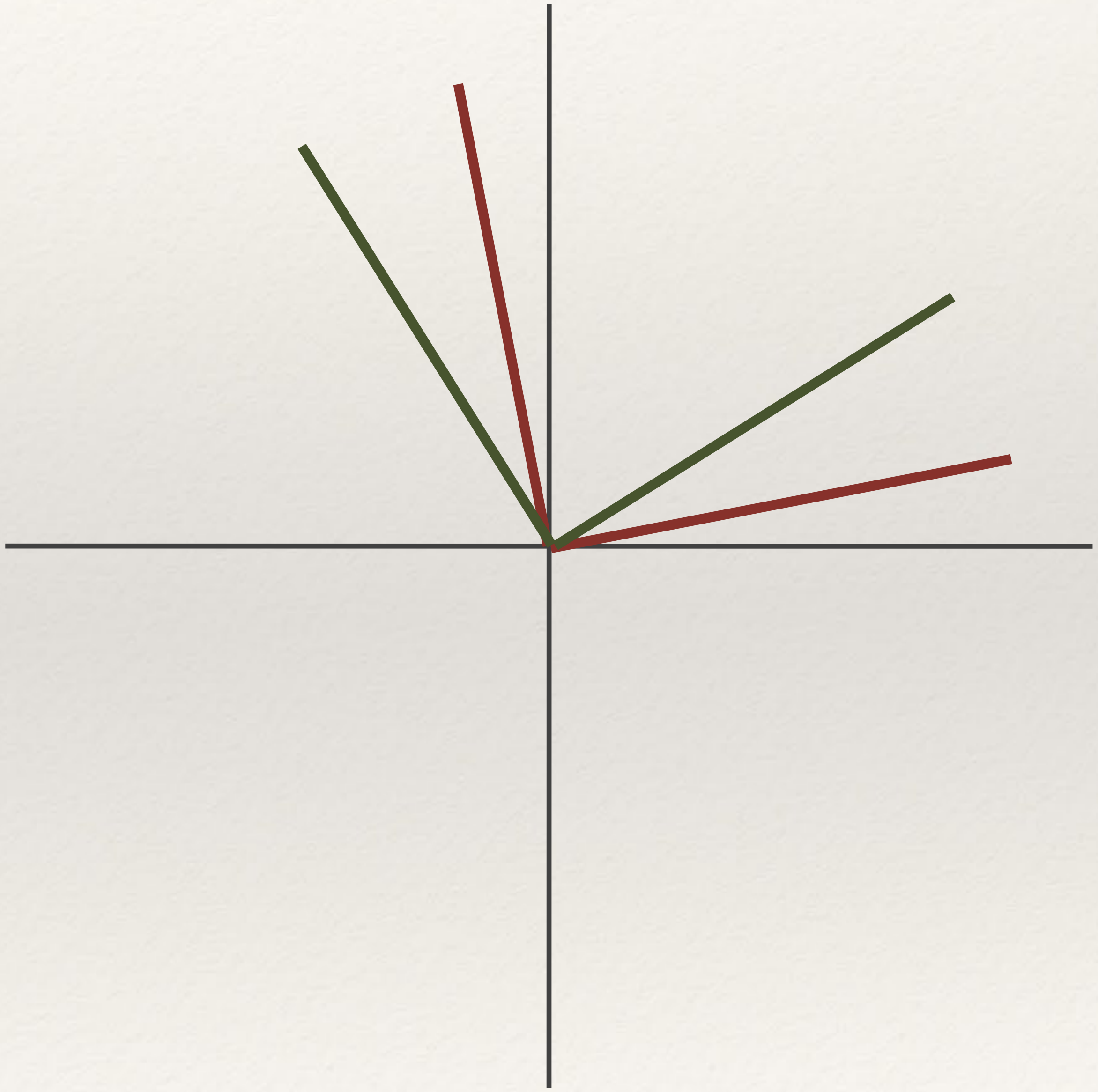
Then

$$\Sigma \propto Z^T Z$$

Principal axes of variation

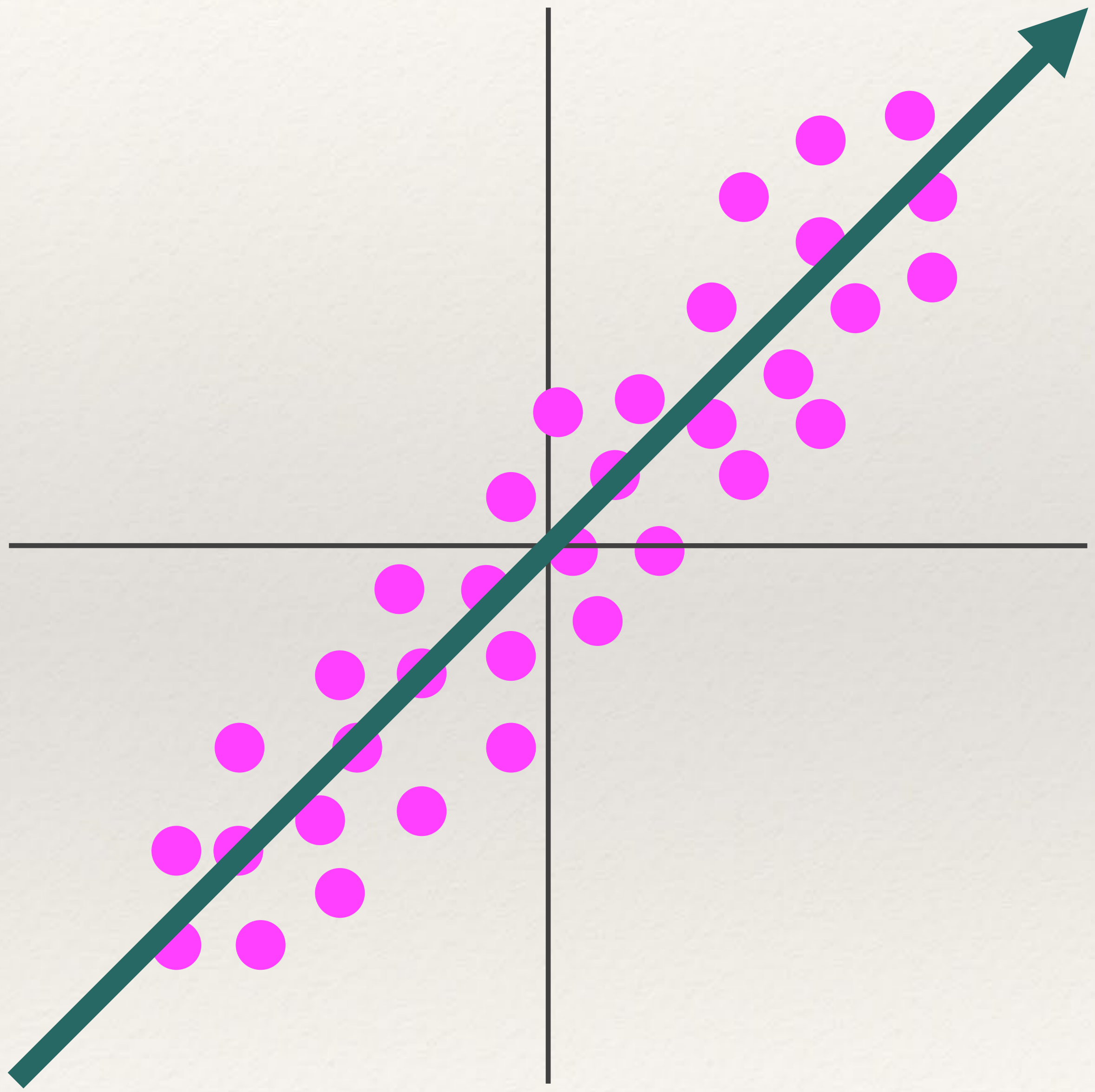
Basis

- A basis is a set of n **linearly independent** vectors in an n dimensional space
  - The vectors are **orthogonal**
  - They form a “coordinate system”
  - There are an infinite number of possible basis



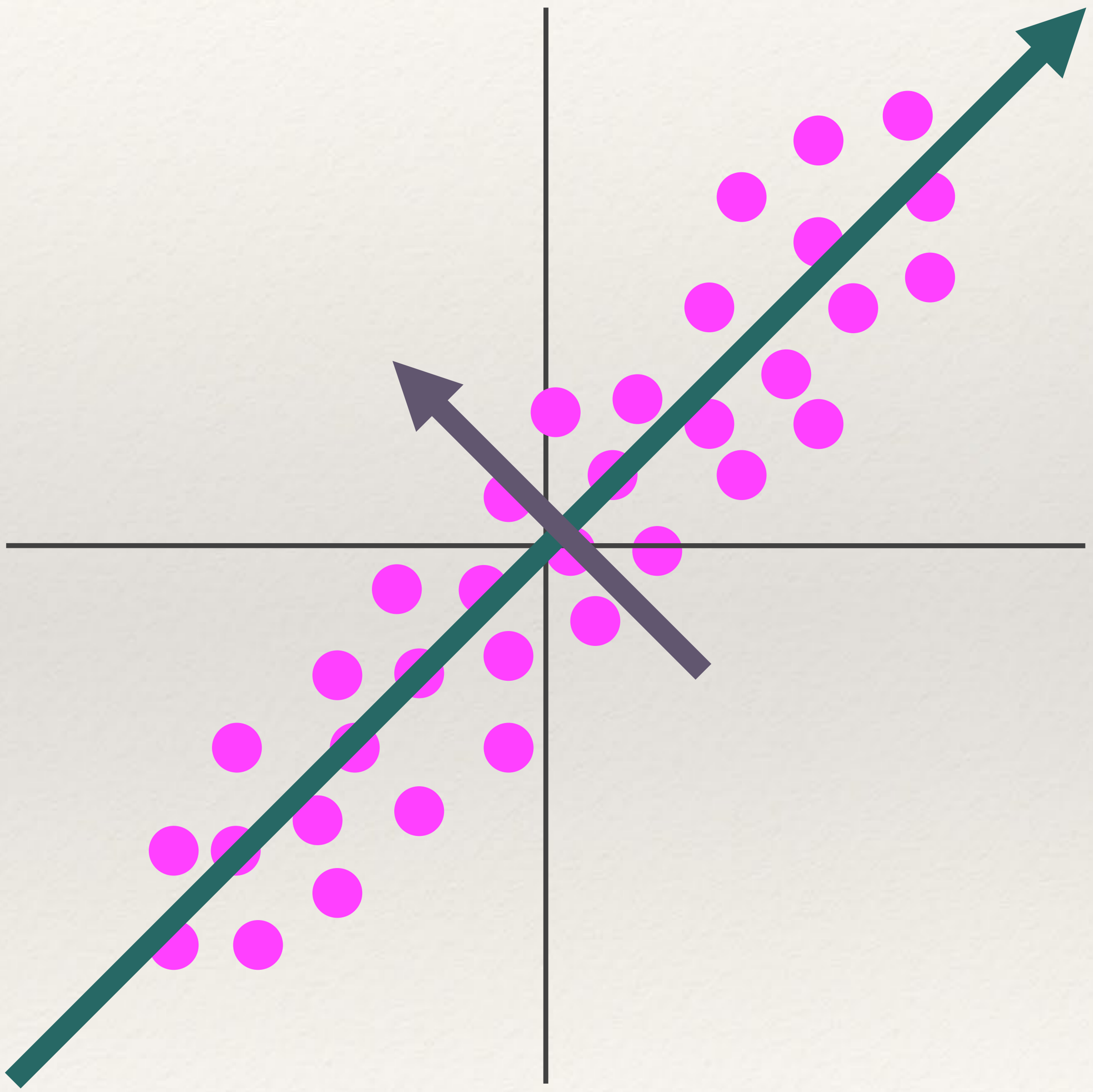
The first principal axis

- For a given set of n dimensional data, the ***first principle axis*** (or just ***principal axis***) is the vector that describes the direction of **greatest** variance.



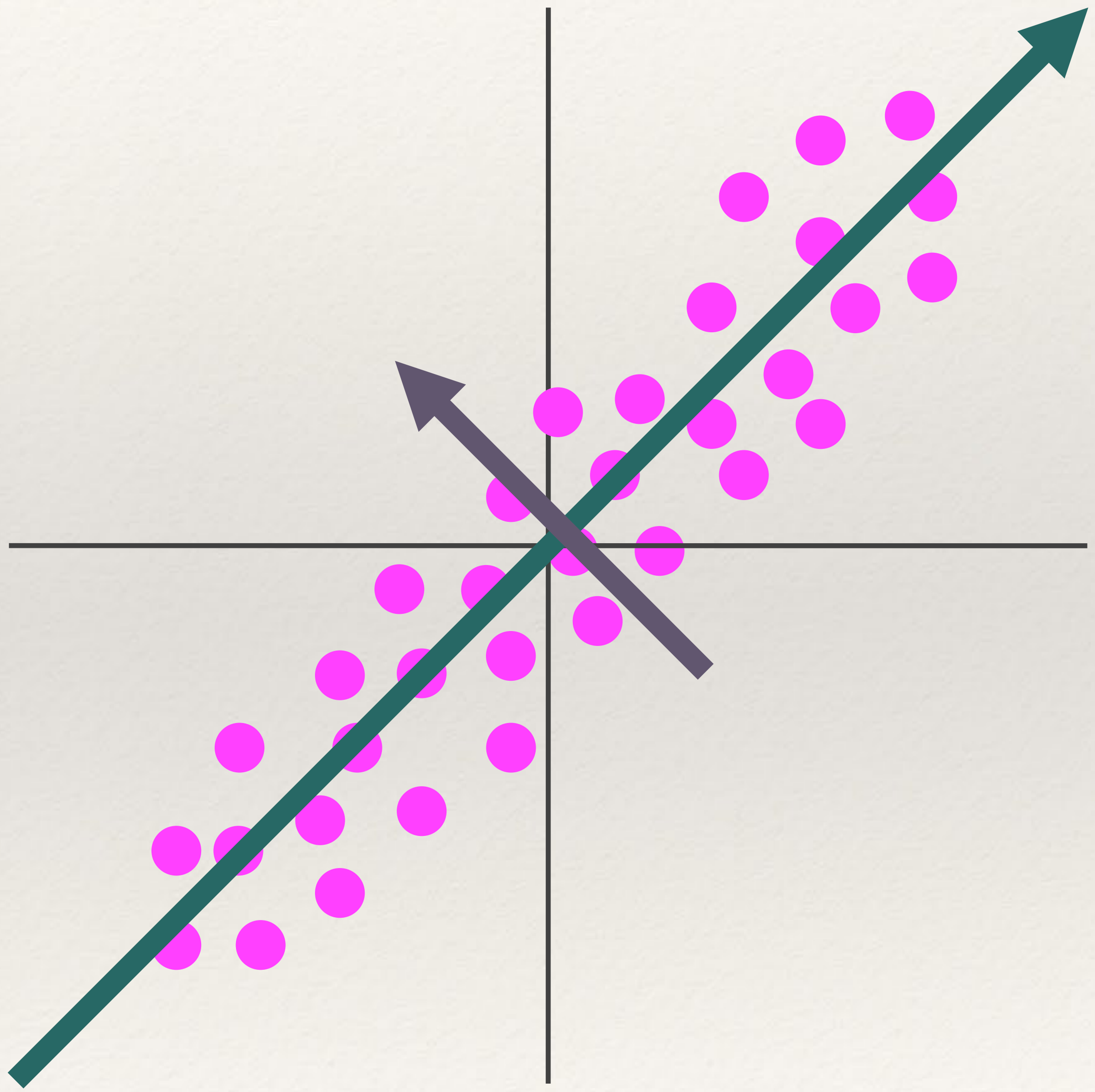
The second principal axis

- The **second principal axis** is a vector orthogonal (perpendicular) to the first major axis.



Three principal axes in 3D

- In a space with 3 or more dimensions, the second principal axis is the direction of the second greatest variance orthogonal to the first principal axes.
  - The set of *n* **principal axes** of an *n* dimensional space are a **basis**



Eigenvectors, Eigenvalues and Eigendecomposition

Eigenvectors and Eigenvalues

Very important equation!

$$\mathbf{A}v = \lambda v$$

Eigenvectors and Eigenvalues

a $n \times n$ square matrix

a scalar value,
known as an
eigenvalue

$$\boxed{\mathbf{A}}\boxed{v}=\boxed{\lambda}\boxed{v}$$

an n dimensional vector,
known as an **eigenvector**

Eigenvectors and Eigenvalues

$$\mathbf{A}v = \lambda v$$

There are at most *n* eigenvector-eigenvalue pairs

If **A** is **symmetric**, then the set of eigenvectors is **orthogonal**

Can you see where this is going?

Eigenvectors and Eigenvalues

$$\mathbf{A}v = \lambda v$$

- If A is a **covariance matrix**, then the eigenvectors are the **principal axes**.
- The eigenvalues are proportional to the **variance** of the data along each eigenvector.
- The eigenvector corresponding to the **largest eigenvalue** is the first P.C.

Finding the EVecs and EVals

- For small matrices ($n \leq 4$) there are algebraic solutions to finding all the eigenvector-eigenvalue pairs
- For larger matrices, numerical solutions to the **Eigendecomposition** must be sought.

Eigendecomposition

columns of **Q** are the eigenvectors

$$\mathbf{A} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1}$$

diagonal eigenvalue matrix ($\boldsymbol{\Lambda}_{ii} = \lambda_i$)

Eigendecomposition

$$\mathbf{A} = \mathbf{Q}\mathit{\Lambda}\mathbf{Q}^{-1}$$

If **A** is *real symmetric* (i.e. a covariance matrix), then $\mathbf{Q}^{-1} = \mathbf{Q}^{T}$ (i.e. eigenvectors are orthogonal), so:

$$\mathbf{A} = \mathbf{Q}\mathit{\Lambda}\mathbf{Q}^{T}$$

Eigendecomposition

In summary, the Eigendecomposition of a covariance matrix **A**:

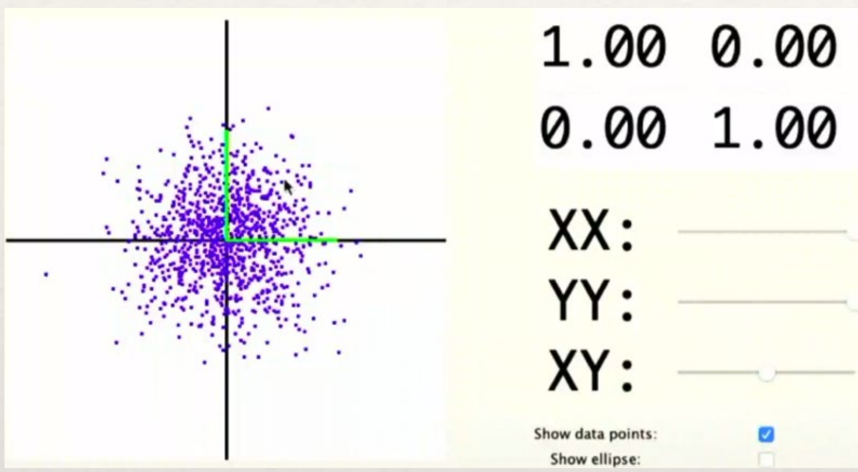
$$\mathbf{A} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^{T}$$

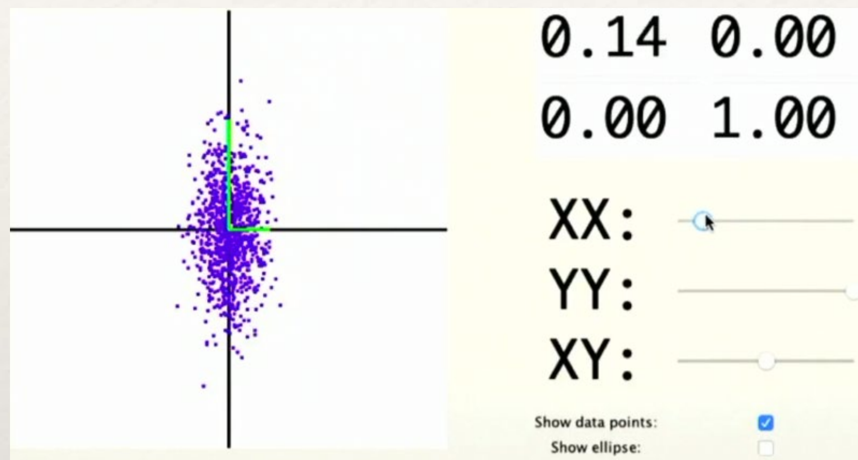
Gives you the principal axes and their relative magnitudes

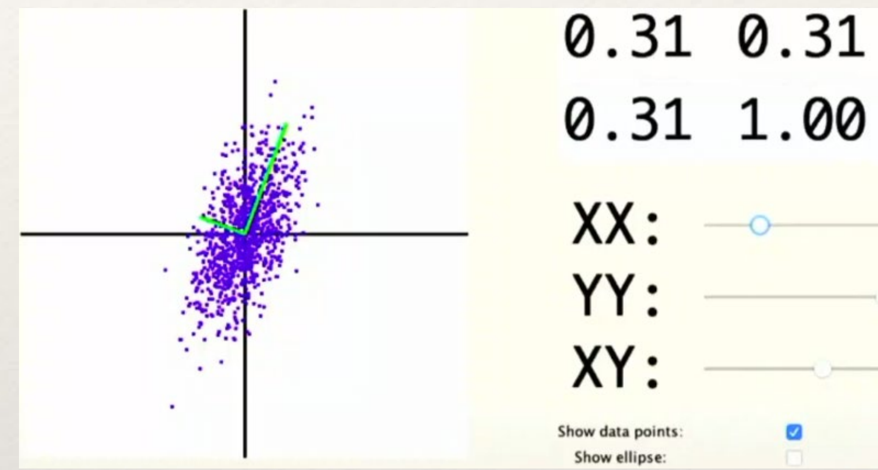
Ordering

- Standard Eigendecomposition implementations will order the eigenvectors (columns of $\mathbf{Q}$) such that the eigenvalues (in the diagonal of $\mathbf{\Lambda}$) are sorted in order of decreasing value.
    - Some solvers are optimised to only find the top k eigenvalues and corresponding eigenvectors, rather than all of them.

Covariance, Eigendecomposition and principal axes







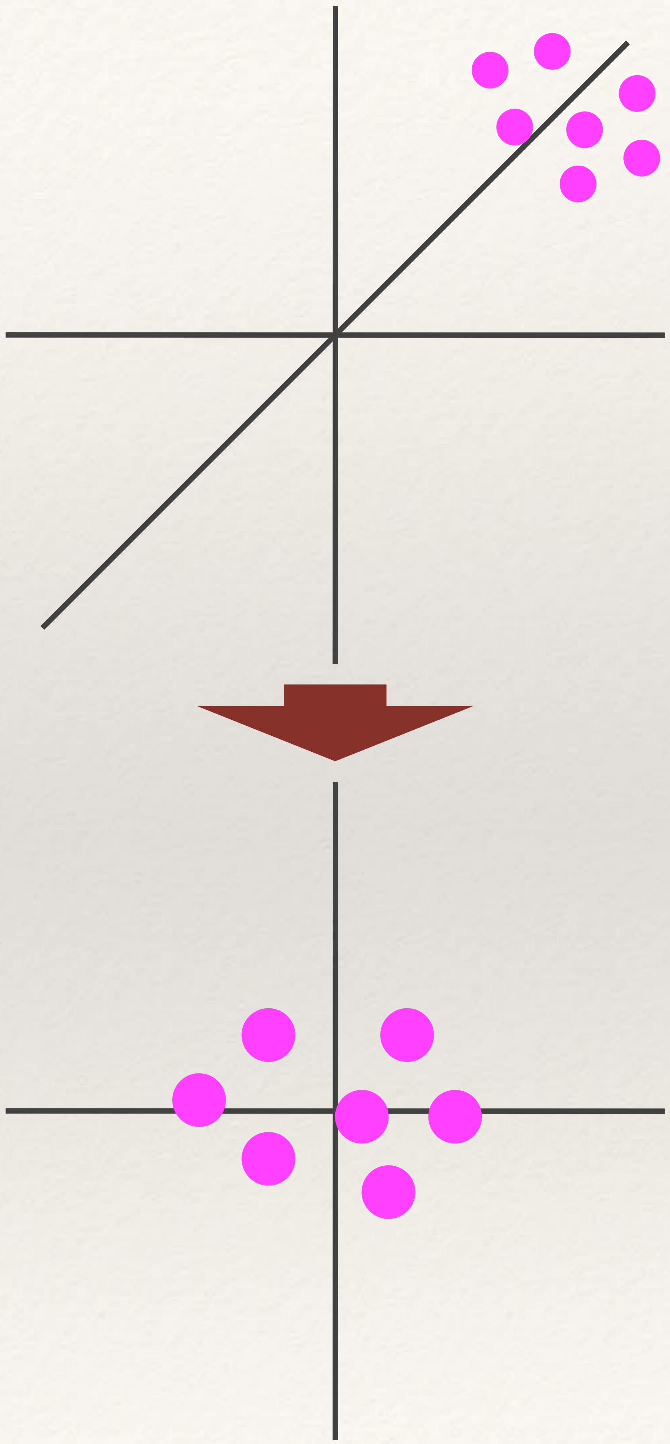
$$Q = \begin{matrix} 0.00 & 1.00 \\ 1.00 & 0.00 \end{matrix} \qquad Q = \begin{matrix} 0.00 & 1.00 \\ 1.00 & 0.00 \end{matrix} \qquad Q = \begin{matrix} 0.36 & -0.93 \\ 0.93 & 0.36 \end{matrix}$$

$$\Lambda = \begin{matrix} 1.00 & 0.00 \\ 0.00 & 1.00 \end{matrix} \qquad \Lambda = \begin{matrix} 1.00 & 0.00 \\ 0.00 & 0.14 \end{matrix} \qquad \Lambda = \begin{matrix} 1.12 & 0.00 \\ 0.00 & 0.19 \end{matrix}$$

Principal Component Analysis (PCA)

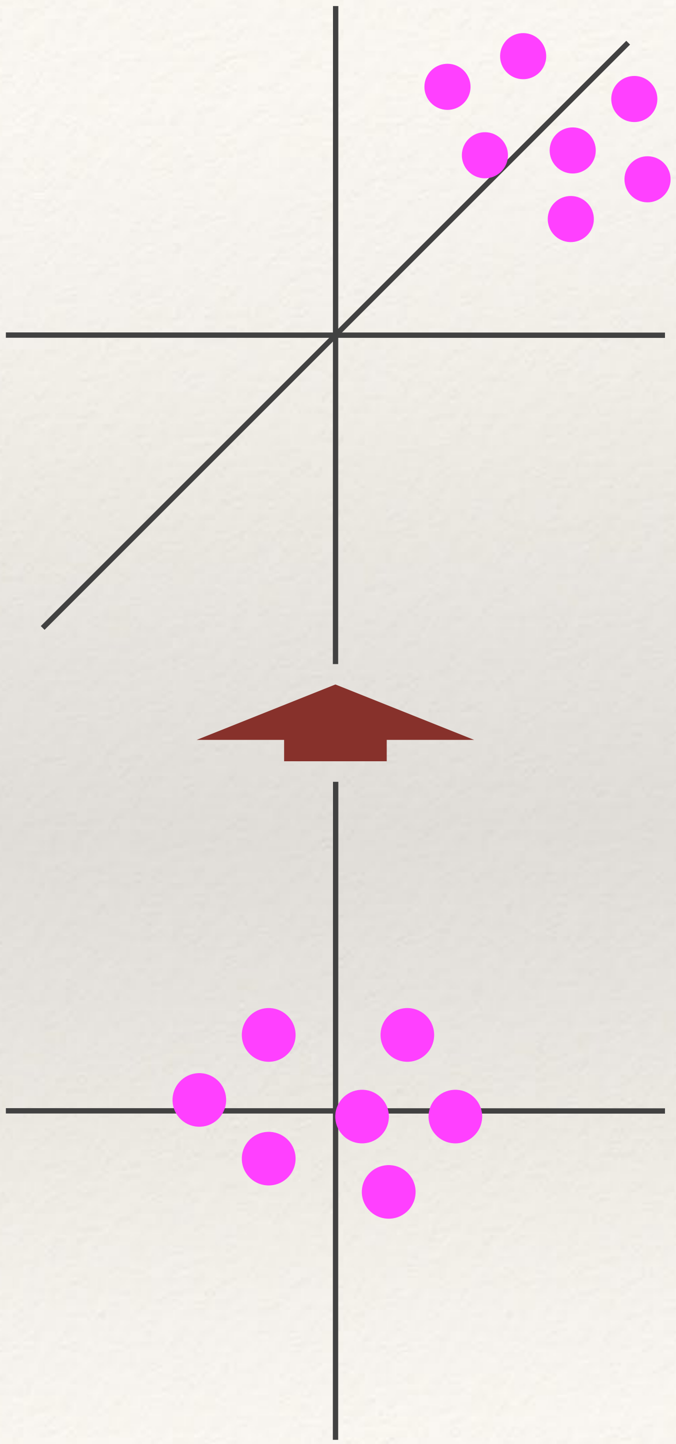
Linear Transform

- A linear transform **W** projects data from one space into another:
  - **T** = **ZW**
    - Original data stored in the rows of **Z**
  - **T** can have fewer dimensions than **Z**.



Linear Transform

- The effects of a linear transform can be reversed if **W** is **invertible**:
  - **Z** = **TW**$^{-1}$
    - A lossy process if the dimensionality of the spaces is different



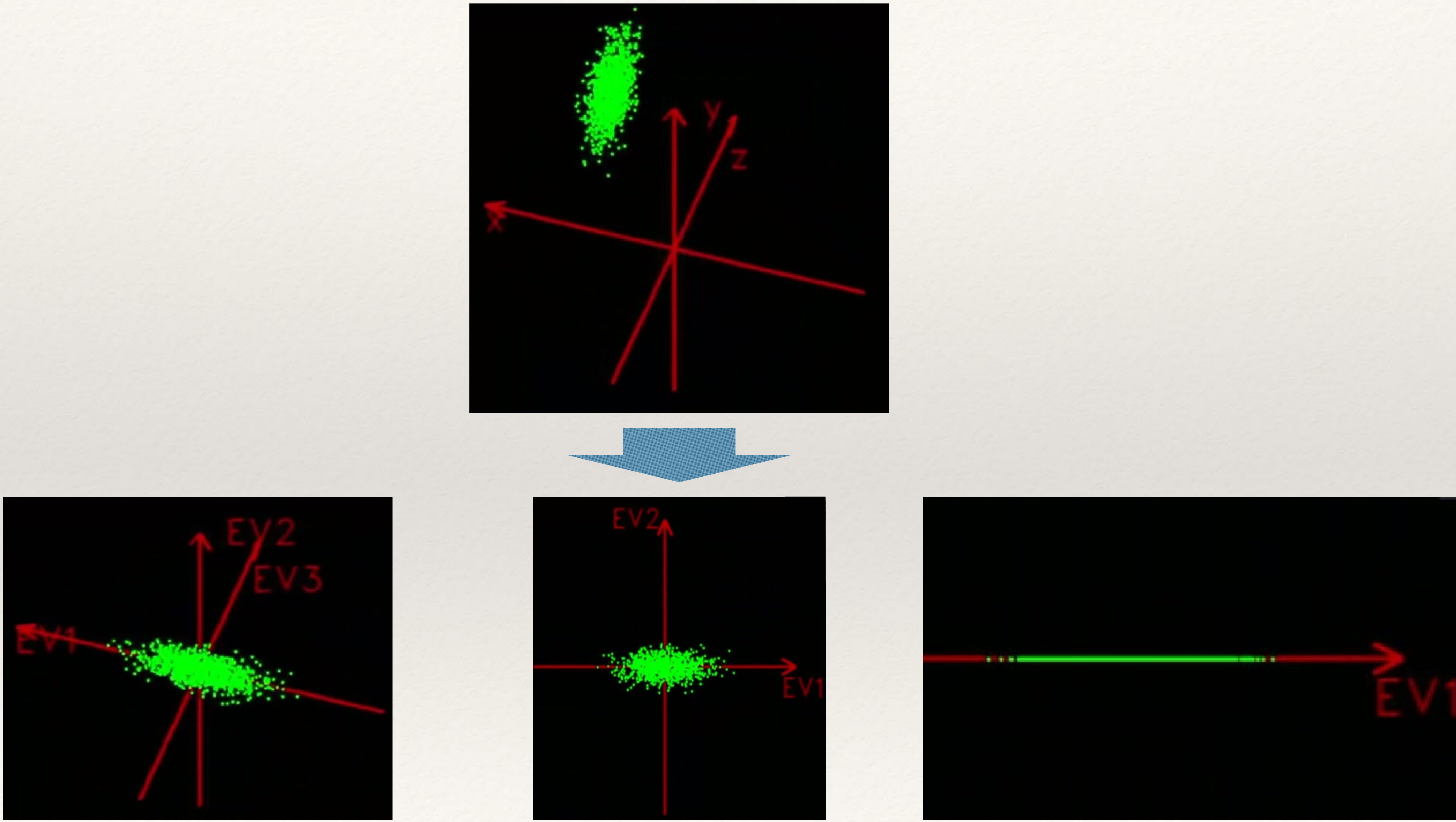
PCA

- PCA is an **Orthogonal Linear Transform** that maps data from its original space to a space defined by the principal axes of the data.
  - The transform matrix **W** is just the eigenvector matrix **Q** from the Eigendecomposition of the covariance matrix of the data.
  - Dimensionality reduction can be achieved by removing the eigenvectors with low eigenvalues from **Q** (i.e. keeping the first *L* columns of **Q** assuming the eigenvectors are sorted by decreasing eigenvalue).

PCA Algorithm

1. Mean-centre the data vectors
2. Form the vectors into a matrix $\mathbf{Z}$, such that each row corresponds to a vector
3. Perform the Eigendecomposition of the matrix $\mathbf{Z}^T\mathbf{Z}$, to recover the eigenvector matrix $\mathbf{Q}$ and diagonal eigenvalue matrix $\mathbf{\Lambda}$: $\mathbf{Z}^T\mathbf{Z} = \mathbf{Q\Lambda Q}^T$
4. Sort the columns of $\mathbf{Q}$ and corresponding diagonal values of $\mathbf{\Lambda}$ so that the eigenvalues are decreasing.
5. Select the L largest eigenvectors of $\mathbf{Q}$ (the first L columns) to create the transform matrix Q_L.
6. Project the original vectors into a lower dimensional space, $\mathbf{T}_L$: $\mathbf{T}_L = \mathbf{ZQ}_L$

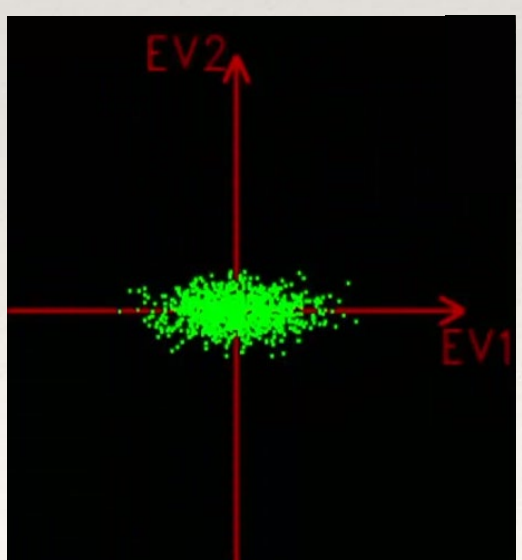
PCA



3D 2D 1D

PCA

3D

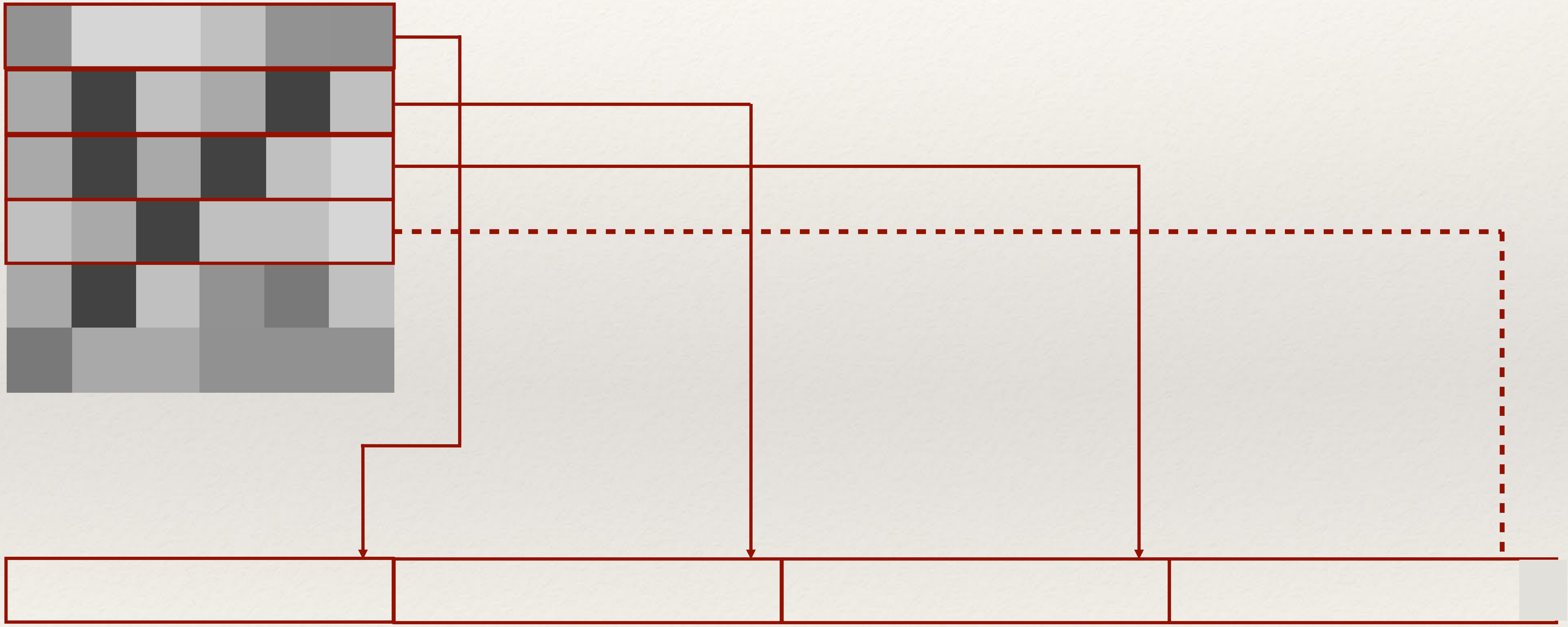


2D

1D

Eigenfaces
(Eigenimages)

A simple kind of feature…



… but with some problems …

- Not invariant to:
  - Change in the position/orientation/scale of the object in the image
  - Changes in lighting
  - Size of the image
- Highly susceptible to image noise

Making it invariant

- Require (almost) the same object pose across images (i.e. full frontal faces)
- Align (rotate, scale and translate) the images so that the a common feature is in the same place (i.e. the eyes in a set of face images)
- Make all the aligned images the same size
- (optional) Normalise (or perhaps histogram equalise) the images so they are invariant to global intensity changes

…but there is still a bit of a problem…

- The feature vectors are huge!
  - If the images are 100x200 pixels, the vector has 20000 dimensions
  - That's not really practical…
  - Also, the vectors are still highly susceptible to imaging noise and variations due to slight mis-alignments

Potential solution… Apply PCA

- PCA can be used to reduce the dimensionality
  - smaller number of dimensions allows greater robustness to noise and mis-alignment
    - there are fewer degrees of freedom, so noise/mis-alignment has much less effect
    - and the dominant features are captured
  - Fewer dimensions makes applying machine-learning much more tractable.

EigenFaces and reconstruction

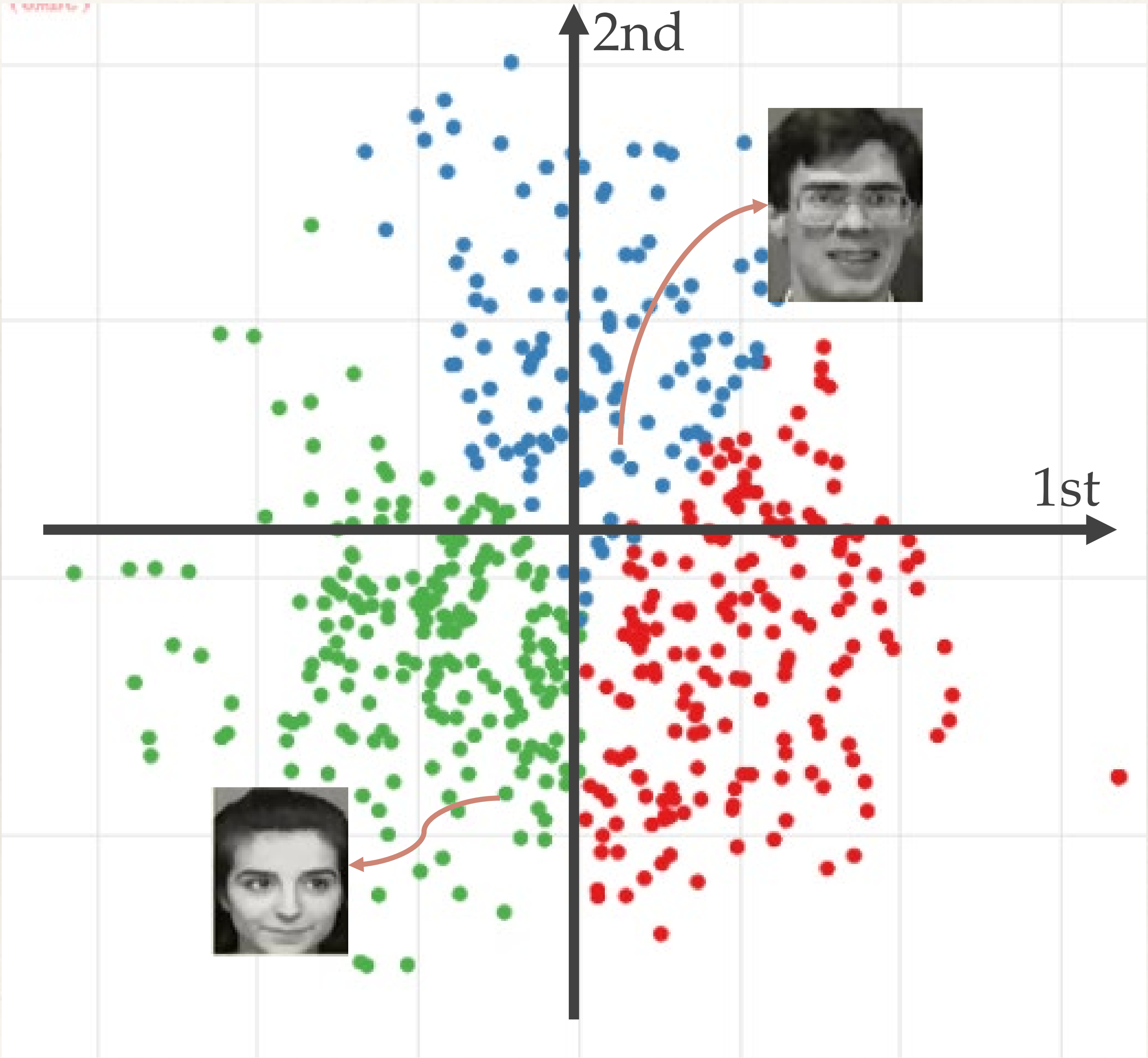
Face dataset

Mean-face

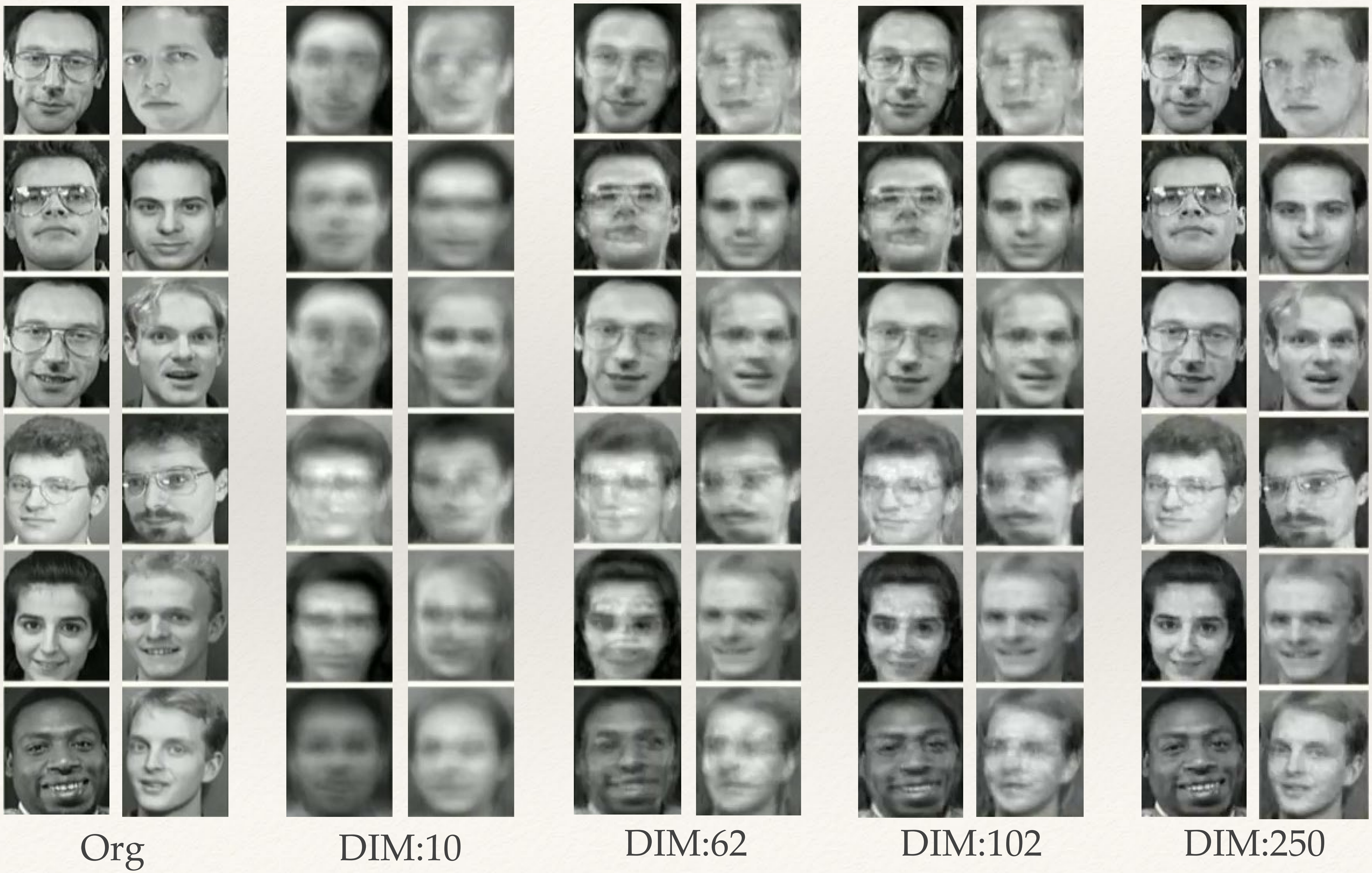
Mean-centred faces

Example for two components

- Visualisation of the face datapoints projected on the first two principal components



Reconstructed faces



Summary

- Covariance measures the "shape" of data by measuring how different dimensions change together.
- The principle axes are a basis, aligned such they describe most directions of greatest variance.
- The Eigendecomposition of the covariance matrix produces pairs of eigenvectors (corresponding to the principal axes) and eigenvalues (proportional to the variance along the respective axis).
- PCA aligns data with its principal axes, and allows dimensionally reduction by discounting axes with low variance.
- Eigenfaces applies PCA to vectors made from pixel values to make robust low-dimensional image descriptors.

Further reading and exercises

- **Further reading**
  - Mark's book covers PCA in the appendices.
  - Wikipedia has good coverage of all the key ideas:
    - http://en.wikipedia.org/wiki/Variance
    - http://en.wikipedia.org/wiki/Covariance
    - http://en.wikipedia.org/wiki/Covariance_matrix
    - http://en.wikipedia.org/wiki/Eigenvalue,_eigenvector_and_eigenspace
    - http://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix
    - http://en.wikipedia.org/wiki/Eigenface
- **Practical exercises – EigenFaces and PCA**
  - OpenIMAJ tutorial chapter 13
  - https://sandipanweb.wordpress.com/2018/01/06/eigenfaces-and-a-simple-face-detector-with-pca-svd-in-python/
  - https://learnopencv.com/eigenface-using-opencv-c-python/

* *Acknowledgements:* Based on earlier Computer Vision lecture slides by Dr. Jon Hare.