

COMP3204/COMP6223: Computer Vision

Covariance and Principal Components

Jonathon Hare
jsh2@ecs.soton.ac.uk

Variance and Covariance

Random Variables and Expected Values

- ❖ Mathematicians talk variance (and covariance) in terms of *random variables* and *expected values*
- ❖ A random variable is a variable that takes on different values due to chance.
 - ❖ The set of sample values from a single dimension of a featurespace can be considered to be a random variable
- ❖ 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

(Covariance is related to Correlation... see notes)



Covariance Matrix

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

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



Covariance Matrix

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

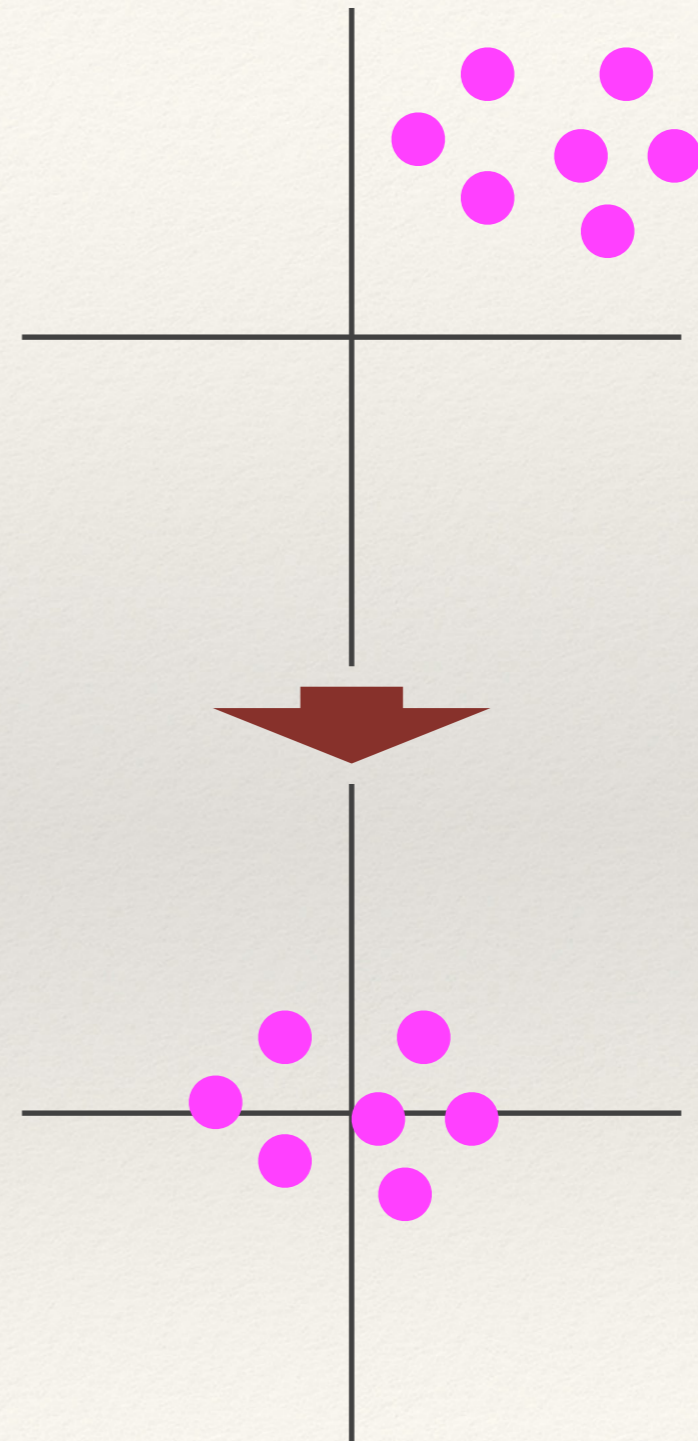
The covariance matrix is a **square symmetric matrix**



Demo: 2D covariance

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

$$\mathbf{Z} = \begin{bmatrix} V_{11} & V_{12} & V_{12} & \dots \\ V_{21} & V_{22} & V_{22} & \dots \\ V_{31} & V_{32} & V_{32} & \dots \\ V_{41} & V_{42} & V_{42} & \dots \end{bmatrix}$$

Each row is a **mean centred**
feature vector

Then

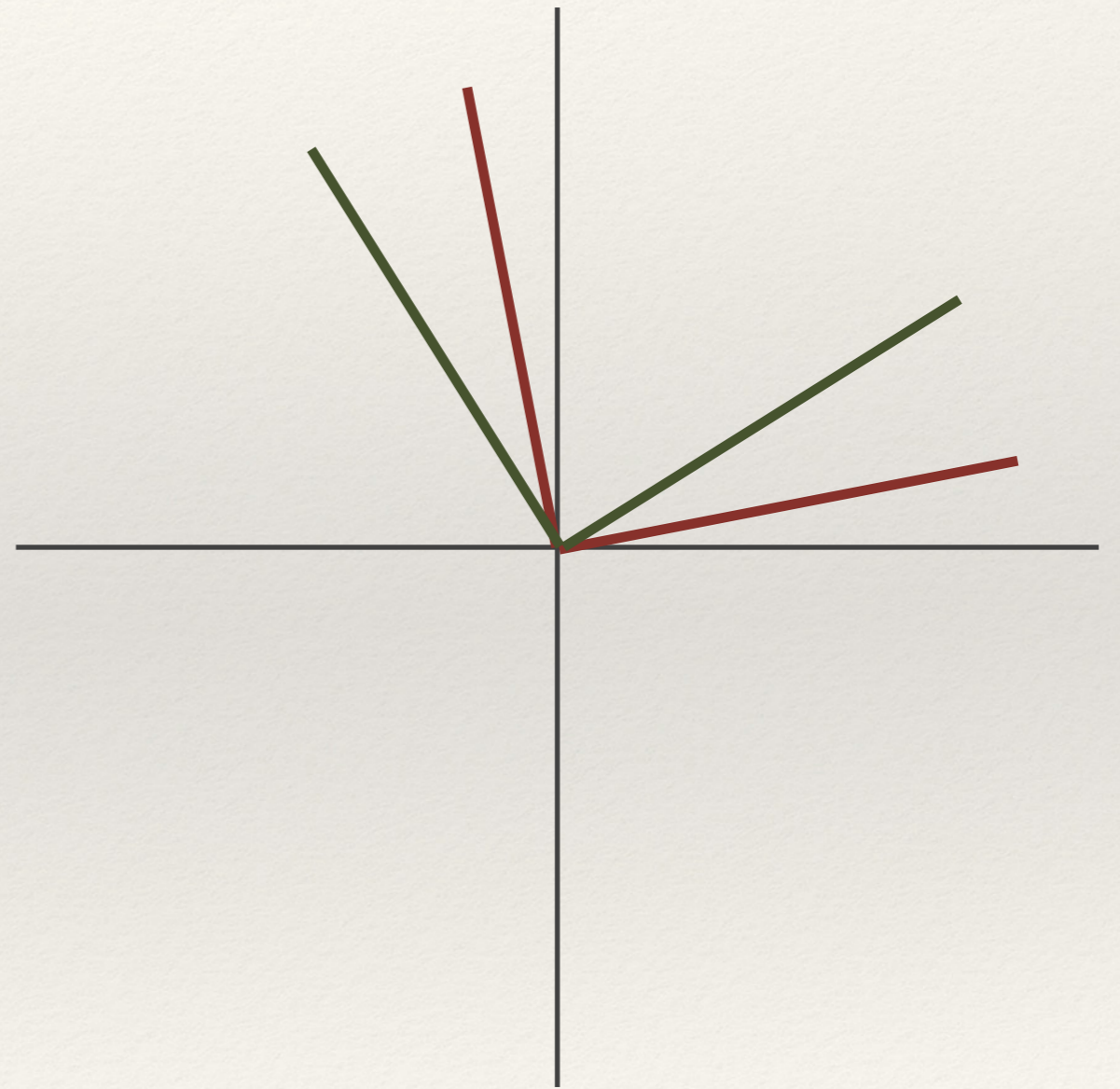
$$\Sigma \propto \mathbf{Z}^T \mathbf{Z}$$



Principal axes of variation

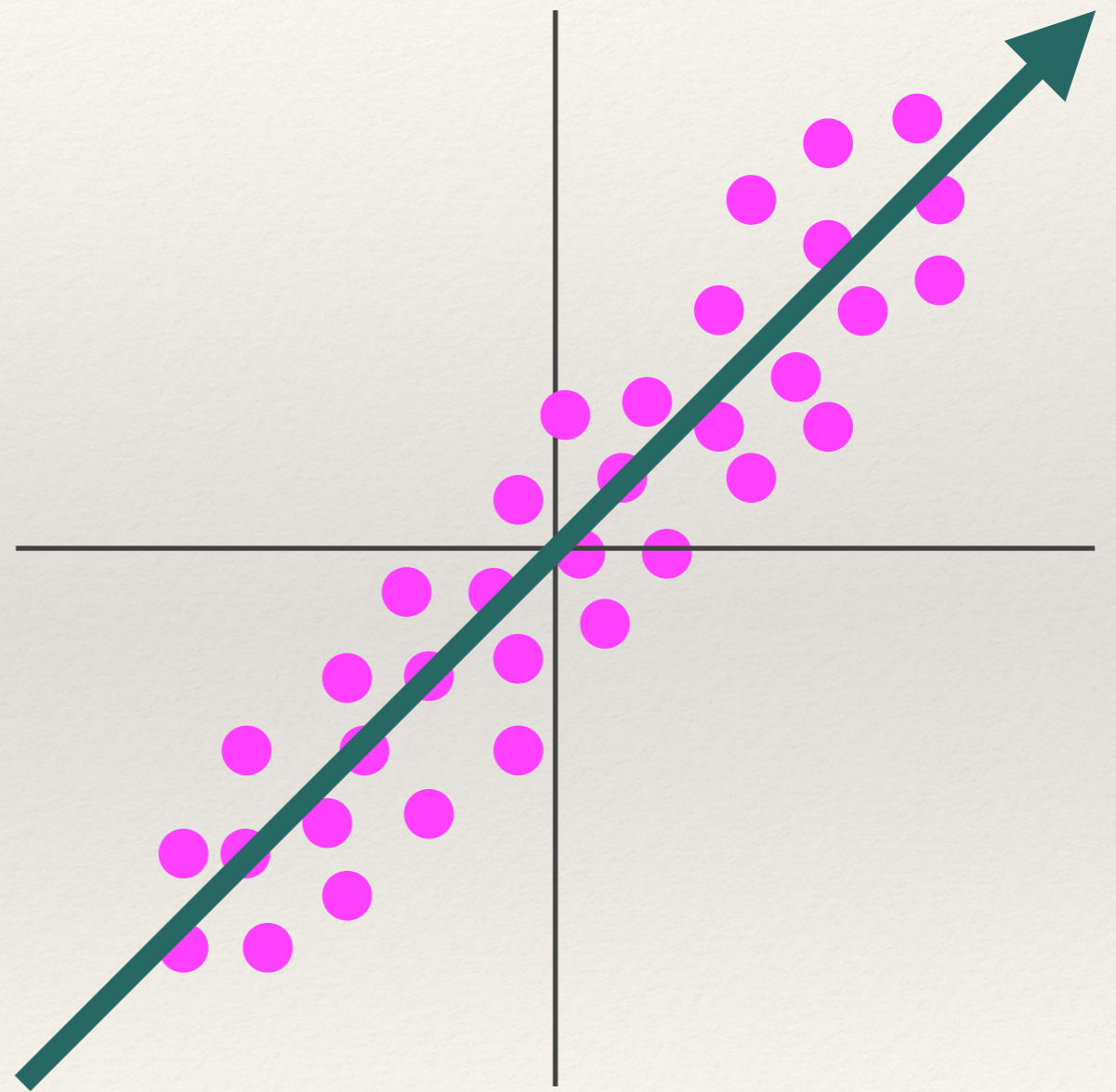
Basis

- ❖ 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



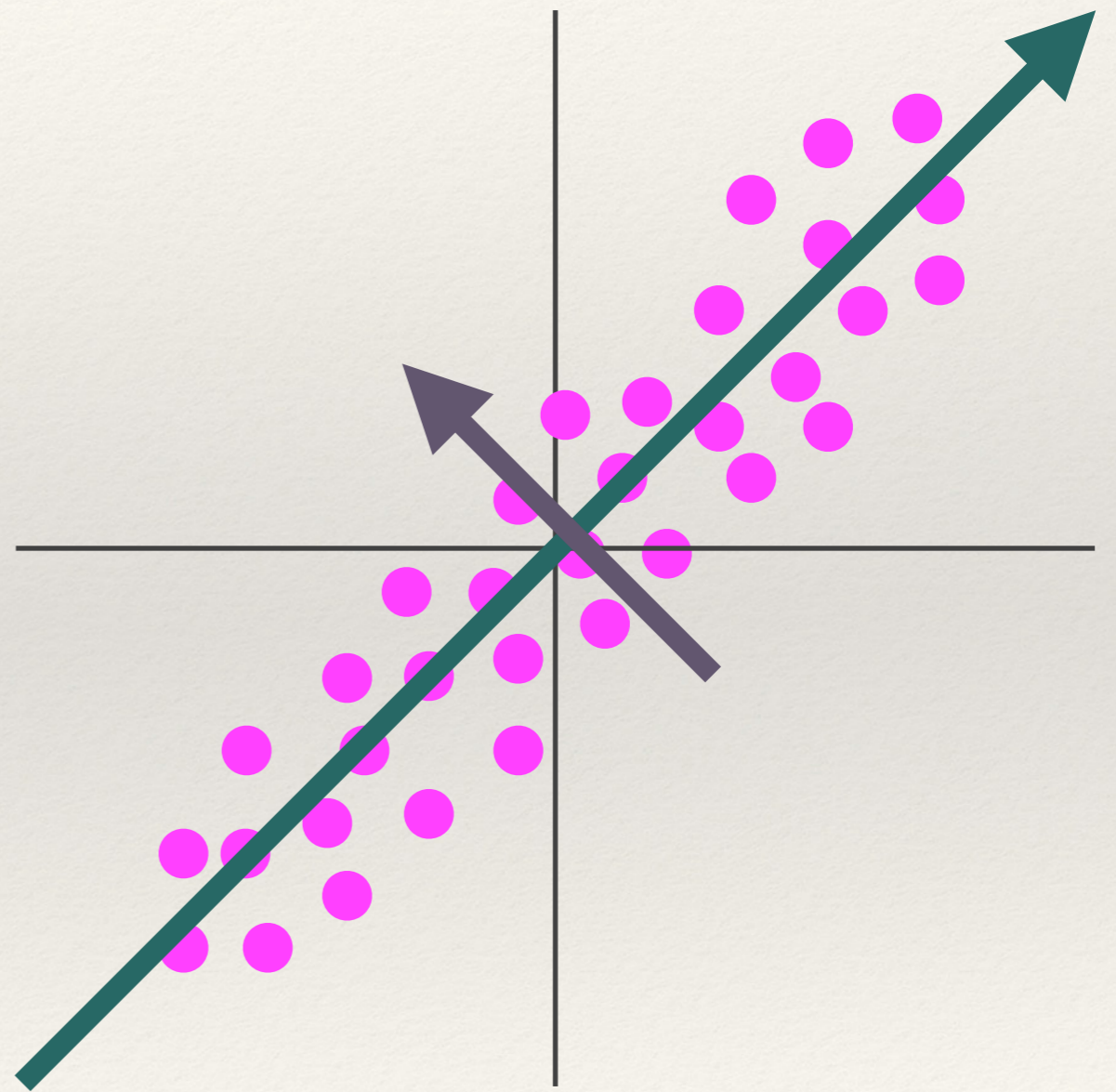
The first principal axis

- ❖ 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**.



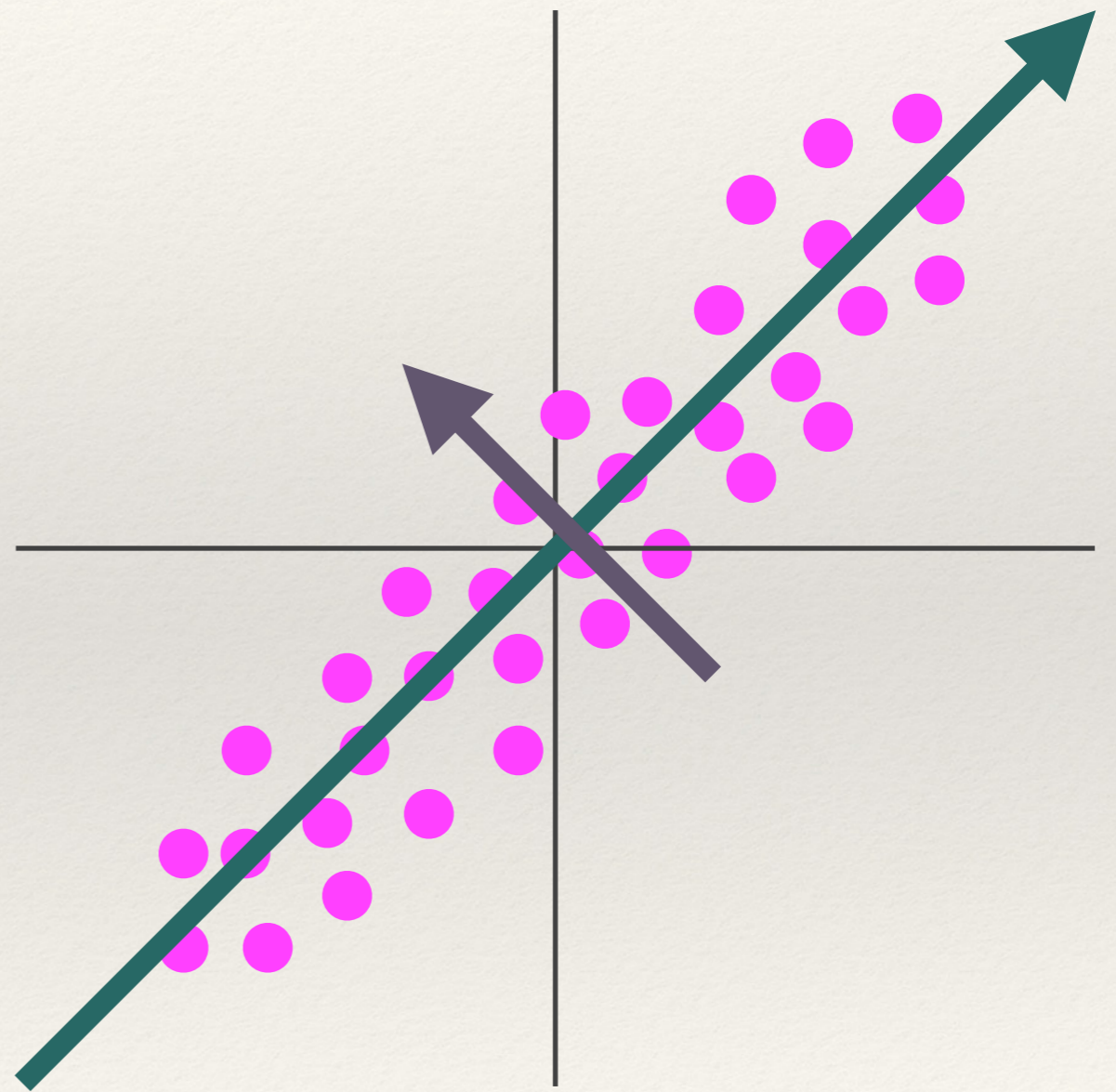
The second principal axis

- ❖ The **second principal axis** is a vector in the direction of greatest variance orthogonal (perpendicular) to the first major axis.



The third principal axis

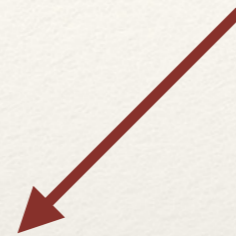
- ❖ In a space with 3 or more dimensions, the third principal axis is the direction of greatest variance orthogonal to both the first and second principal axes.
- ❖ The forth... .. and so on...
- ❖ The set of n **principal axes** of an n dimensional space are a **basis**



Eigenvectors, Eigenvalues and Eigendecomposition

Eigenvectors and Eigenvalues

Very important equation!



$$Av = \lambda v$$



Eigenvectors and Eigenvalues

a $n \times n$ square matrix

a scalar value,
known as an
eigenvalue

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

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



Eigenvectors and Eigenvalues

$$Av = \lambda v$$

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

$$Av = \lambda v$$

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 \mathbf{Q} are the eigenvectors

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

diagonal eigenvalue matrix ($\Lambda_{ii} = \lambda_i$)



Eigendecomposition

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

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

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



Eigendecomposition

In summary, the Eigendecomposition
of a covariance matrix \mathbf{A} :

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

Gives you the principal axes and
their relative magnitudes



Ordering

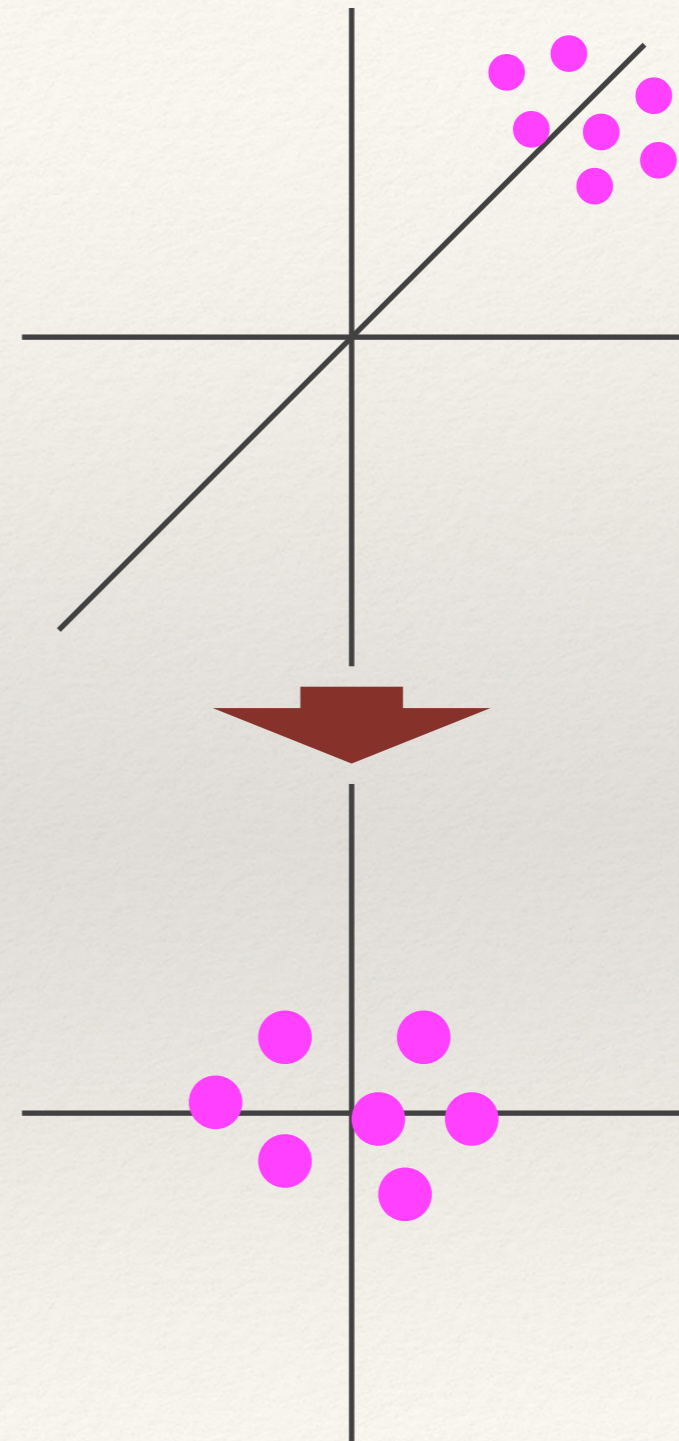
- ❖ Standard Eigendecomposition implementations will order the eigenvectors (columns of \mathbf{Q}) such that the eigenvalues (in the diagonal of $\mathbf{\Lambda}$) are sorted in order of decreasing value.
- ❖ Some solvers are optimised to only find the top k eigenvalues and corresponding eigenvectors,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

*Demo: Covariance,
Eigendecomposition and principal axes*

Principal Component Analysis

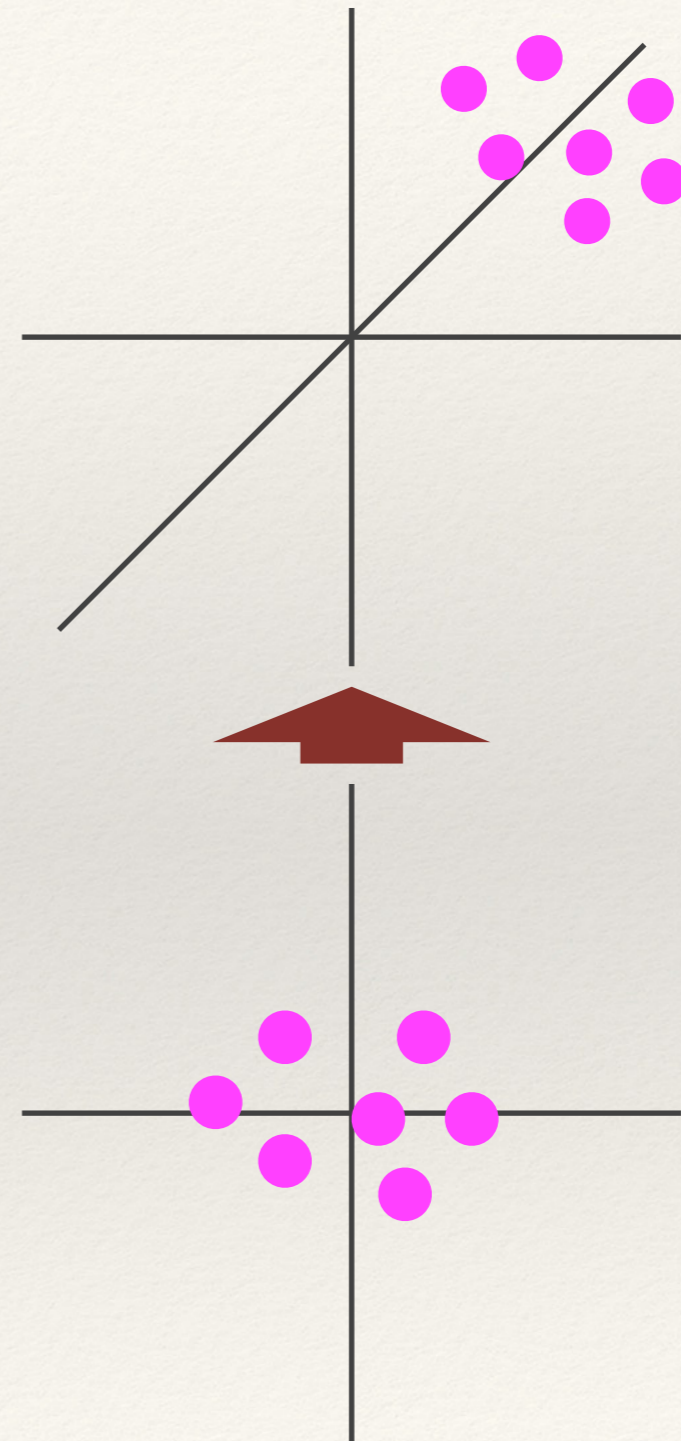
Linear Transform

- ❖ 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 Transforms

- ❖ The effects of a linear transform can be reversed if \mathbf{W} is **invertible**:
- ❖ $\mathbf{Z} = \mathbf{T}\mathbf{W}^{-1}$
- ❖ 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 \mathbf{W} is just the eigenvector matrix \mathbf{Q} from the Eigendecomposition of the covariance matrix of the data.
- ❖ Dimensionality reduction can be achieved by removing the eigenvectors with low eigenvalues from \mathbf{Q} (i.e. keeping the first L columns of \mathbf{Q} assuming the eigenvectors are sorted by decreasing eigenvalue).



PCA Algorithm

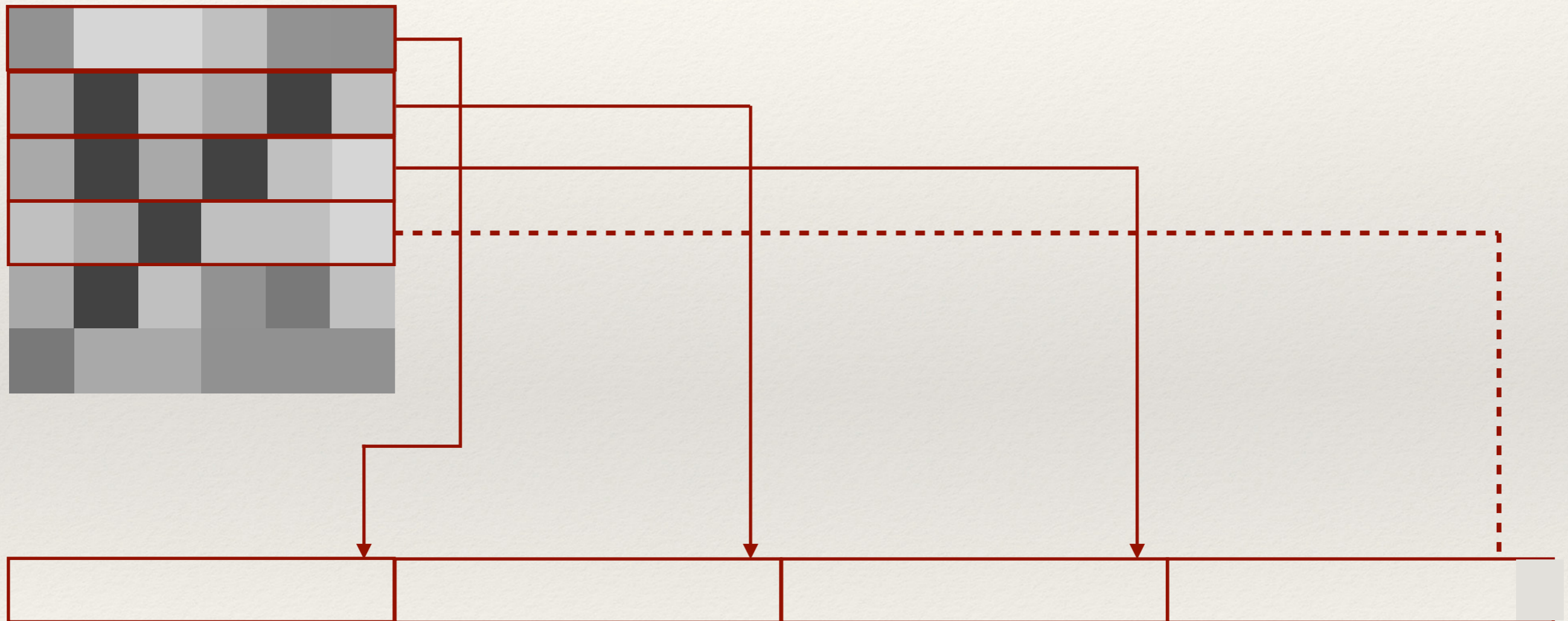
1. Mean-centre the data vectors
2. Form the vectors into a matrix \mathbf{Z} , such that each row corresponds to a vector
3. Perform the Eigendecomposition of the matrix $\mathbf{Z}^T\mathbf{Z}$, to recover the eigenvector matrix \mathbf{Q} and diagonal eigenvalue matrix $\mathbf{\Lambda}$: $\mathbf{Z}^T\mathbf{Z} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$
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 \mathbf{Q}_L .
6. Project the original vectors into a lower dimensional space, \mathbf{T}_L : $\mathbf{T}_L = \mathbf{Z}\mathbf{Q}_L$



Demo: PCA

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 misalignments

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.

Demo: a dataset of faces

Demo: mean-face

Demo: mean-centred faces

Demo: principal components

Demo: reconstructed faces

Demo: reconstruction from weights

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.