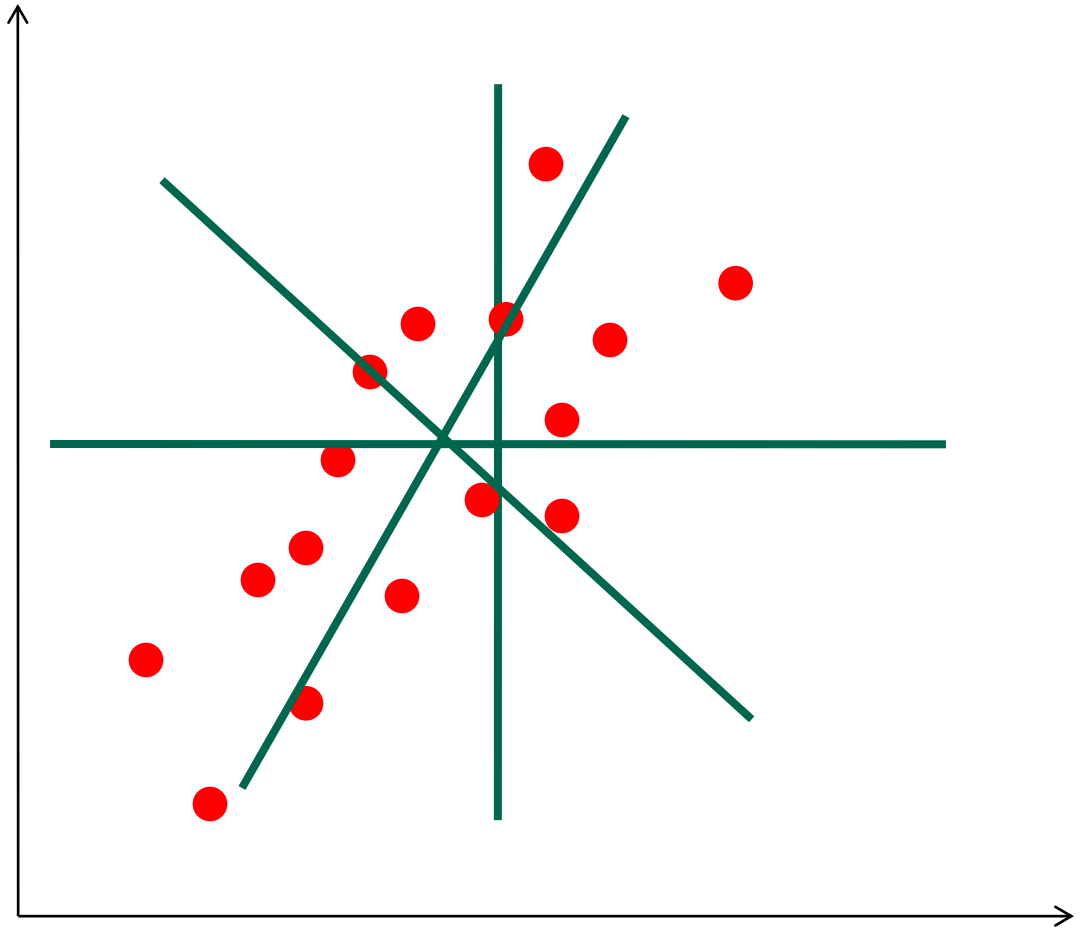
Prof. Marios Savvides

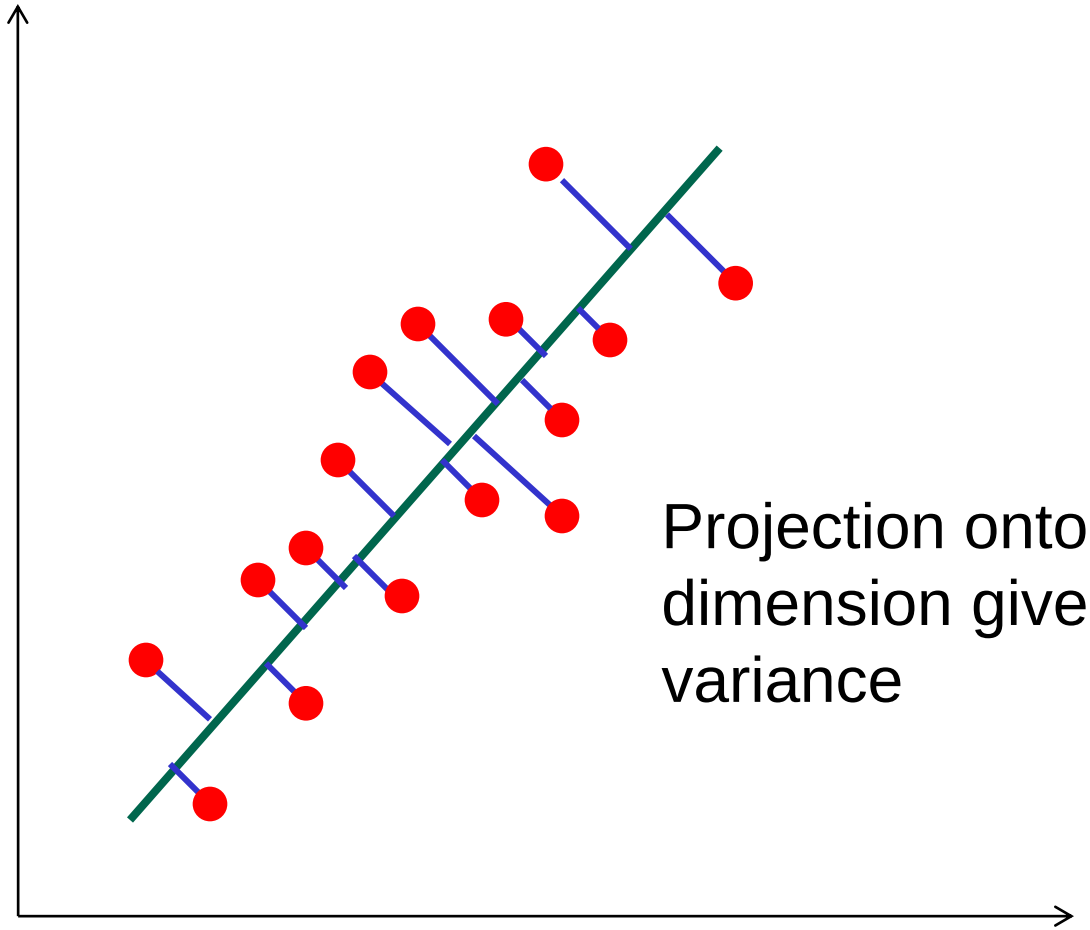
Pattern Recognition Theory

Lecture 7 : Principal Component Analysis II (PCA II)

PCA - Recap



PCA - Recap



Projection onto this dimension gives the largest variance

PCA - Recap

- Find $\boldsymbol{\omega}$ that maximizes Var($\boldsymbol{\omega}^T\mathbf{x}$)

  Subject to the constraint that $\boldsymbol{\omega}$ is a direction vector of unit norm.

$$L(\omega, \lambda) = \omega^T \Sigma \omega - \lambda\left(\omega^T \omega - 1\right)$$

  Taking derivative and set to zero. We see that this reduces to an eigen value problem

$$\Sigma\omega = \lambda\omega$$

Points to Note

- $\omega^T \Sigma \omega$ is the variance in the direction ω
- The coraviance matrix is symmetric, so has ORTHOGONAL eigenvectors.

$$\omega_i^T \omega_k = 0 \quad \text{for all } i \neq k$$
$$\omega_i^T \omega_k = 1 \quad \text{for all } i = k \quad (\omega^T \omega = 1)$$

- It is positive semidefinite, so has only real and positive eigenvalues.

- They are uncorrelated, and information about how data varies in the direction of one of the eigenvectors tells you nothing about how data varies in the directions of the eigenvectors.

- This means we have an orthogonal basis that we can use to represent the data. Normalize the eigenvectors to unit norm and you will have an orthonormal basis.

Expanding a Signal Using a Basis

- Assume you have a discrete vector signal x
- You have a set of N basis vectors $\mathbf{v}_i$ which you can use to represent a signal as follows

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{V}\mathbf{p}$$

- The signal is a linear combination of the basis vectors where the p_i are coefficients.

$$\mathbf{V} = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \qquad \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$$

Computing the Weight Coefficients

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{V}\mathbf{p}$$

- Since this is an orthogonal basis, we can easily find the coefficients p. These are just the projections of the signal onto each basis.

$$\begin{aligned} p_1 &= \mathbf{x}^{\mathbf{T}}\mathbf{v}_1 \\ p_2 &= \mathbf{x}^{\mathbf{T}}\mathbf{v}_2 \\ &\vdots \\ p_n &= \mathbf{x}^{\mathbf{T}}\mathbf{v}_n \end{aligned}$$

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} \leftarrow & \mathbf{v}_1 & \rightarrow \\ \leftarrow & \mathbf{v}_2 & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{v}_n & \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow \\ \mathbf{x} \\ \downarrow \end{bmatrix} = \mathbf{V}^{\mathbf{T}}\mathbf{x}$$

Model the Variations About the Mean

- Don't forget that you model the variance about the mean!

  Don't forget to subtract the global mean before you analyze your test data.

$$\mathbf{p} = \mathbf{V}^{\mathbf{T}}\left(\mathbf{x} - \mathbf{m}\right)$$

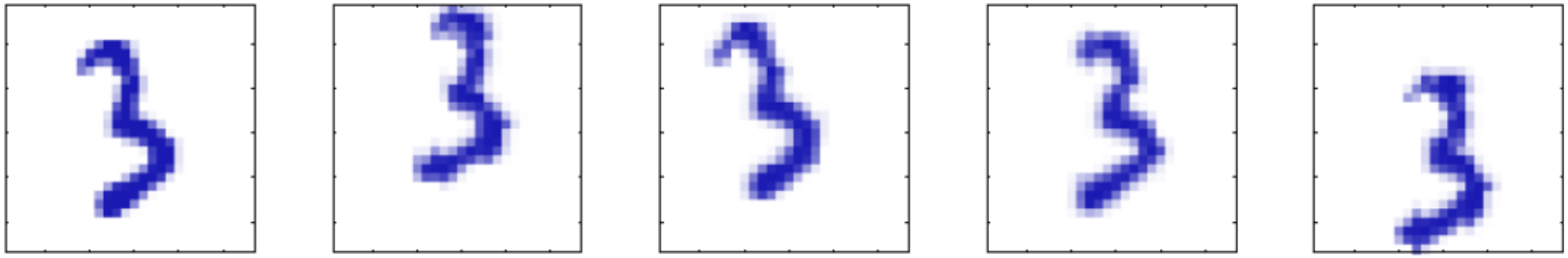
- The mean is the mean of all your training data samples.

- Don't forget to add the mean back before you reconstruct the data !

$$\mathbf{x} = \sum_{i=1}^{n} p_i \mathbf{v_i} + \mathbf{m} = p_1 \begin{bmatrix} \uparrow \\ \mathbf{v_1} \\ \downarrow \end{bmatrix} + p_2 \begin{bmatrix} \uparrow \\ \mathbf{v}_2 \\ \downarrow \end{bmatrix} + \cdots + p_n \begin{bmatrix} \uparrow \\ \mathbf{v}_n \\ \downarrow \end{bmatrix} + \mathbf{m} = \mathbf{Vp} + \mathbf{m}$$

A Visual Example

Consider a hand-written digit 3

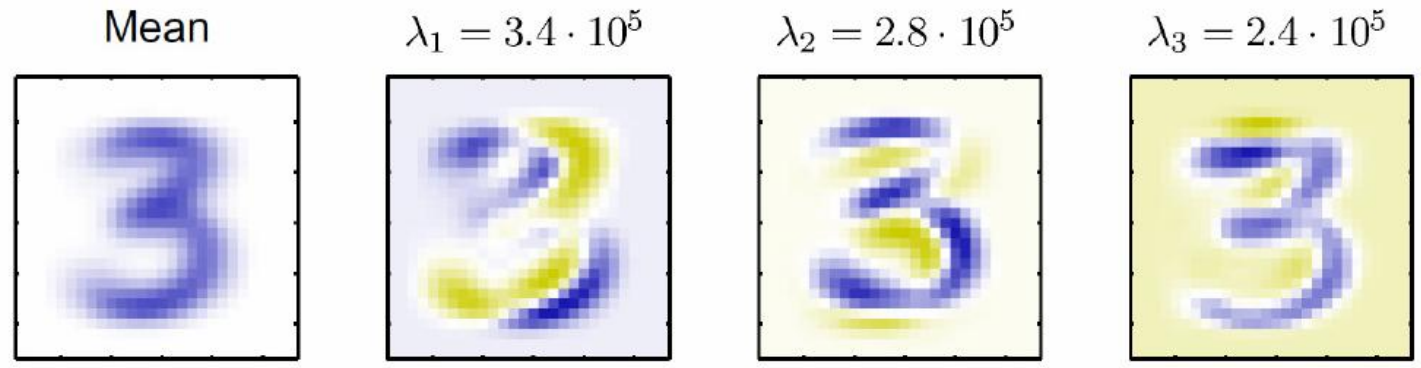


•28x28 pixel images → number of dimensions, d = 784.

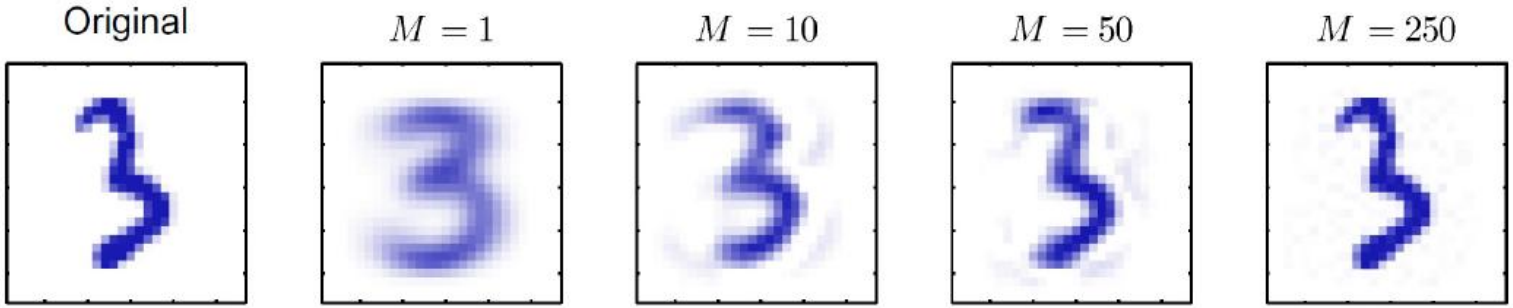
•These images were constructed by translating and rotating one image.

A ‘visual’ example

The mean image and the eigen vectors corresponding to the largest three eigen values



Reconstructing the digit using the first M eigen vectors

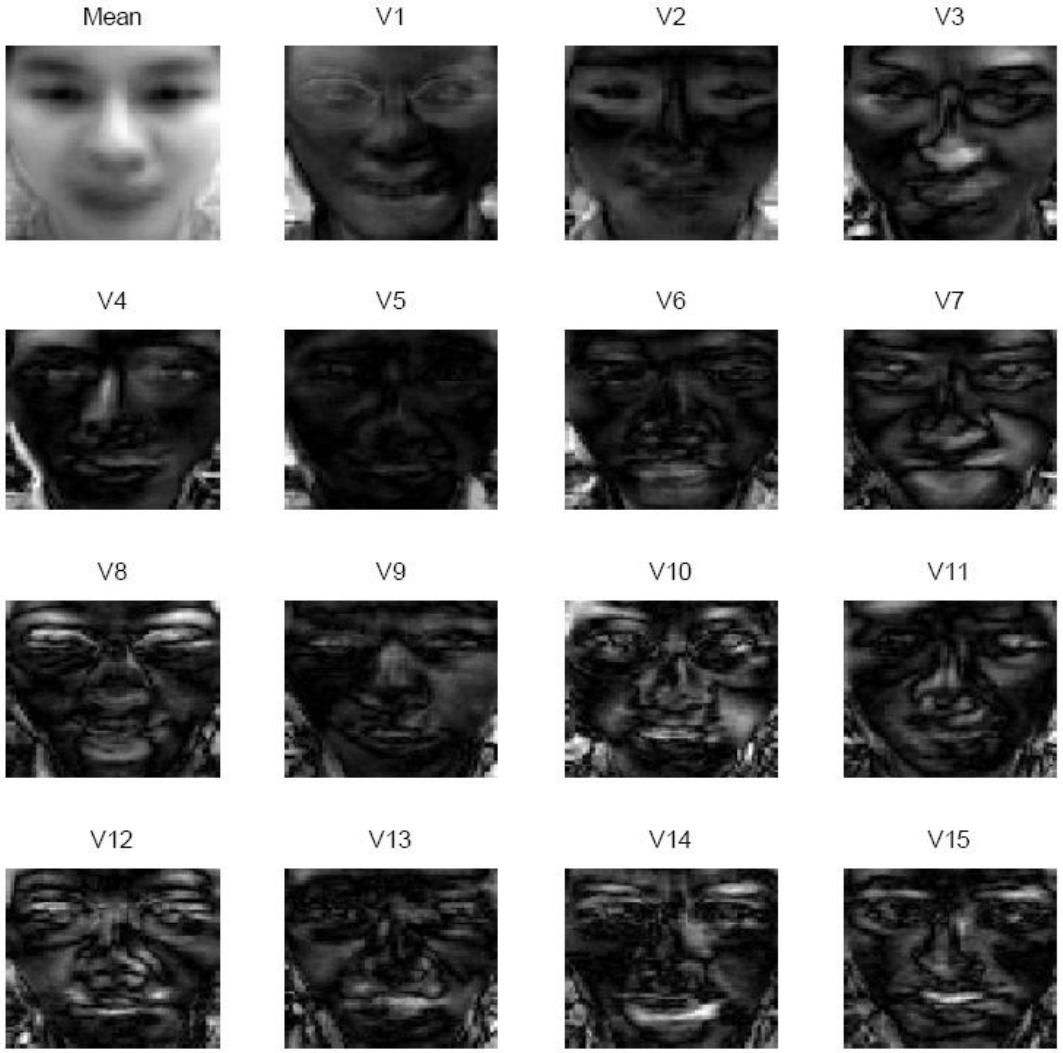


Another Example: Human Faces

- AMP Lab face expression database
- 13 people
- Images are 64x64 cropped and aligned.
- Variations are due to varying expressions in the video sequence.
- 75 images in each person's sequence

- Experiment
  - Take first 5 images per person as training
  - Total 5x13=65 training images
  - Do PCA and extract basis eigvectors
  - Measure reconstruction ability
  - Perform dimensionality reduction to 3D (take only the first 3 eigenvectors and look at how the data clusters)

PCA Example: Eigenfaces



How Much Reconstruction Error

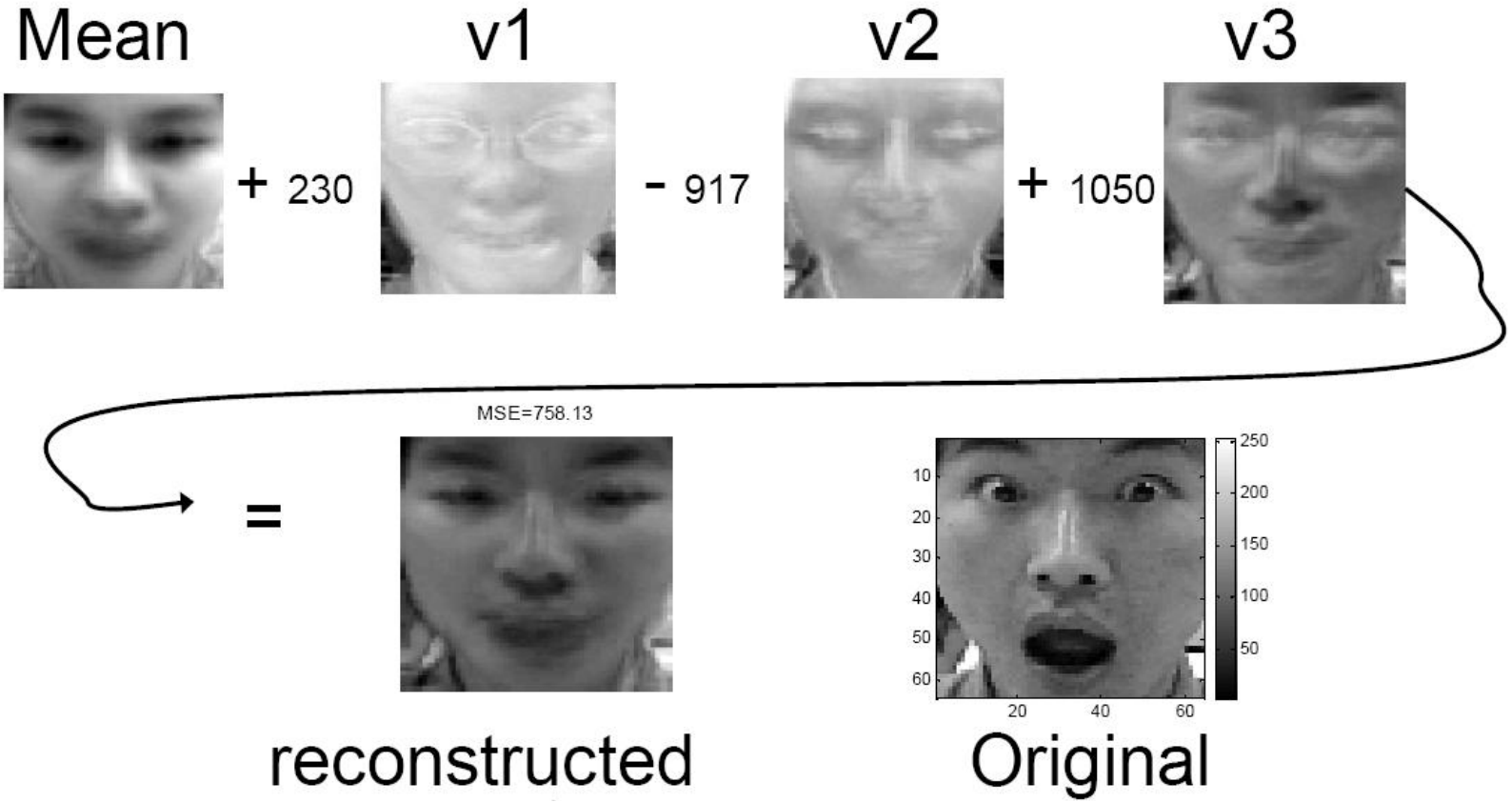
- If you want to say reconstruct an image with only 20% MSE (or 80% reconstruction)
- Then use the eigenvectors with largest M eigenvalues from the total of N non-zero eigenvalues satisfying this ratio:

$$reconstruction\% = \frac{\sum_{i=1}^{M} \lambda_i}{\sum_{i=1}^{N} \lambda_i} * 100$$

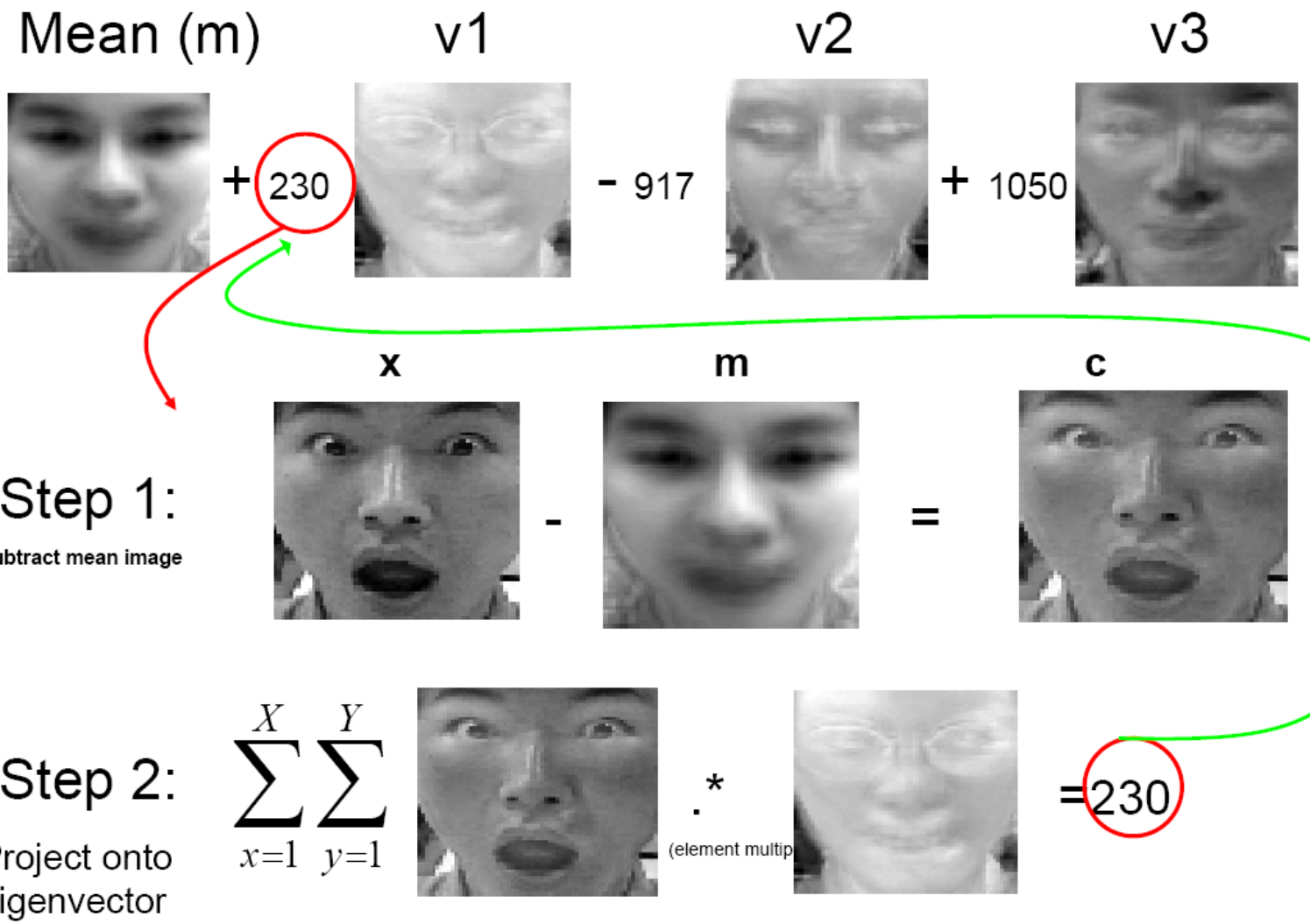
Sum of the eigenvalues of the kept eigenvectors

Sum of all eigenvalues

Using Only 3 Most Dominant Eigenvectors



Computing Coefficients



Reconstruction

Feature Extraction Via Dimensionality Reduction

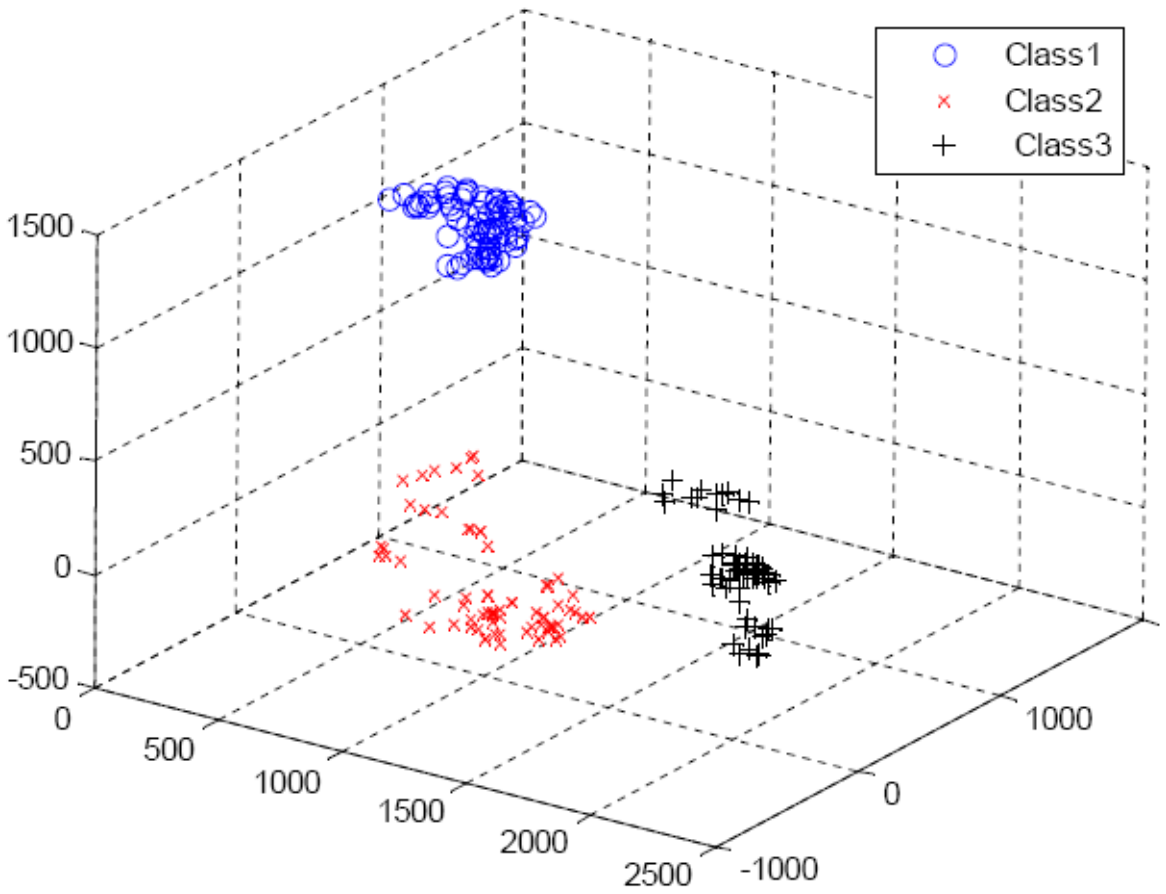
- Dimensionality reduction ➔ throw away many eigenvectors and use only a subset (corresponding to the more dominant ones)
- Project the data samples onto these vectors and store the projection coefficients for each data sample → we thus obtain a **feature vector.**
- Why feature reduction? – because now we need to store only the projection coefficients and not the entire image!

Dimensionality Reduction to 3D Subspace

How to classify?

•Project onto the first 3 eigen vectors. Now we can visualize the coefficients in a 3D space.

•Use Nearest-Neighbor search (based on a distance metric) to all stored vectors and find the closest in order to classify.



Practical Issues in Computing PCA

- Assume we have image data of size 200x200 pixels.
- This means our data vector is d=40,000 dimensions.
- The size the covariance matrix will be 40,000x40,000
- This is a BIG MATRIX. You will run out of memory when you try to create a matrix this big.
- Also when usually we have N training samples, where N<<d.
- We will have at most N-1 eigenvectors and nonzero-eigenvalues.
- Computing the big 40,000x40,000 matrix is therefore a waste.

The Gram Matrix Trick

- We know that $\boldsymbol{\Sigma} = E\left[(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^{\mathbf{T}}\right] = \mathbf{X}\mathbf{X}^{\mathbf{T}}$

- WE must solve

$$\boldsymbol{\Sigma}\mathbf{v} = \lambda\mathbf{v}$$

$$\mathbf{X}\mathbf{X}^{\mathbf{T}}\mathbf{v} = \lambda\mathbf{v}$$

Premultiply by X^T →

$$\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{X}^{\mathbf{T}}\mathbf{v} = \lambda\mathbf{X}^{\mathbf{T}}\mathbf{v}$$

Replace $\mathbf{X}^{\mathbf{T}}\mathbf{v}$ by $\mathbf{v}'$ →

$$\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{v}' = \lambda\mathbf{v}'$$ ← Now solve this new eigen value/eigen vector problem

- X^TX is a gram or inner-product matrix. It is of size NxN, which is not dependent on the dimensionality of the data (d), but rather on the number of data samples N. It will be easier to compute if N<<d.

The Gram Matrix Trick

- Now that we've obtained all the eigenvectors v' of the Gram matrix, how do we obtain the eigenvectors of the covariance matrix XX^T ?
- Remember from the previous equations that:

$$XX^T\mathbf{v} = \lambda\mathbf{v} \quad \text{and} \quad \mathbf{v}' = X^T\mathbf{v}$$

- Substitute the second equation in the first equation we obtain

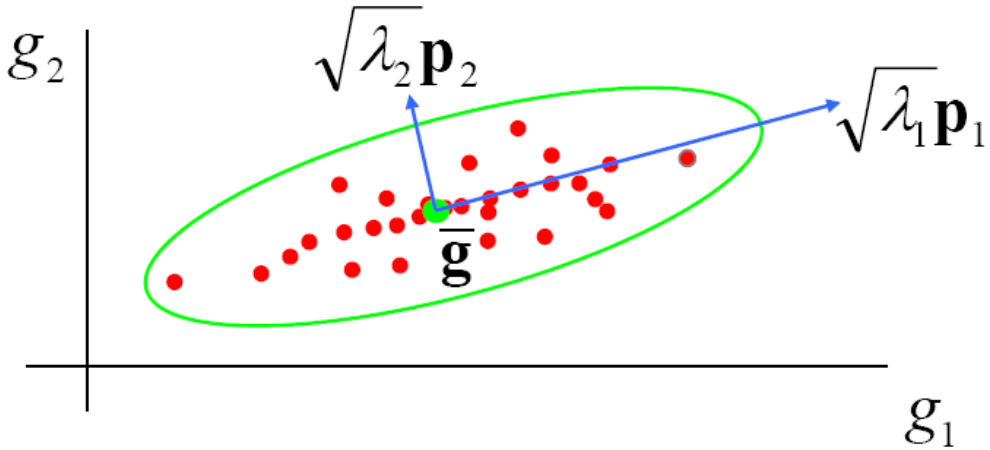
$$X\mathbf{v}' = \lambda\mathbf{v}$$

- Thus $\mathbf{v} \cong X\mathbf{v}'$. We do not care about the scaling term because we will unit-normalize the eigenvectors anyway to obtain an orthonormal basis.

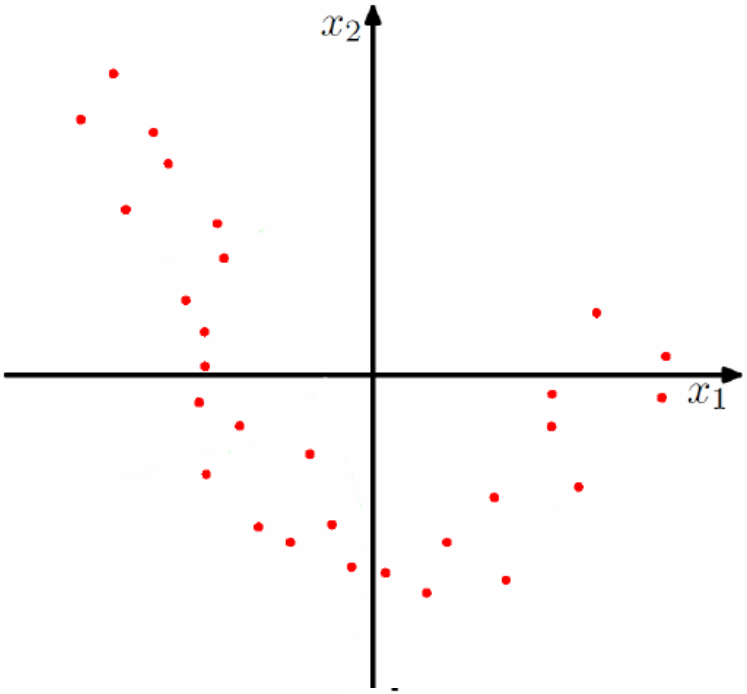
PCA: Afterthoughts

- The mean and the covariance specify a Gaussian model
- So PCA seems to be related to fitting a Gaussian model to the data
  - Find eigen vectors & eigen values of covariance matrix
  - Transform the original coordinate system by translation and rotation into a new space
    - Mean becomes origin
    - "Largest" eigen vector becomes 1st axis, second "largest" eigvec becomes 2nd axis, and so on.

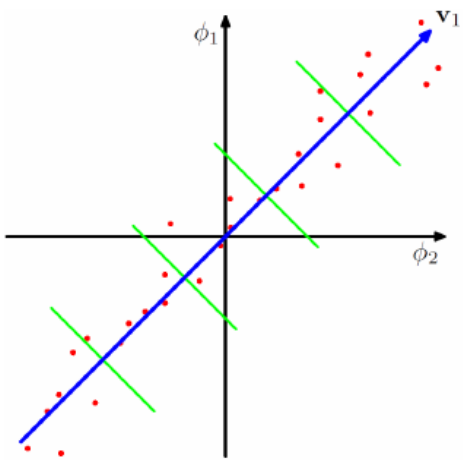
- PCA de-correlates the data.
- Dimensionality reduction is obtained by using only a subset of the new axes that account for most of the variance.

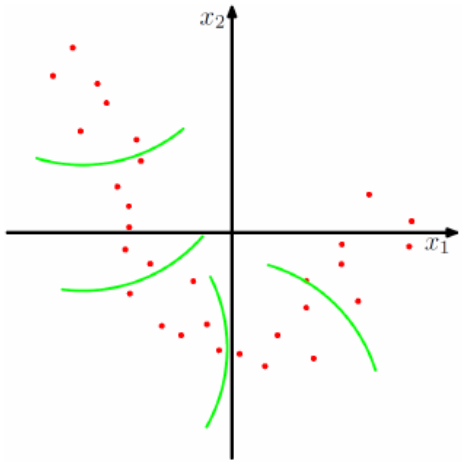


Later: Kernel PCA

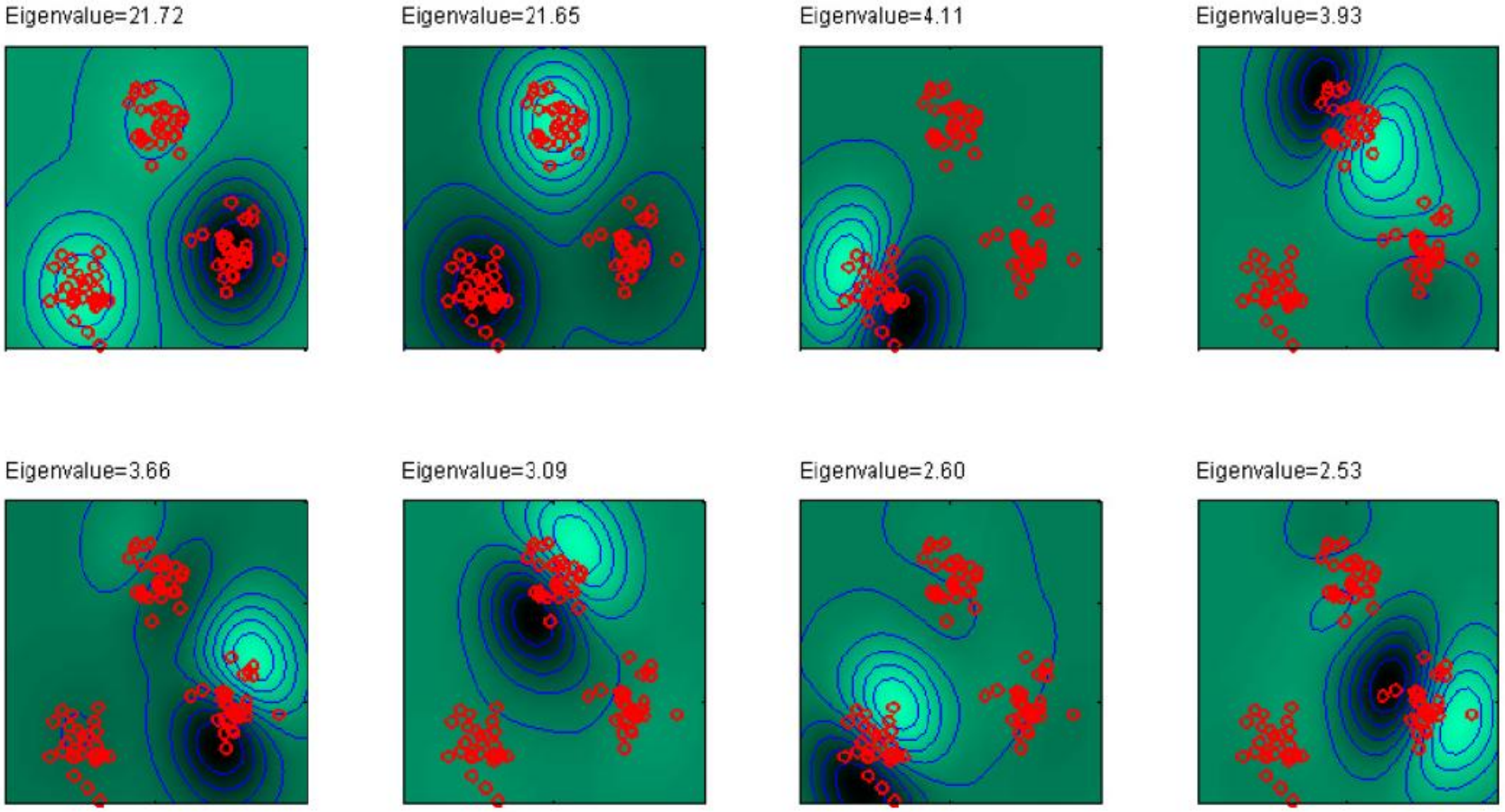


- If data is not normally distributed, it could be much more efficient to project the data on a curve
- Finding this curve is much harder than the linear case→ Kernels come to the rescue.
- Map data to higher dimensional space
- Do standard PCA there
- Corresponds to non-linear PCA in lower dimensional space.





Later: Kernel PCA



KernelPCA with Gaussian kernel. Data is clustered. First two components encode cluster. Higher components encode structure within clusters

Recap

- PCA
- Using an Eigen basis
- Reconstruction Error
- PCA on Faces
- Gram Trick