Cmput 466 / 551

Principle Component Analysis

HTF: Ch14 (parts)

R Greiner
Department of Computing Science
University of Alberta

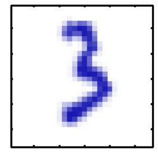
http://www.quora.com/Mathematics/What-do-eigenvalues-and-eigenvectors-represent-intuitively

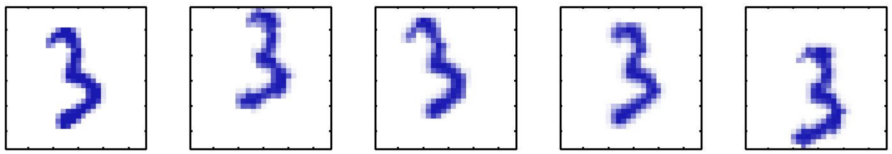
Thanks: Tom Mitchell… via Ron Parr

Outline

- Motivation
  - Dimensionality reduction
  - … while preserving “variance”
- Formal definition
  - Eigenvalues/vectors, …
- Example: Eigenfaces
- Why run PCA?
  - Data Compression
  - Anomaly Detection
  - Preprocessing for Supervised Learning

Inherent Dimensionality

- If ONLY CONSIDERING 
  how many dimensions needed to describe



- #pixels??
  - 28 x 28 = 784 ?
- But only “3” dimensions
  - vertical translation
  - horizontal translation
  - rotation

Principle Components Analysis

Idea:

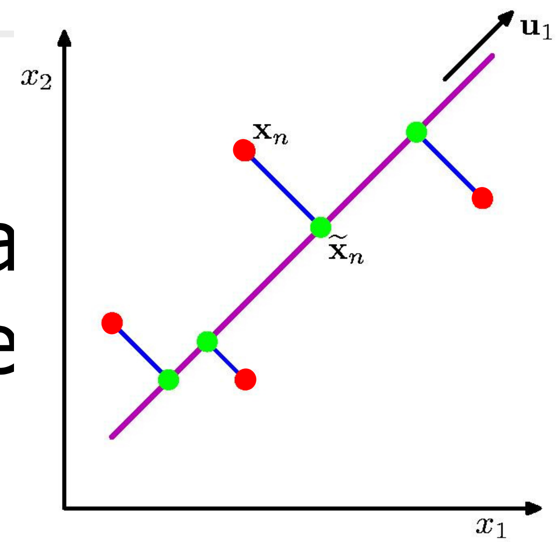
- Given data points in d-dimensional space,
  - project into *lower dimensional* space
  - while *preserving as much information* as possible
- Eg
  - find best planar approximation to 3D data
  - find best 12-D approximation to 10^4-D data

$\Rightarrow$ choose projection that

minimizes *squared error*

in reconstructing original data

Principle Component Analysis



PCA ≡ Orthogonal projection of data onto lower-dimension linear space that...

- maximizes variance of projected data
  - purple line
- minimizes average projections ≡ mean squared distance between data point and projections
  - sum of (squares of) blue lines

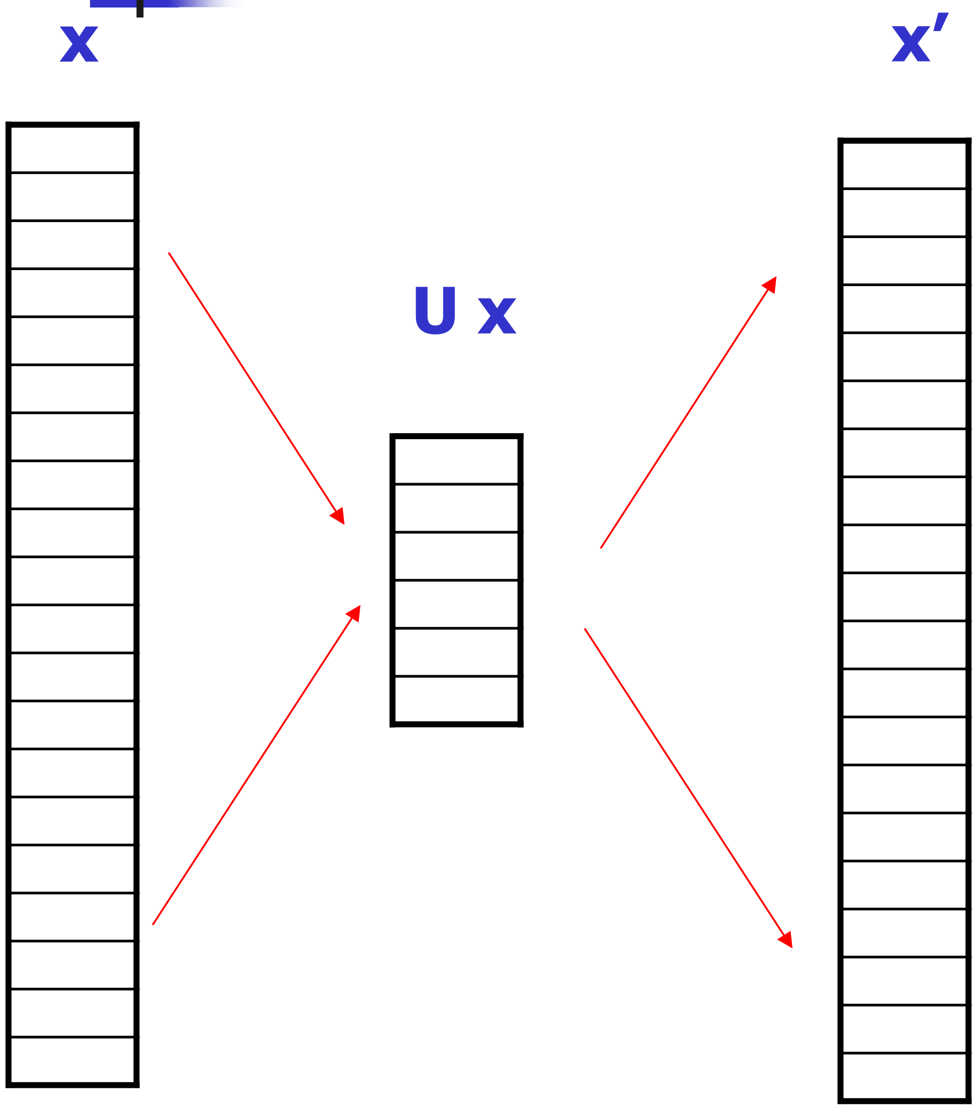
Challenge: Facial Recognition

- Want to identify specific person, based on facial image
- Robust to …
  - Facial hair, glasses, …
  - Different lighting

⇒ Can’t just use given 256 x 256 pixels

- Need another option!

PCA ≈ Auto-Encoder…



- Quality (minimize):
  || **x** – **x’** ||
- Eg, measure “faceness” of image
- … using linear transforms …

Reduce Dimensional… lossy…

⇗ [3.2, 12, -1.6] ⇒

⇒ [3.2, 12, -1.6,
4.1, -2.0, 5.1] ⇒

⇘ [3.2, 12, -1.6,
4.1, -2.0, 5.1,
-2.7, 2.1, -3.3, …] ⇒

Why do we care?

- Lower dimensional representations permit
  - Compression
  - Noise filtering
  - Visualization

- As preprocessing for classification:
  - Reduces feature space dimension
    - Simpler Classifiers
    - Efficiency
    - Possibly better generalization (?)
  - May facilitate simple methods
    - (nearest neighbor)

Review: Eigenvectors

- Each eigenvector $\mathbf{u}$ of matrix A satisfies:

$$\boxed{A\,\mathbf{u} = \lambda\,\mathbf{u}}$$

- For symmetric, full-rank A, eigenvectors…
  - … are orthogonal $\mathbf{u}_i^T\mathbf{u}_j = 0$ if $i \neq j$
  - … form an basis for A :

    For any $\mathbf{x}$, $\boxed{\mathbf{x} = \sum_i \alpha_i\,\mathbf{u}_i}$

  - Can be scaled s.t. $\mathbf{u}_i^T\mathbf{u}_i = 1$ (orthonormal)
  - Here: $\boxed{\alpha_i = \mathbf{u}_i^T\,\mathbf{x}}$

Review: Projection

- Orthonormal basis $\rightarrow$ trivial projection
- Given basis $U = \{ \mathbf{u}_1, \ldots, \mathbf{u}_k \}$
  can project any d-dim $\mathbf{x}$ to k values
  - $\alpha_1 = \mathbf{u}_1^T \mathbf{x} \quad \alpha_2 = \mathbf{u}_2^T \mathbf{x} \quad \ldots \quad \alpha_k = \mathbf{u}_k^T \mathbf{x}$
  - $\alpha = \mathbf{U}^T \mathbf{x}$
  - $\mathbf{x} \approx \sum_{i=1}^{k} \alpha_i \, \mathbf{u}_i = \sum_{i=1}^{k} (\mathbf{u}_i^T \mathbf{x}) \, \mathbf{u}_i$
    - "=" if $d=k$ (ie, all values)
- We will use "centered" vectors:
  $\mathbf{x}' = \mathbf{x} - \bar{\mathbf{x}}$ where $\bar{\mathbf{x}} = \frac{1}{M} \sum_{n=1}^{M} \mathbf{x}^{(n)}$

$$\boxed{\alpha_i = \mathbf{u}_i^T (\mathbf{x} - \bar{\mathbf{x}})}$$

PCA: Find Projections to Minimize Reconstruction Error

- Given set of M *d*-dim vectors $\mathbf{x}^{(n)} = [\, x_1^n, \dots, x_d^n \,]$
- Can represent each using any d orthogonal basis vectors

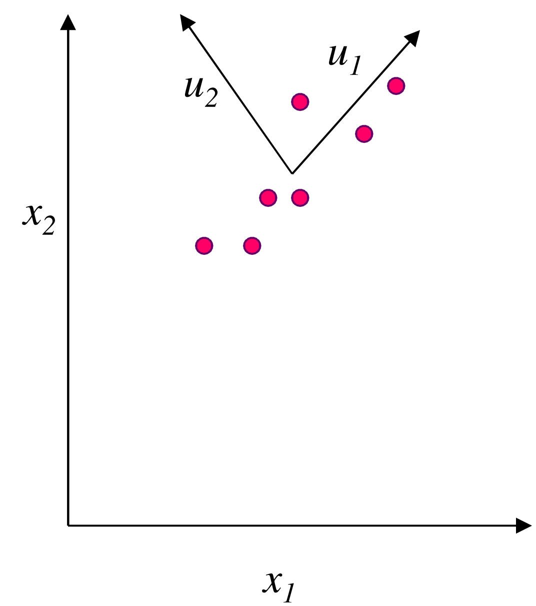
$$\mathbf{x}^{(n)} = \sum_{i=1}^{d} \alpha_i^{(n)} \mathbf{u}_i \qquad \mathbf{u}_i^{\top} \mathbf{u}_j = \delta_{ij}$$

PCA:
- Given k<d.
- Find orthogonal basis { $\mathbf{u}_1, \dots, \mathbf{u}_k$ }

  that minimizes $E_k = \sum_{n=1}^{M} \left| \mathbf{x}^{(n)} - \hat{\mathbf{x}}_k^{(n)} \right|^2$
- where $\hat{\mathbf{x}}_k^{(n)} = \bar{\mathbf{x}} + \sum_{i=1}^{k} \alpha_i^{(n)} \mathbf{u}_i$

Mean

$$\bar{\mathbf{x}} = \frac{1}{M} \sum_{n=1}^{M} \mathbf{x}^{(n)}$$

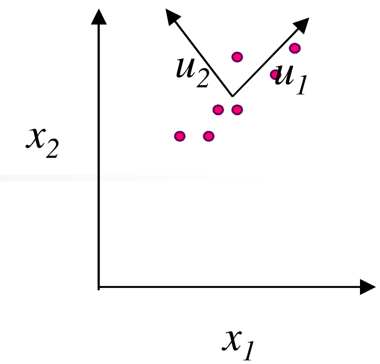


PCA

PCA:

- Given k<d.
- Find orthogonal basis { $\mathbf{u}_1, \ldots, \mathbf{u}_k$ }

  that minimizes $E_k = \sum_{n=1}^{M} \left| \mathbf{x}^{(n)} - \hat{\mathbf{x}}_k^{(n)} \right|^2$
- where $\hat{\mathbf{x}}_k^{(n)} = \bar{\mathbf{x}} + \sum_{i=1}^{k} \alpha_i^{(n)} \mathbf{u}_i$



- Note $\hat{\mathbf{x}}_d^{(n)} = \bar{\mathbf{x}} + \sum_{i=1}^{d} \alpha_i^{(n)} \mathbf{u}_i \equiv \mathbf{x}^{(n)}$
- So... $\mathbf{x}^n - \hat{\mathbf{x}}_k^{(n)} = \sum_{i=k+1}^{d} \alpha_i^{(n)} \mathbf{u}_i = \sum_{i=k+1}^{d} \left((\mathbf{x}^{(n)} - \underline{\mathbf{x}})^T \mathbf{u}_i \right) \mathbf{u}_i$
- $E_k = \| \mathbf{x}^n - \hat{\mathbf{x}}_k^{(n)} \|^2 = \sum_{n=1}^{M} \left\| \sum_{i=k+1}^{d} \left((\mathbf{x}^n - \underline{\mathbf{x}})^T \mathbf{u}_i \right) \mathbf{u}_i \right\|^2 = \sum_{n=1}^{M} \sum_{i=k+1}^{d} \left[(\mathbf{x}^n - \underline{\mathbf{x}})^T \mathbf{u}_i \right]^2$

$$= \sum_{i=k+1}^{d} \sum_{n=1}^{M} \left[\mathbf{u}_i^T (\mathbf{x}^n - \underline{\mathbf{x}}) \right] \left[(\mathbf{x}^n - \underline{\mathbf{x}})^T \mathbf{u}_i \right]$$

$$= \sum_{i=k+1}^{d} \mathbf{u}_i^T \, \Sigma \, \mathbf{u}_i$$

Covariance matrix:

$$\Sigma = \sum_{n} (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$$

Justifying Use of Eigenvectors

- Goal
  - minimize: $\mathbf{u}^T \Sigma \mathbf{u}$
  - subject to: $\mathbf{u}^T \mathbf{u} = 1$
- Use Lagrange Multipliers… minimize:

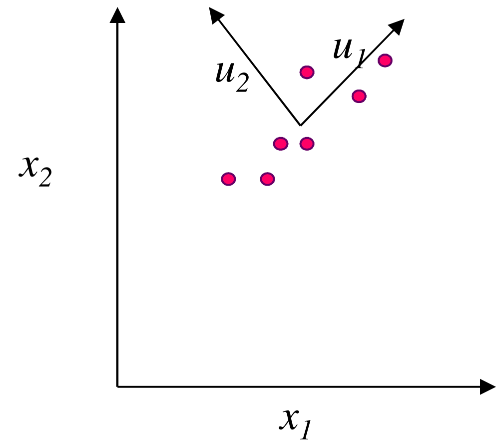
  $$f(\mathbf{u}) = \mathbf{u}^T \Sigma \mathbf{u} - \lambda[\mathbf{u}^T \mathbf{u} - 1]$$

- Set derivative to 0: $\boxed{\Sigma \mathbf{u} - \lambda \mathbf{u} = 0}$

- Def'n of eigenvalue λ, eigenvector $\mathbf{u}$!

- If multiple vectors $\mathbf{u}_i$:
  - Minimize sum of independent terms…
  - Each is eigen value/vector

PCA



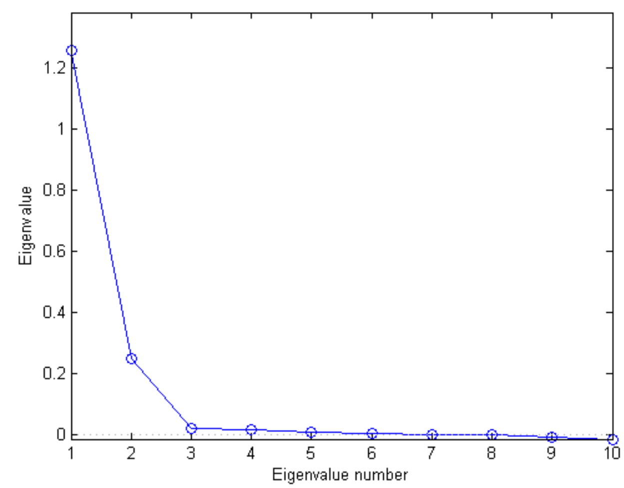
Minimize $$E_k = \sum_{i=k+1}^{d} \mathbf{u}_i^T \Sigma \mathbf{u}_i$$

$$\rightarrow \Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

Eigenvalue Eigenvector

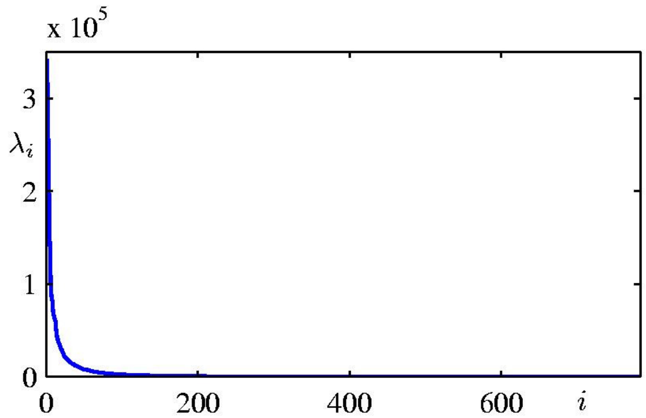
$$\Rightarrow E_k = \sum_{i=k+1}^{d} \mathbf{u}_i^T \Sigma \mathbf{u}_i = \sum_{i=k+1}^{d} \mathbf{u}_i^T \lambda_i \mathbf{u}_i$$

$$= \sum_{i=k+1}^{d} \lambda_i \mathbf{u}_i^T \mathbf{u}_i = \sum_{i=k+1}^{d} \lambda_i$$

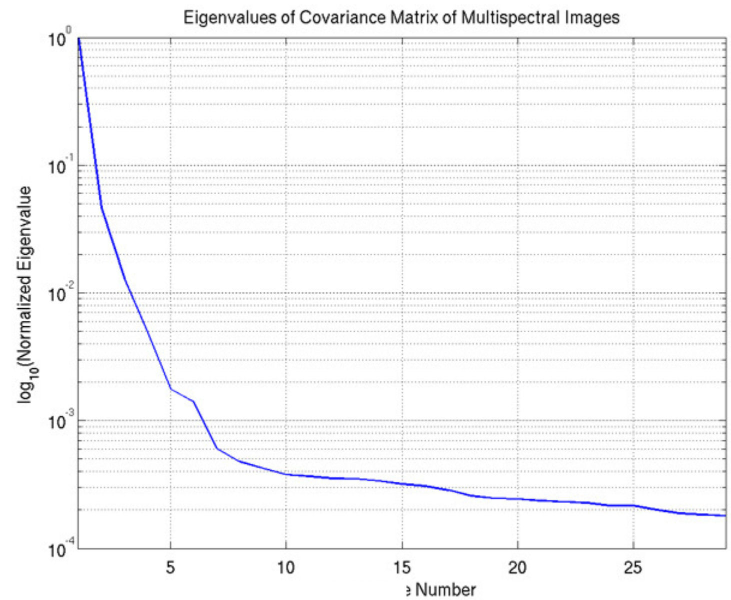


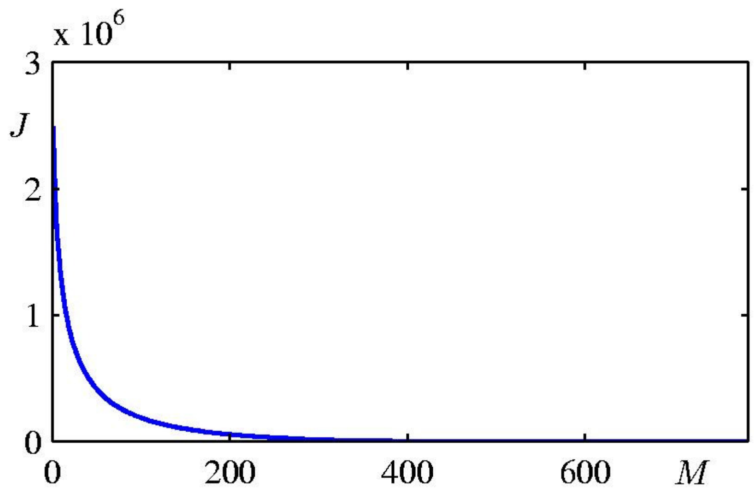
So... to minimize E_k, take SMALLEST eigenvalues { λ_i }

Eigenvalues (sorted)



(a)





(b)

PCA Algorithm

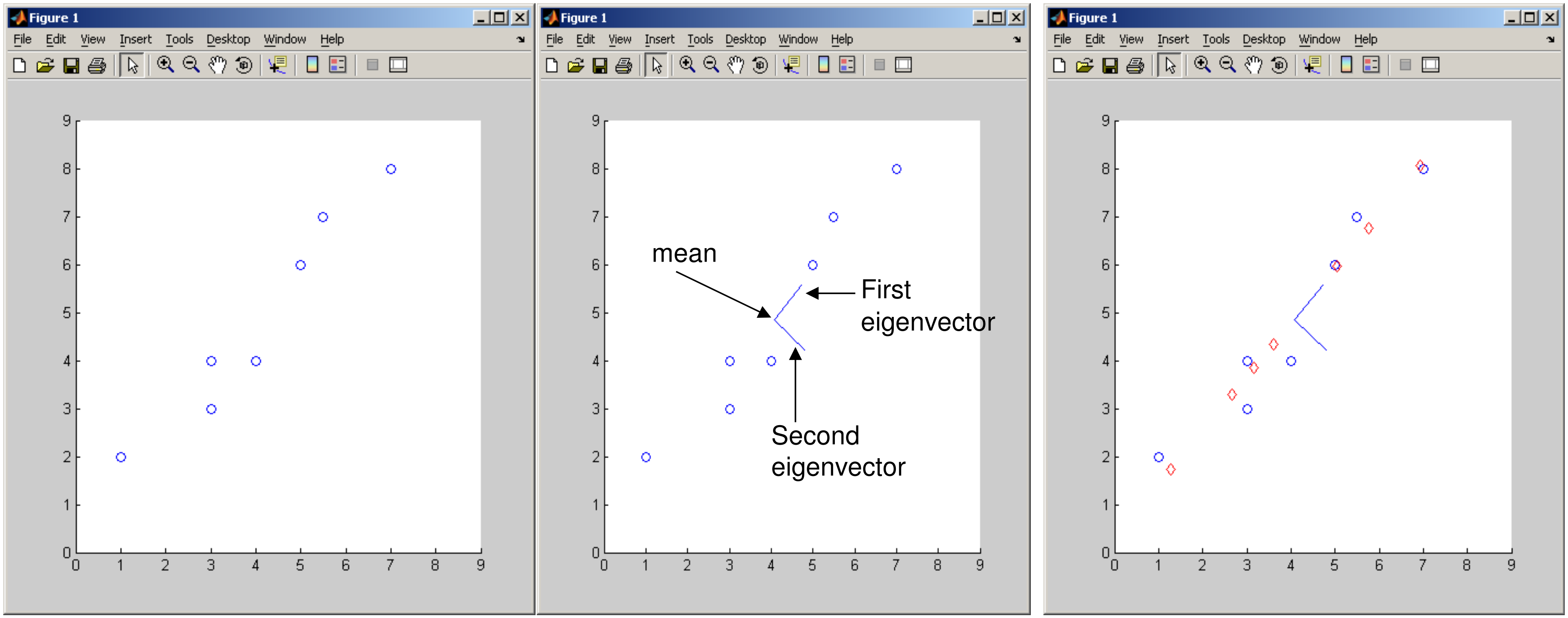
PCA algorithm($\mathbf{X}$, k): top k eigenvalues/eigenvectors

% $\mathbf{X}$ = d $\times$ N data matrix,
% … each data point $\mathbf{x}^{(n)}$ = column vector

- $\bar{\mathbf{x}} = \frac{1}{M} \sum_{n=1}^{M} \mathbf{x}^{(n)}$
- $\mathbf{A}$ $\leftarrow$ subtract mean $\bar{\mathbf{x}}$ from each column vector $\mathbf{x}^{(n)}$ in $\mathbf{X}$
- $\Sigma \leftarrow \mathbf{A}\,\mathbf{A}^T$ … covariance matrix of $\mathbf{A}$
- $\{ (\lambda_i, \mathbf{u}_i) \}_{i=1..d}$ = eigenvectors/eigenvalues of Σ
  … $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$
- Return $\{ \lambda_i, \mathbf{u}_i \}_{i=1..k}$
  % top k principle components

PCA Example

$$\hat{\mathbf{x}}_k^{(n)} = \overline{\mathbf{x}} + \sum_{i=1}^{k} \alpha_i^{(n)} \mathbf{u}_i$$



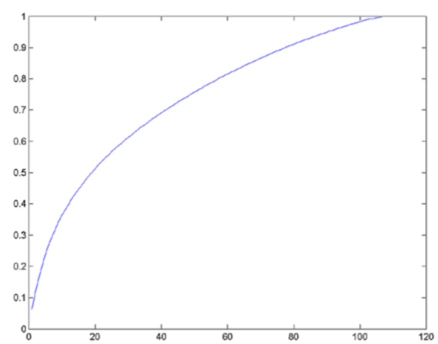
Reconstructed data using only first eigenvector (k=1)

Percentage of Variance

- Recall "error": $\sum_{n=1}^{M} |\mathbf{x}^{(n)} - \hat{\mathbf{x}}_k^{(n)}|^2$
- Compare with total variation of the data?

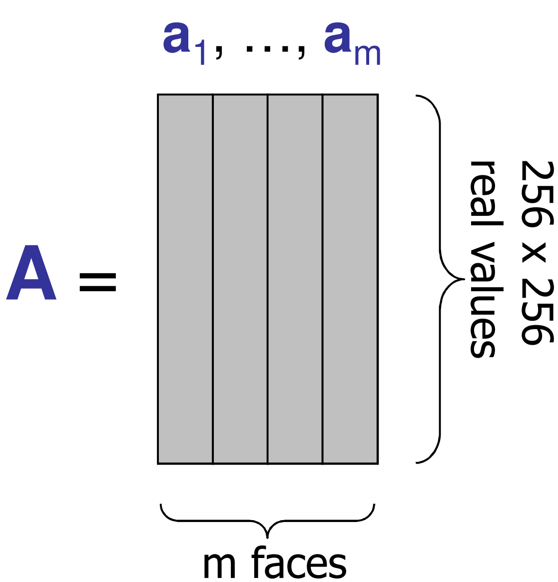
$$\sum_{n=1}^{M} |\mathbf{x}^{(n)}|^2$$

- PercentageVariance $PV(k) = \frac{\sum_{n=1}^{M} |\mathbf{x}^{(n)} - \hat{\mathbf{x}}_k^{(n)}|^2}{\sum_{n=1}^{M} |\mathbf{x}^{(n)}|^2}$
- $PV(k) < 0.01 \Rightarrow$ "99% of variance is retained"
- Note: $PV(k) = \frac{\sum_{i=k+1}^{d} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$



Applying PCA: Eigenfaces

- Example data set: Images of faces
  - "Eigenface" approach
    [Turk & Pentland], [Sirovich & Kirby]
- Each face **a** is …
  - 256 x 256 values (luminance at location)
  - **a** in $\Re^{256 \times 256}$ (view as 1D vector)
- Form $A = [\, \mathbf{a}_1, \ldots, \mathbf{a}_m \,]$
- Compute $\Sigma = \mathbf{A}\mathbf{A}^T$
- Problem: Σ is 64K $\times$ 64K … HUGE!!!

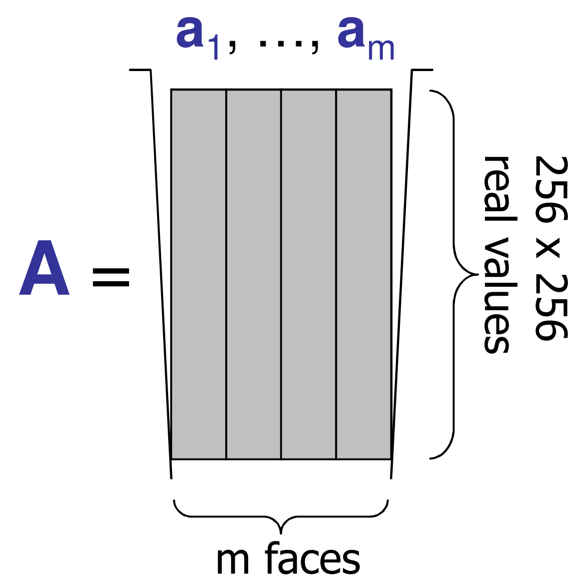


Computational Complexity

- Suppose m instances, each of size d
  - Eigenfaces: $m=500$ faces, each of size $d=64K$
- Given $d \times d$ covariance matrix Σ, can compute
  - all d eigenvectors/eigenvalues in $O(d^3)$
  - first k eigenvectors/eigenvalues in $O(k\ d^2)$
- But if $d=64K$, EXPENSIVE!

A Work-around …

- Note that $m \ll 64K$
- Use $L = A^T A$ instead of $\Sigma = AA^T$
- If $\mathbf{v}$ is eigenvector of L
  then $A\mathbf{v}$ is eigenvector of Σ
  (… same eigenvalue)



Proof:

$$L\ \mathbf{v} = \gamma\ \mathbf{v}$$

$$A^T A\ \mathbf{v} = \gamma\ \mathbf{v}$$

$$A\ (A^T A\ \mathbf{v}) = A(\gamma\ \mathbf{v}) = \gamma\ A\mathbf{v}$$

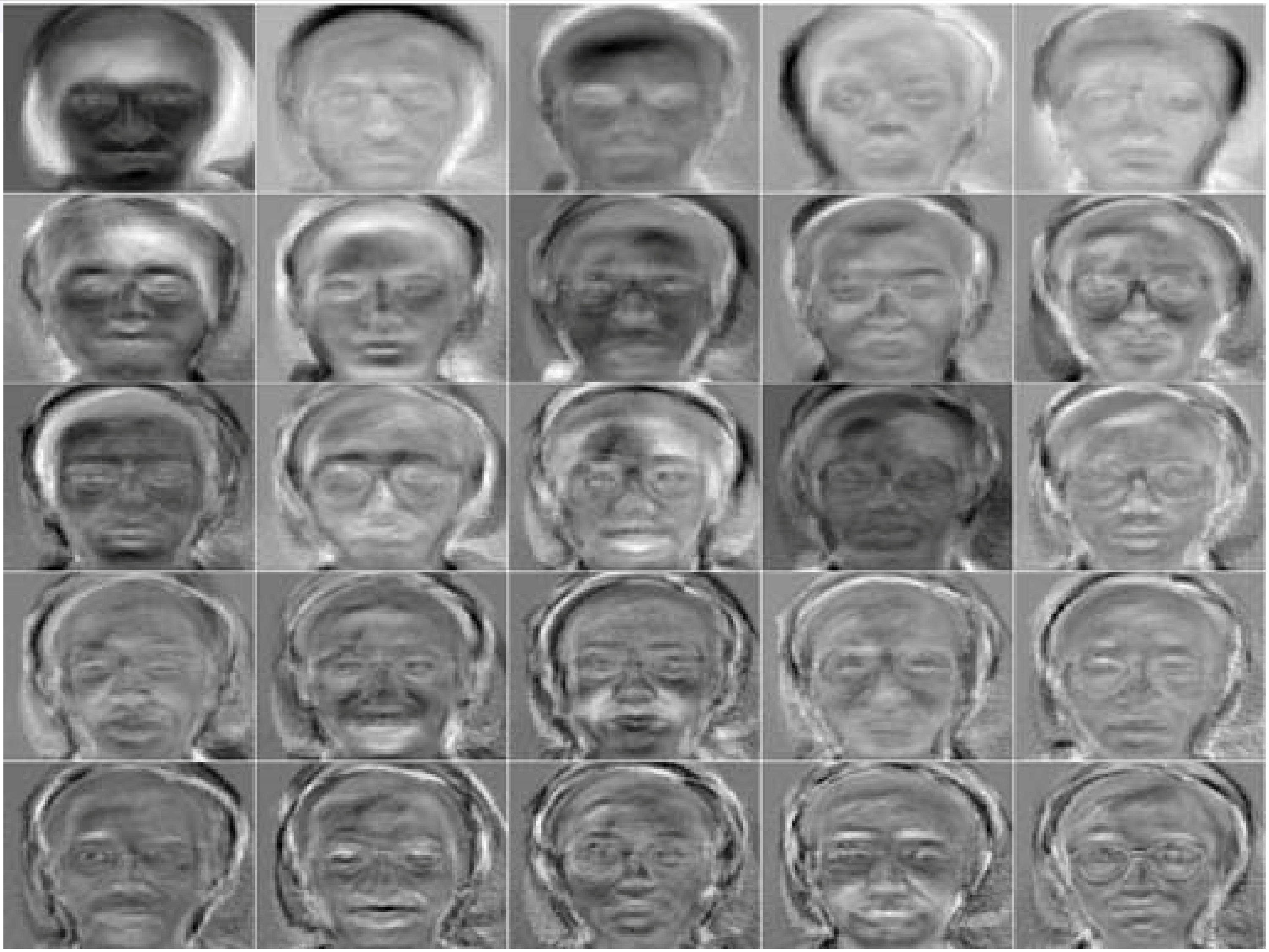
$$(A\ A^T)A\ \mathbf{v} = \gamma\ (A\mathbf{v})$$

$$\Sigma\ (A\mathbf{v}) = \gamma\ (A\mathbf{v})$$

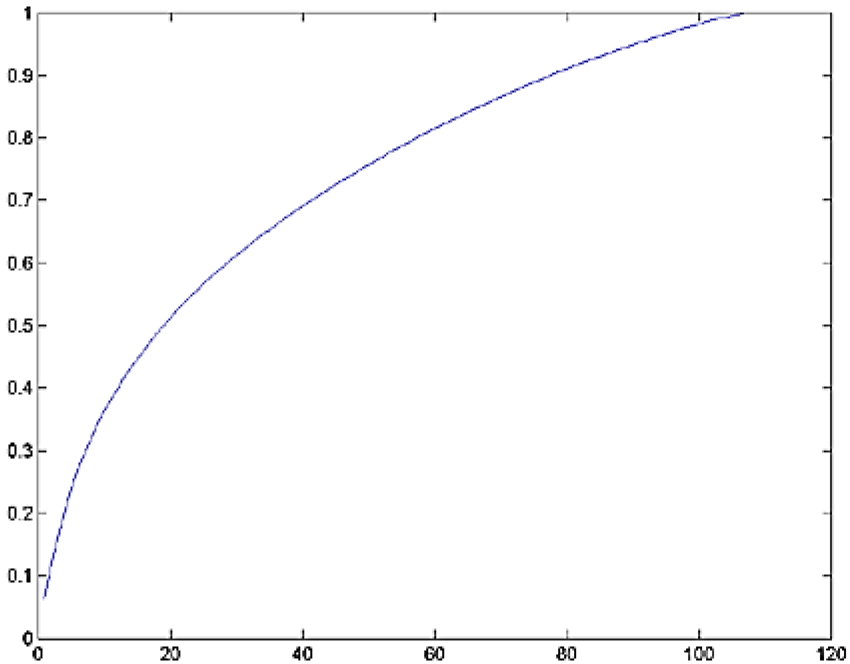
Eigenfaces

http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Principle Components



How Much Variance is Captured?



- Percentage of variance captured
  - k=10: 0.363
  - k=25: 0.566

Reconstructing…

- Takes 7 or 8-ish to get ≈this person…
- … faster if train with…
  - only people w/out glasses
  - same lighting conditions

Shortcomings

- Requires carefully controlled data:
  - All faces centered in frame
  - Same size
  - Some sensitivity to angle
- Alternative:
  - "Learn" one set of PCA vectors for each angle
  - Use the one with lowest error

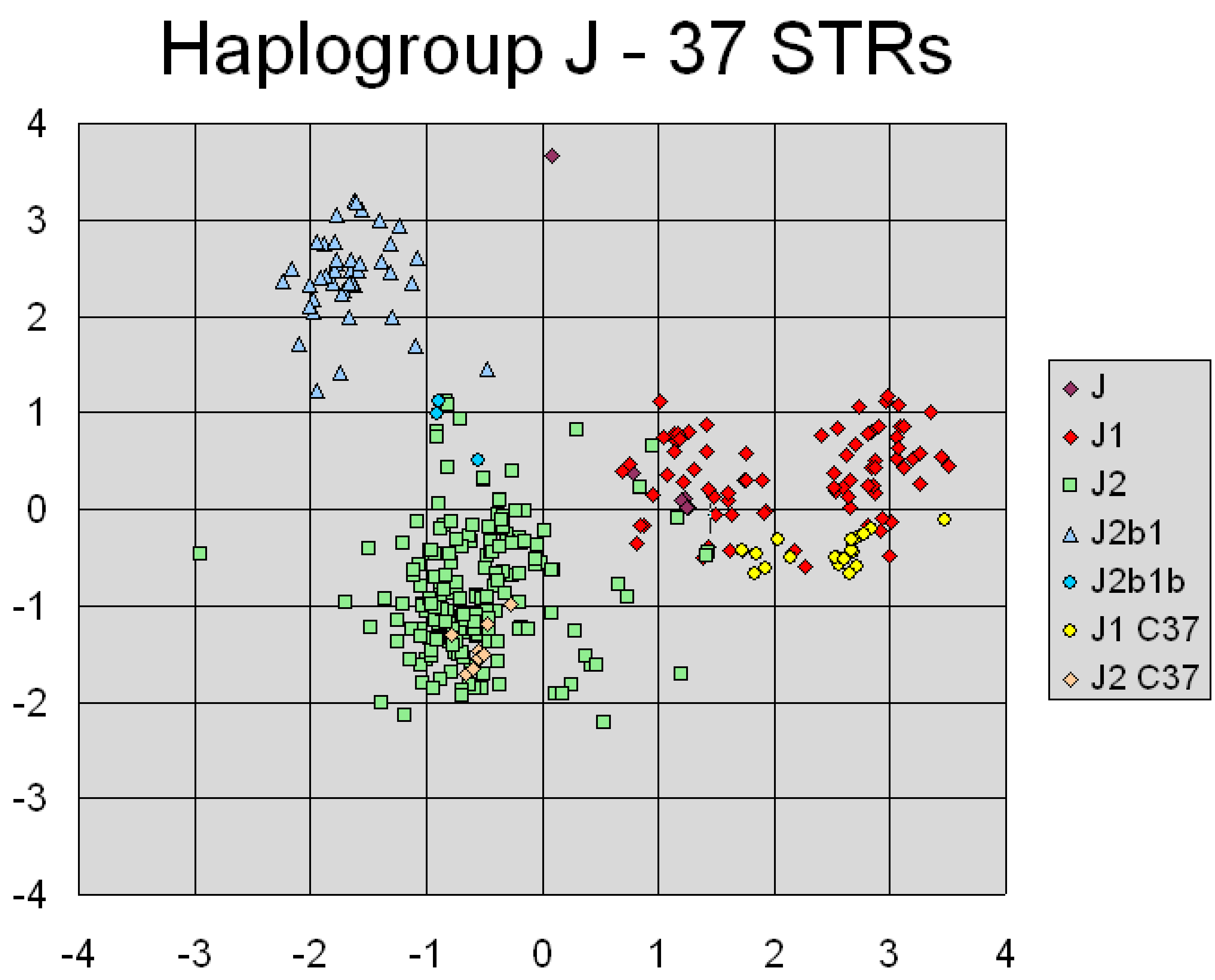
- Method is completely knowledge-free
  - (sometimes this is good!)
  - Doesn't know that faces are wrapped around 3D objects (heads)
  - Makes no effort to preserve class distinctions

Now What? Why run PCA??

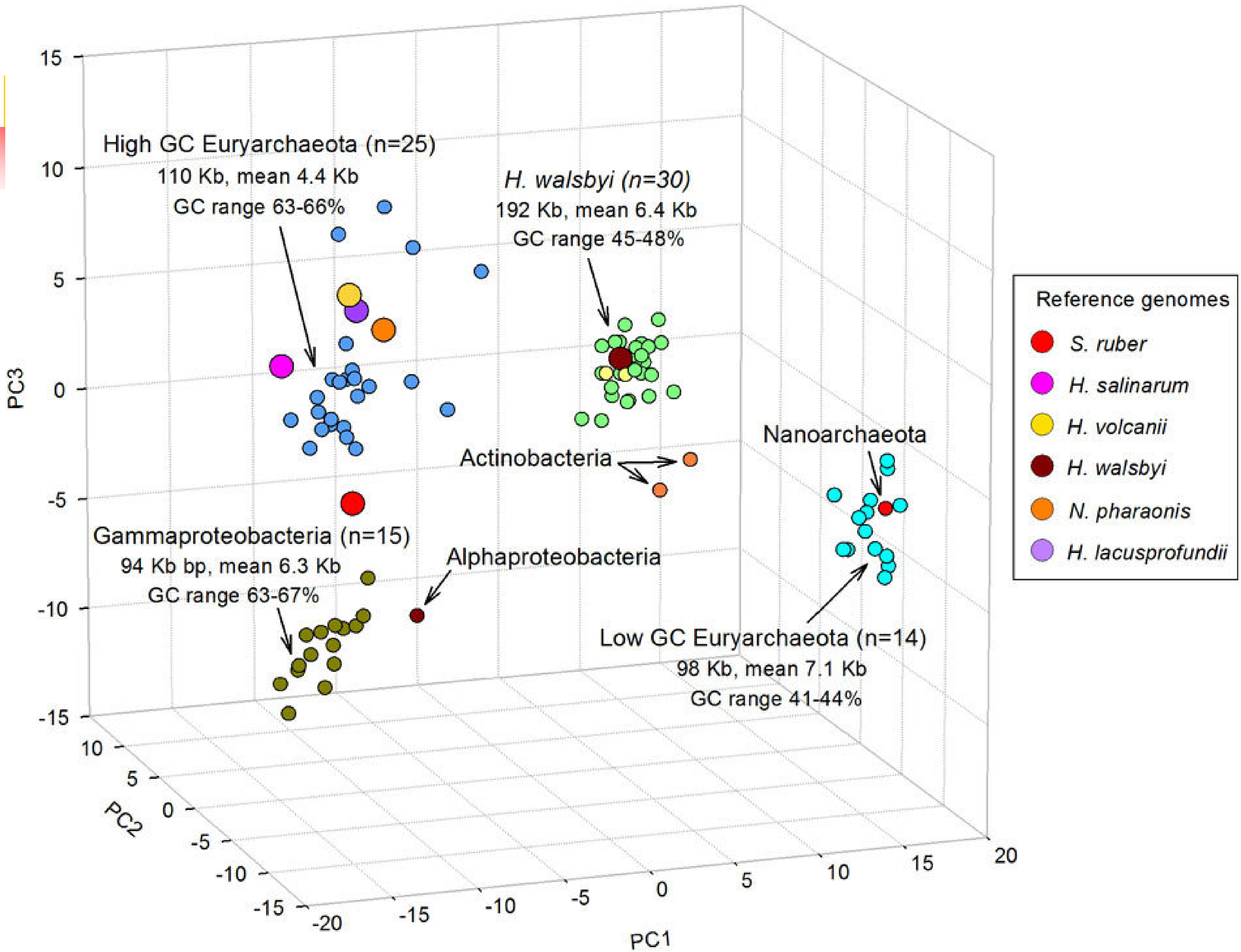
After acquiring eigen-values/vectors…

- Data compression (lossy):
  Compress d-dimension image into k reals:
  - Given new image $\mathbf{x}$, let $\mathbf{y} = \mathbf{x} - \bar{\mathbf{x}}$,
    use $[\mathbf{u}_1^T\mathbf{y}, \ldots, \mathbf{u}_k^T\mathbf{y}]$
  - To recover: $\mathbf{x}' = \bar{\mathbf{x}} + \sum_i (\mathbf{u}_i^T\mathbf{y})\, \mathbf{u}_i$

  Why? Reduce memory needed to store data
- Anomaly Detection
  - Consider error of original $\mathbf{x}$ vs $\bar{\mathbf{x}} + \sum_i^k (\mathbf{u}_i^T\mathbf{y})\, \mathbf{u}_i$
  - Large error suggests $\mathbf{x}$ is not from original distribution…
- Visualization
  - $k=2$ or $k=3$
  - Can have "labels" as red vs blue



http://upload.wikimedia.org/wikipedia/commons/6/69/PCA_of_Haplogroup_J_using_37_STRs.png



http://www.nature.com/srep/2011/111031/srep00135/images/srep00135-f5.jpg

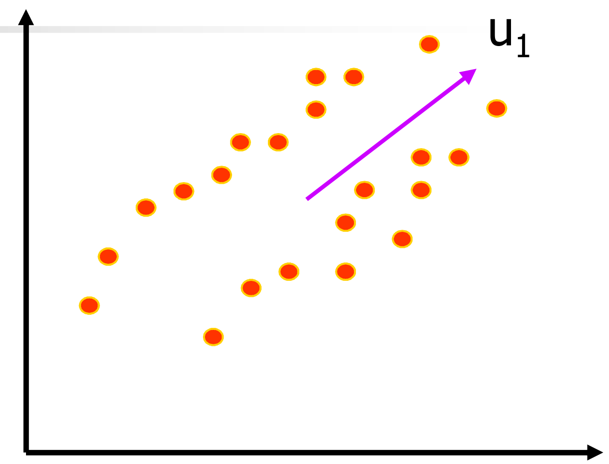
Now What? Why run PCA??

After acquiring eigen-values/vectors…

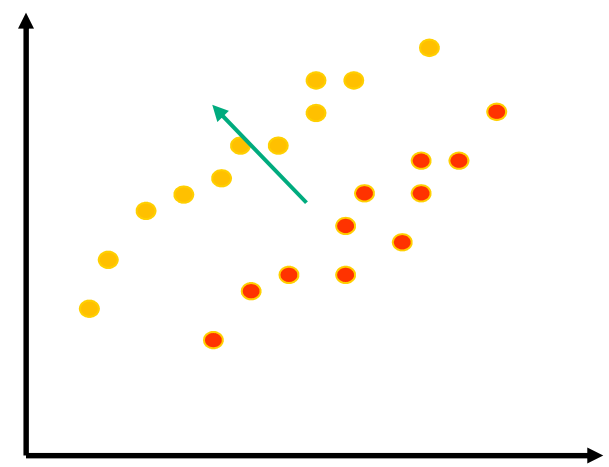
- Preprocessing for supervised learning:
  - Given labeled datasample $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^M]$, $Y=[y^1, \ldots, y^M]$
  - Reduce each $\mathbf{x}^i$ to k reals $\mathbf{r}^i = [r_{i,1}, \ldots, r_{i,k}]^T$
  - Run learner on $R = [\mathbf{r}^1, \ldots, \mathbf{r}^M]$, $Y=[y^1, \ldots, y^M]$

  Why??
  - Speed-up learning
  - Consider other learners …
  - ? to reduce chance of overfitting
  - Note: PCA is throwing away information.
    … perhaps information that is useful wrt classification
    - ⇒ A Ng recommends NOT doing this!
    - … better instead to use regularization

Problematic Data Set



- PCA maximizes variance, (*independent of class*)
  - $\Rightarrow$ magenta
- Fisher Linear Discriminant (FLD) attempts to separate classes
  - $\Rightarrow$ green line



PCA Conclusions

- PCA
  - finds orthonormal basis for data
  - Sorts dimensions in order of "importance"
  - Discard low significance dimensions
- Uses:
  - Get compact description
  - Ignore noise
  - Improve classification (hopefully)
- Not magic:
  - Doesn't know class labels
  - Can only capture linear variations
- One of many tricks to reduce dimensionality!