Dimensionality Reduction

Feature Selection vs. Dimensionality Reduction

- Feature Selection (last time)
  - Select a subset of features.
  - When classifying novel patterns, only a small number of features need to be computed (i.e., faster classification).
  - The measurement units (length, weight, etc.) of the features are preserved.

- Dimensionality Reduction (this time)
  - Transform features into a smaller set.
  - When classifying novel patterns, all features need to be computed.
  - The measurement units (length, weight, etc.) of the features are lost.

How Can We Visualize High Dimensional Data?

- E.g., 53 blood and urine tests for 65 patients

	H-WBC	H-RBC	H-Hgb	H-Hct	H-MCV	H-MCH	H-MCHC
A1	8.0000	4.8200	14.1000	41.0000	85.0000	29.0000	34.0000
A2	7.3000	5.0200	14.7000	43.0000	86.0000	29.0000	34.0000
A3	4.3000	4.4800	14.1000	41.0000	91.0000	32.0000	35.0000
A4	7.5000	4.4700	14.9000	45.0000	101.0000	33.0000	33.0000
A5	7.3000	5.5200	15.4000	46.0000	84.0000	28.0000	33.0000
A6	6.9000	4.8600	16.0000	47.0000	97.0000	33.0000	34.0000
A7	7.8000	4.6800	14.7000	43.0000	92.0000	31.0000	34.0000
A8	8.6000	4.8200	15.8000	42.0000	88.0000	33.0000	37.0000
A9	5.1000	4.7100	14.0000	43.0000	92.0000	30.0000	32.0000

Instances

Features

Difficult to see the correlations between the features...

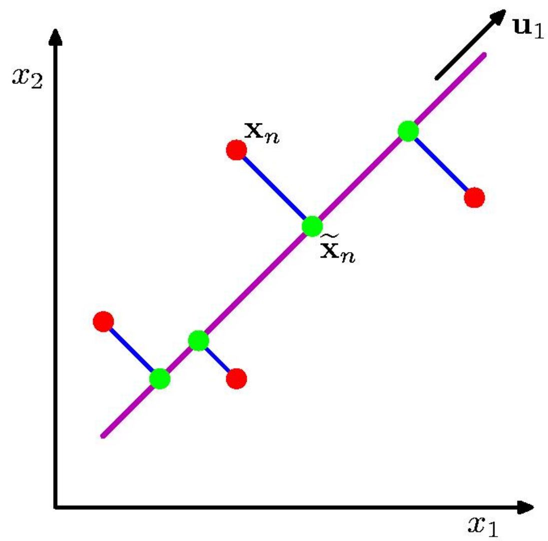
Data Visualization

- Is there a representation better than the raw features?
  - Is it really necessary to show all the 53 dimensions?
  - … what if there are strong correlations between the features?

Could we find the *smallest* subspace of the 53-D space that keeps the *most information* about the original data?

One solution: **Principal Component Analysis**

Principle Component Analysis



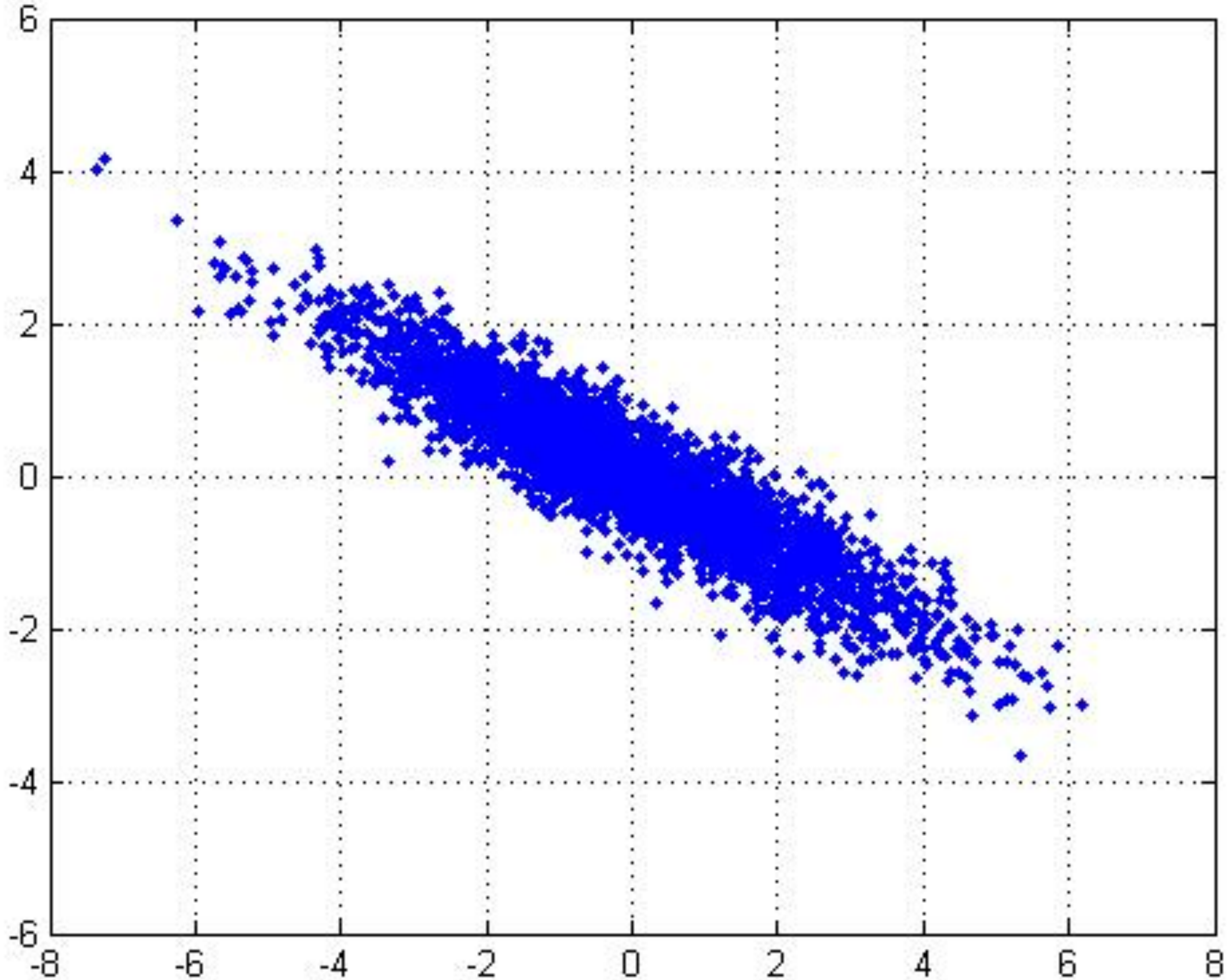
Orthogonal projection of data onto lower-dimension linear space that...

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

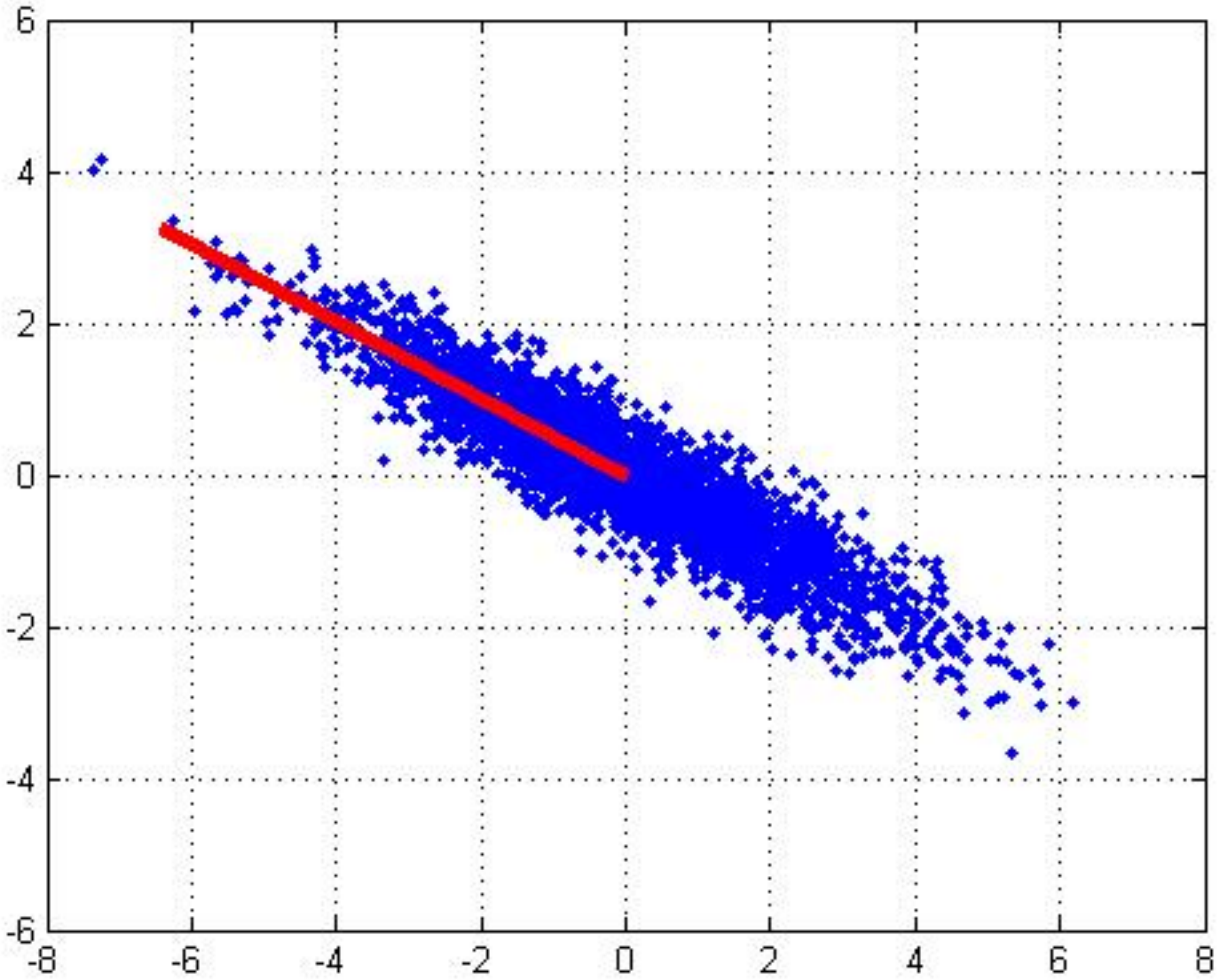
The Principal Components

- **Vectors** originating from the center of mass
- Principal component #1 points in the direction of the **largest variance**
- Each subsequent principal component…
  - is **orthogonal** to the previous ones, and
  - points in the directions of the **largest variance of the residual subspace**

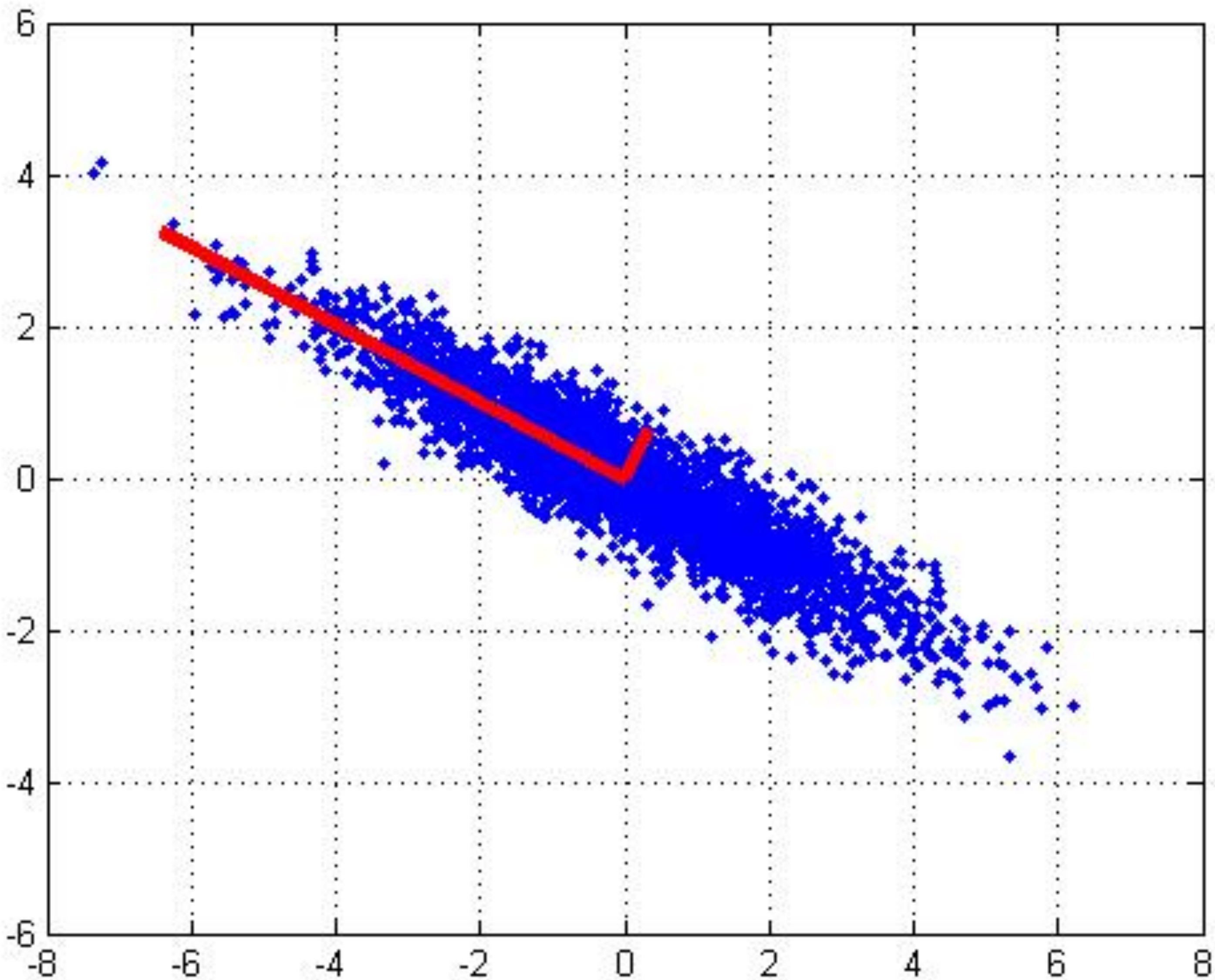
2D Gaussian Dataset



1^{st} PCA axis



2^{nd} PCA axis



PCA Algorithm

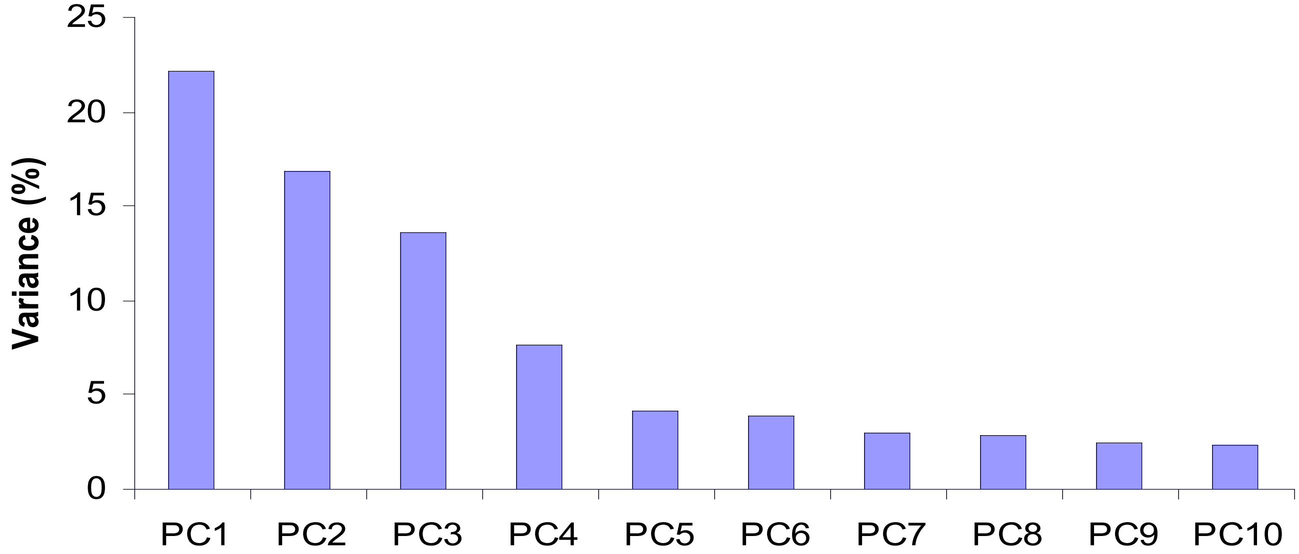
- Given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, compute covariance matrix Σ
  - X is the n x d data matrix
  - Compute data mean (average over all rows of X)
  - Subtract mean from each row of X (centering the data)
  - Compute covariance matrix $\Sigma = X^T X$ (Σ is d x d)

- **PCA** basis vectors are given by the eigenvectors of Σ
  - Q, Λ = numpy.linalg.eig(Σ)
  - $\{\mathbf{q}_i, \lambda_i\}_{i=1..n}$ are the eigenvectors/eigenvalues of Σ
    … $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$

- Larger eigenvalue $\Rightarrow$ more important eigenvectors

Dimensionality Reduction

Can *ignore* the components of lesser significance



You do lose some information, but if the eigenvalues are small, you don't lose much

– choose only the first *k* eigenvectors, based on their eigenvalues
– final data set has only *k* dimensions

PCA

$$X = \begin{bmatrix} 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1 \ldots \\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \ldots \\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0 \ldots \\ \vdots \\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0 \ldots \end{bmatrix}$$

X has *d* columns

Q are the eigenvectors of Σ;
columns are ordered by importance!

Q is *d* x *d*

$$Q = \begin{bmatrix} 0.34 & 0.23 & -0.30 & -0.23 & \ldots \\ 0.04 & 0.13 & -0.40 & 0.21 & \ldots \\ -0.64 & 0.93 & 0.61 & 0.28 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ -0.20 & -0.83 & 0.78 & -0.93 & \ldots \end{bmatrix}$$

Slide by Eric Eaton

PCA

$$X = \begin{bmatrix} 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1 \dots \\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \dots \\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0 \dots \\ \vdots \\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0 \dots \end{bmatrix}$$

Each row of Q corresponds to a feature; keep only first *k* columns of Q

$$Q = \begin{bmatrix} 0.34 & 0.23 & -0.30 & -0.23 & \dots \\ 0.04 & 0.13 & -0.40 & 0.21 & \dots \\ -0.64 & 0.93 & 0.61 & 0.28 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ -0.20 & -0.83 & 0.78 & -0.93 & \dots \end{bmatrix}$$

Slide by Eric Eaton

PCA

- Each column of Q gives weights for a linear combination of the original features

$$Q = \begin{bmatrix} 0.34 & 0.23 & -0.30 & -0.23 & \cdots \\ 0.04 & 0.13 & -0.40 & 0.21 & \cdots \\ -0.64 & 0.93 & 0.61 & 0.28 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ -0.20 & -0.83 & 0.78 & -0.93 & \cdots \end{bmatrix}$$

= 0.34 feature1 + 0.04 feature2 – 0.64 feature3 + ...

PCA

- We can apply these formulas to get the new representation for each instance **x**

$$X = \begin{bmatrix} 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1 \ldots \\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \ldots \\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0 \ldots \\ \vdots \\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0 \ldots \end{bmatrix} \begin{array}{l} \\ \\ \mathbf{x}_3 \\ \\ \\ \end{array} \qquad \hat{Q} = \begin{bmatrix} 0.34 & 0.23 \\ 0.04 & 0.13 \\ -0.64 & 0.93 \\ \vdots & \vdots \\ -0.20 & -0.83 \end{bmatrix}$$

- The new 2D representation for $\mathbf{x}_3$ is given by:

$$\hat{x}_{31} = 0.34(0) + 0.04(0) - 0.64(1) + \ldots$$

$$\hat{x}_{32} = 0.23(0) + 0.13(0) + 0.93(1) + \ldots$$

- The re-projected data matrix is given by $\hat{X} = X\hat{Q}$

PCA Example

Data

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Center Data

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

Covariance Matrix

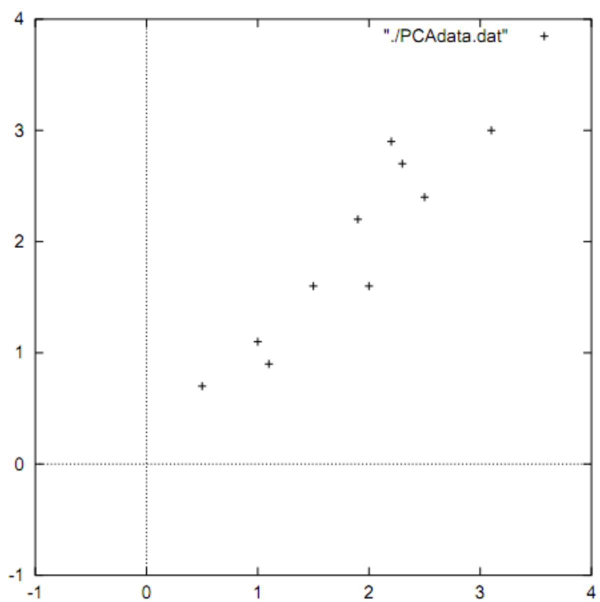
0.61655	0.61544
0.61544	0.71655

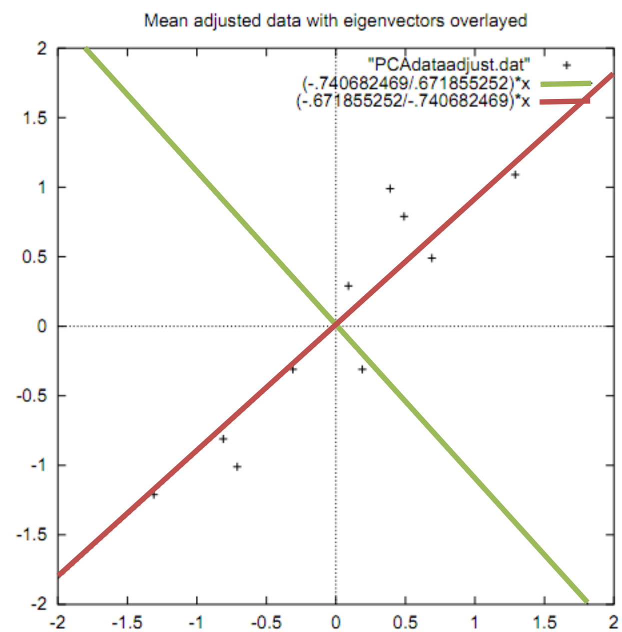
Eigenvectors

-0.73518	-0.67787
0.67787	-0.73518

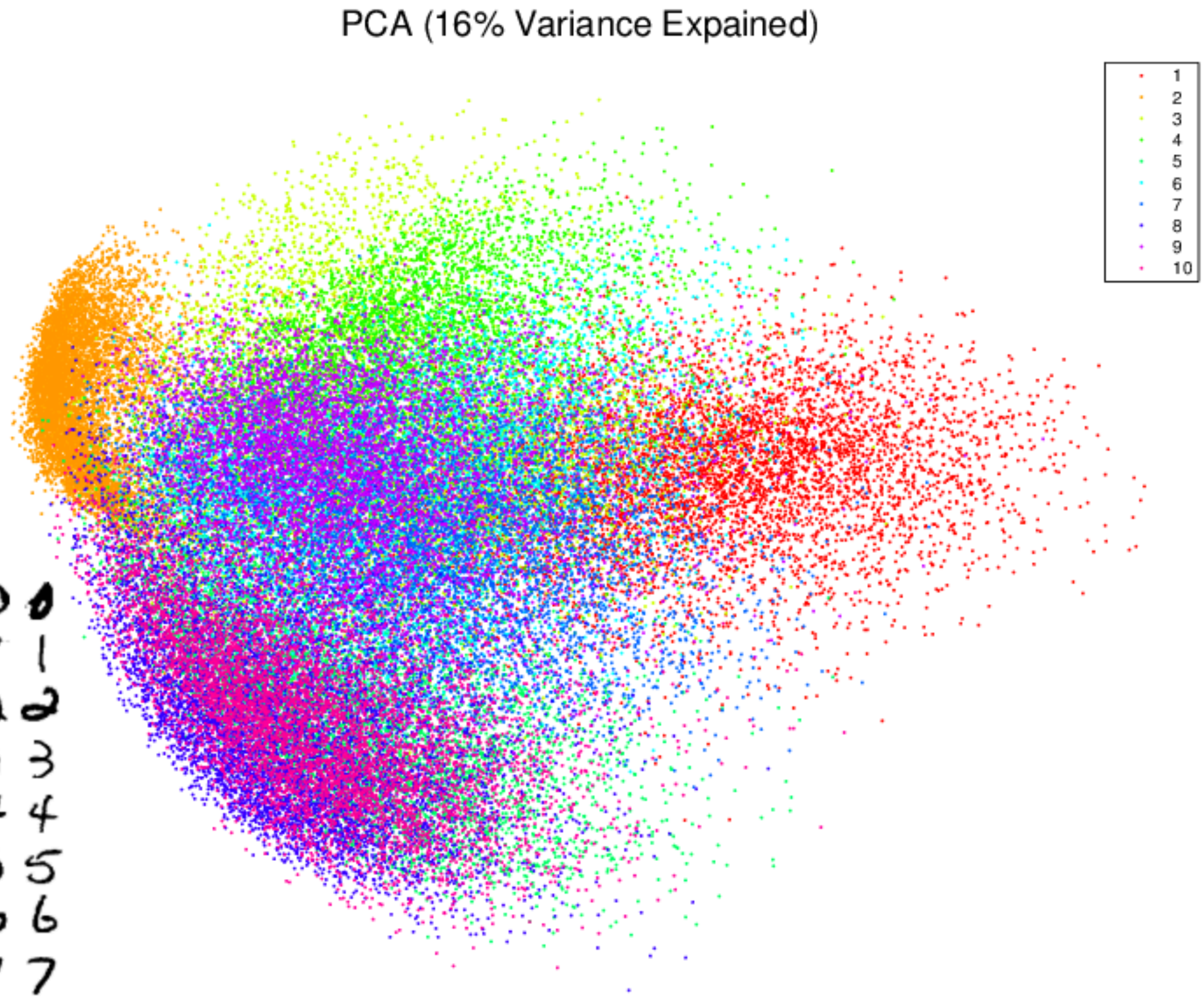
Eigenvalues

0.04908	0
0	1.28403





PCA Visualization of MNIST Digits



Challenge: Facial Recognition

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

⇒ Can't just use the given 256 x 256 pixels

PCA applications - Eigenfaces

- Eigenfaces are

  the eigenvectors of the covariance matrix of the probability distribution of the vector space of human faces

- Eigenfaces are the 'standardized face ingredients' derived from the statistical analysis of many pictures of human faces

- A human face may be considered to be a combination of these standard face ingredients

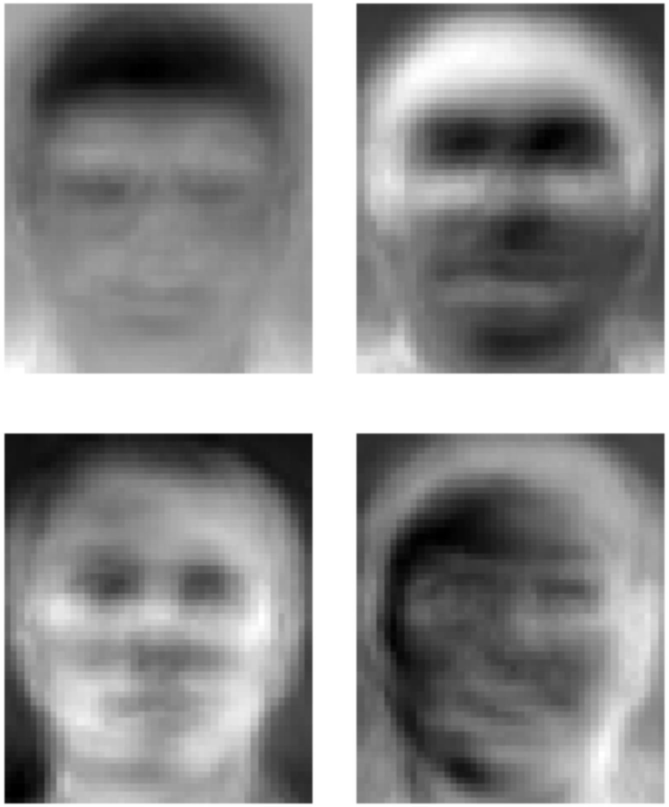
PCA applications -Eigenfaces

To generate a **set of eigenfaces:**

1. Large set of digitized images of human faces is taken under the same lighting conditions.
2. The images are normalized to line up the eyes and mouths.
3. The eigenvectors of the covariance matrix of the statistical distribution of face image vectors are then extracted.
4. These eigenvectors are called eigenfaces.

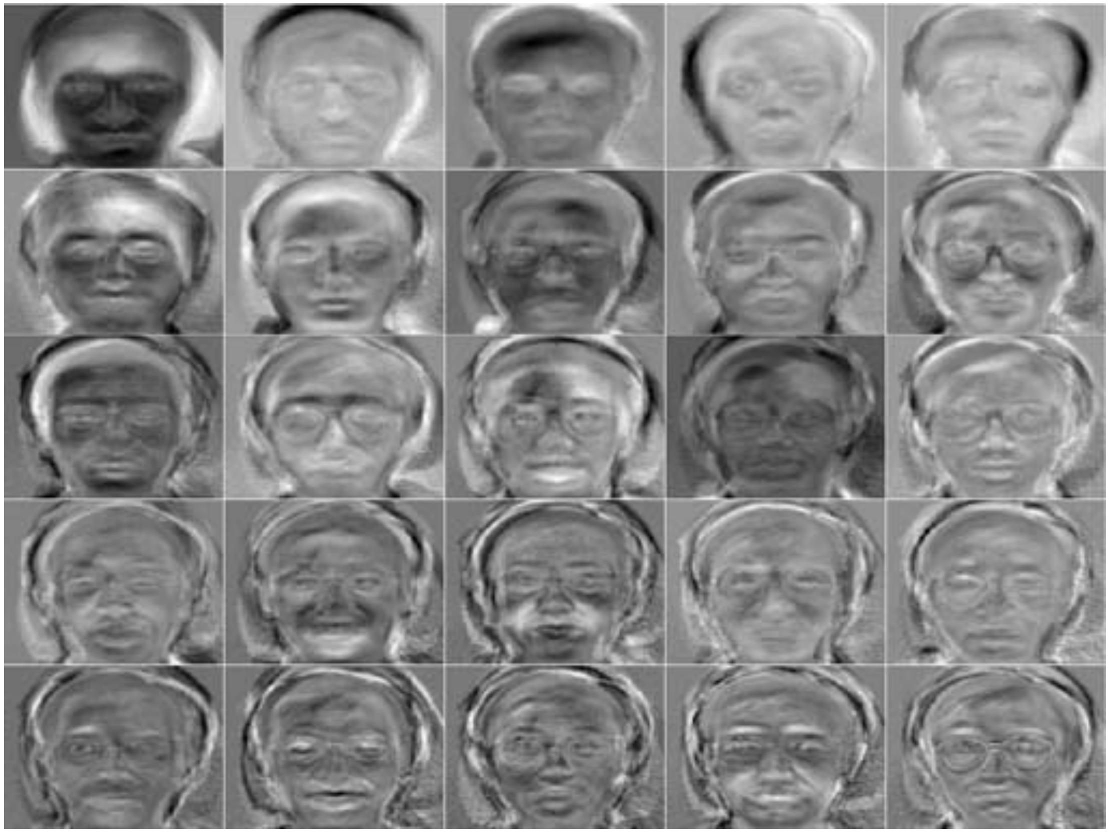
PCA applications -Eigenfaces

- the principal eigenface looks like a bland androgynous average human face



http://en.wikipedia.org/wiki/Image:Eigenfaces.png

Eigenfaces



Eigenfaces – Face Recognition

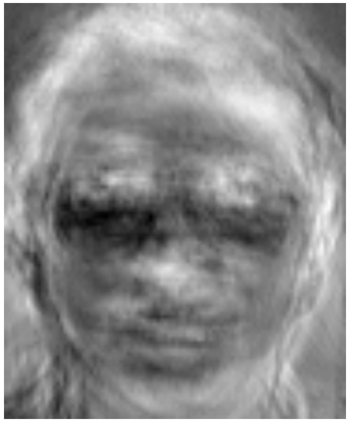
- When properly weighted, eigenfaces can be summed together to create an approximate gray-scale rendering of a human face.
- Remarkably few eigenvector terms are needed to give a fair likeness of most people's faces
- Hence eigenfaces provide a means of applying data compression to faces for identification purposes.
- Similarly, Expert Object Recognition in Video

Eigenfaces

- <u>Experiment and Results</u>

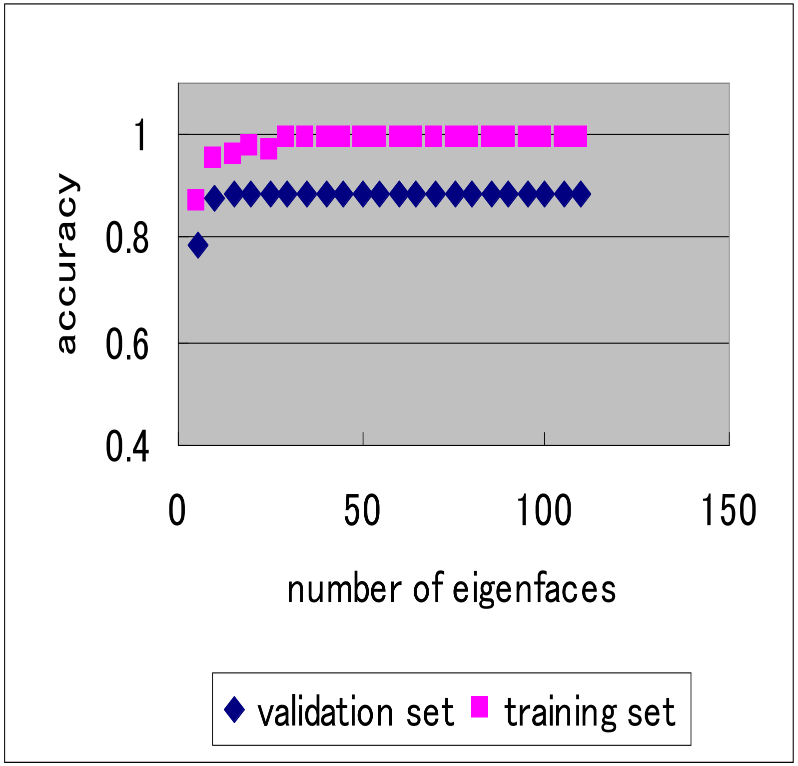
  Data used here are from the ORL database of faces.
  Facial images of 16 people each with 10 views are used.
  - Training set contains 16×7 images.
    Test set contains 16×3 images.

First three eigenfaces :



Classification Using Nearest Neighbor

- Save average coefficients for each person. Classify new face as the person with the closest average.
- Recognition accuracy increases with number of eigenfaces until ~15.



Best recognition rates

Training set 99%

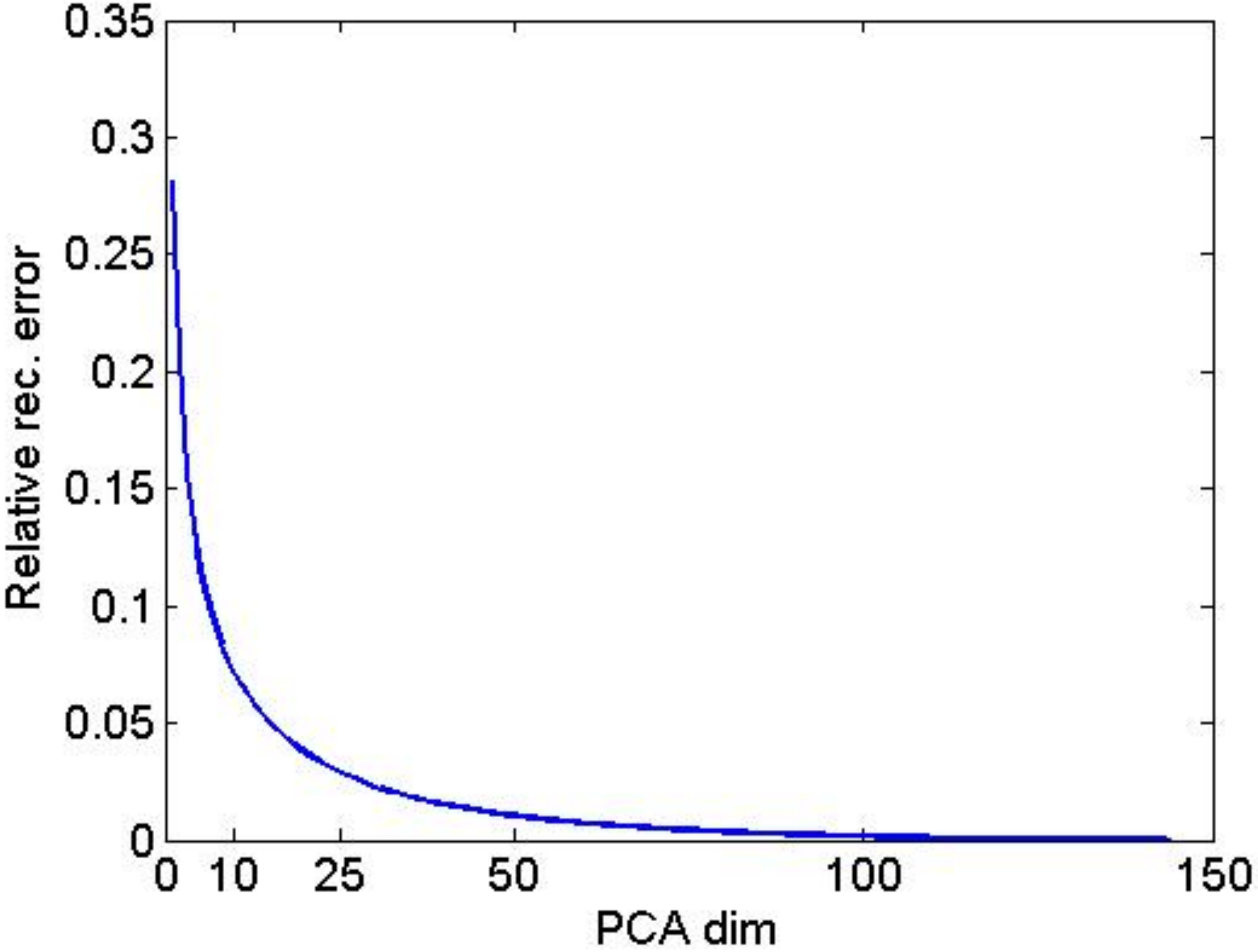
Test set 89%

Image Compression

Original Image

- Divide the original 372x492 image into patches:
    - Each patch is an instance that contains 12x12 pixels on a grid
- View each as a 144-D vector

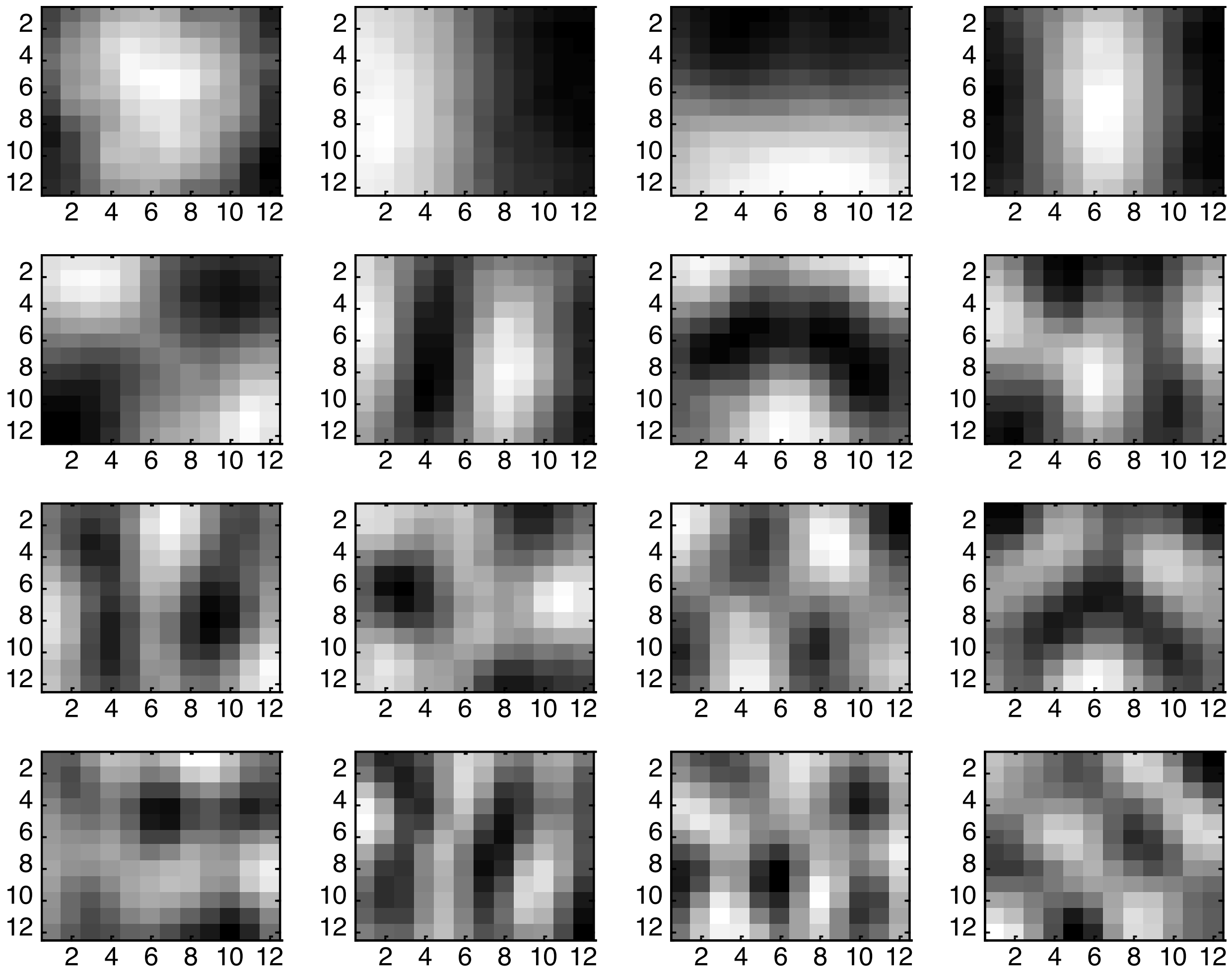
L_2 error and PCA dim



PCA compression: 144D → 60D

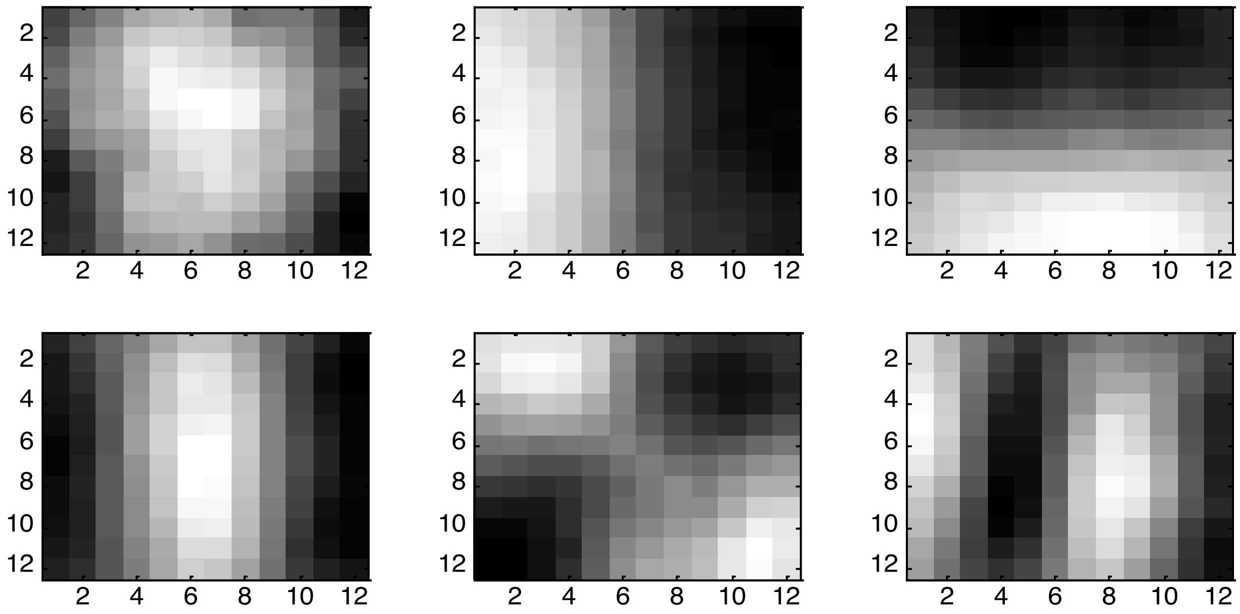
PCA compression: 144D → 16D

16 most important eigenvectors



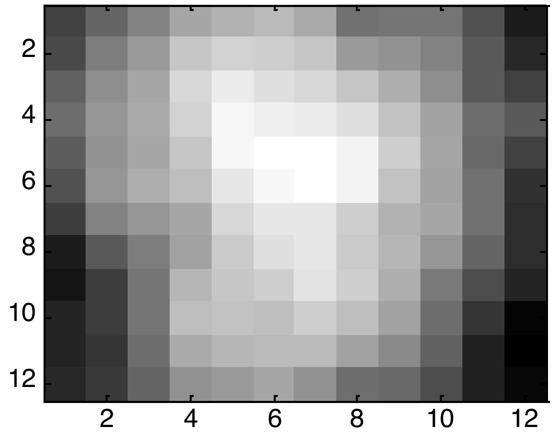
PCA compression: 144D → 6D

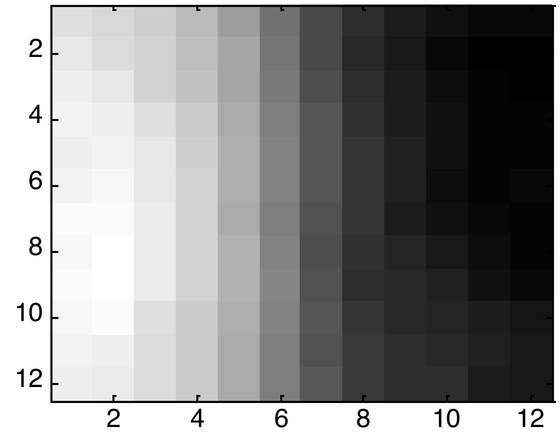
6 most important eigenvectors

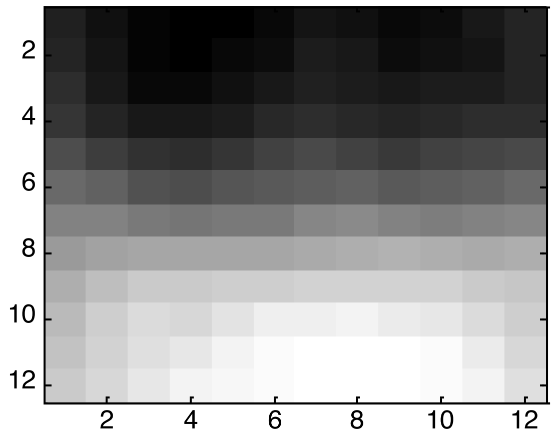


PCA compression: 144D → 3D

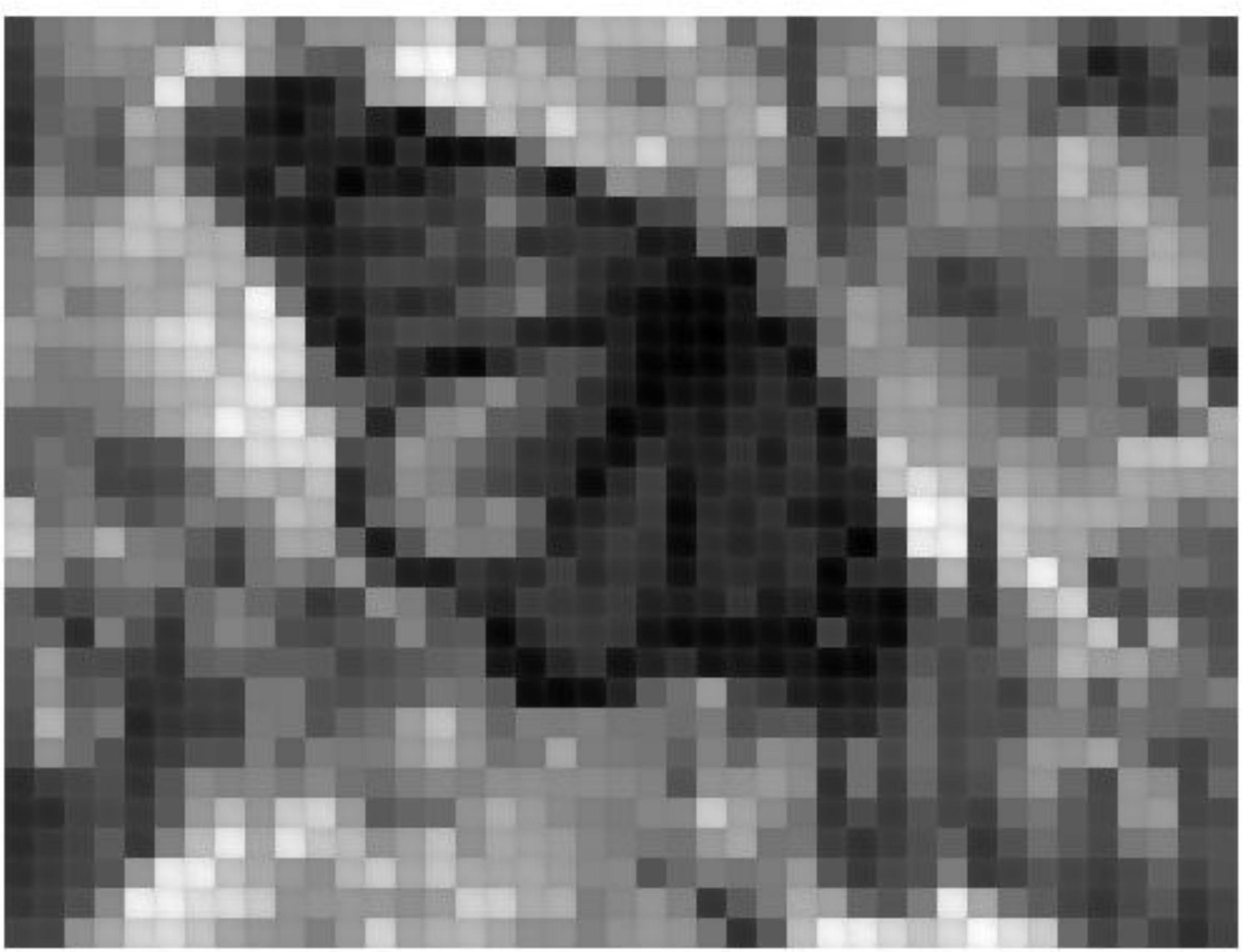
3 most important eigenvectors



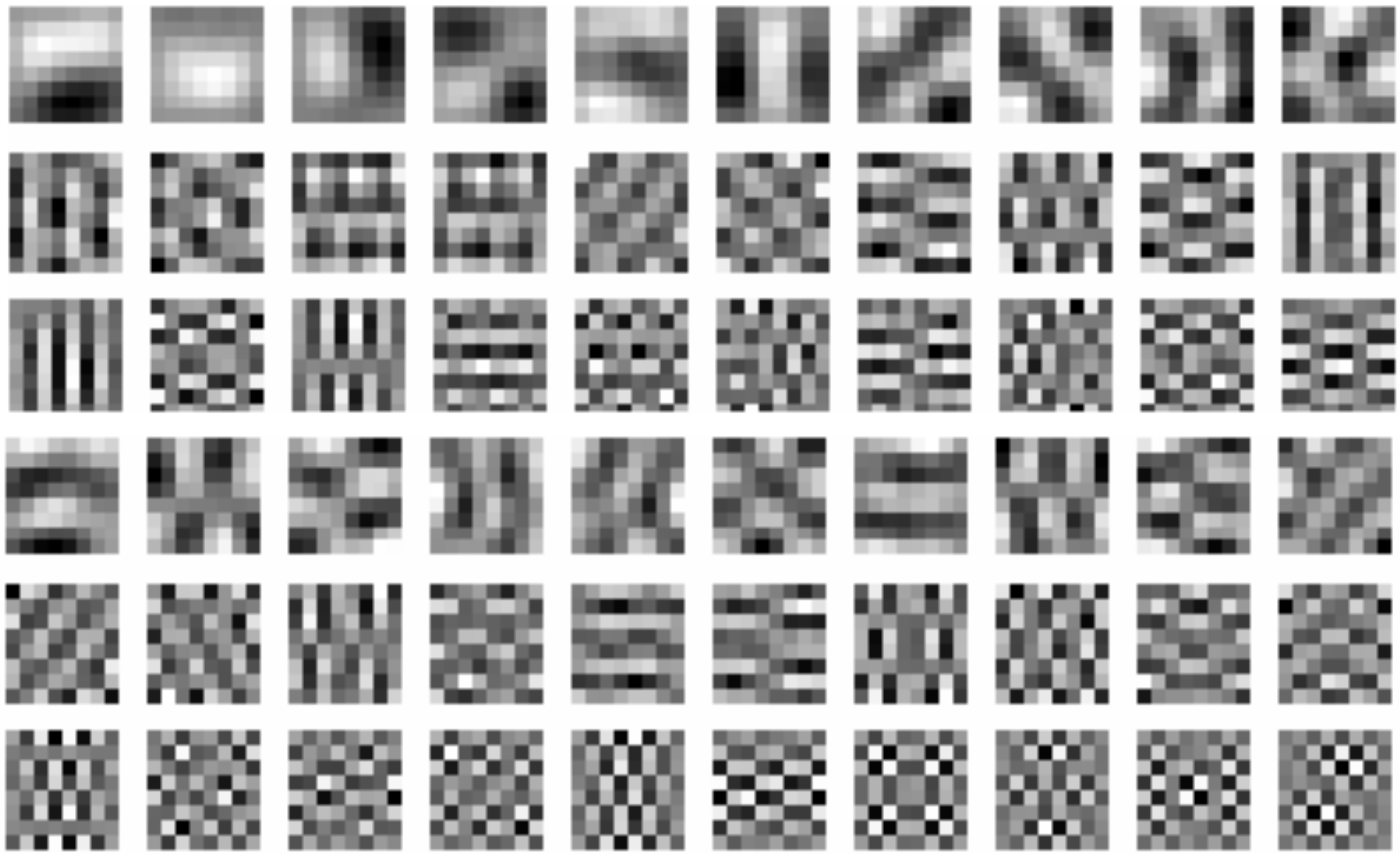




PCA compression: 144D → 1D

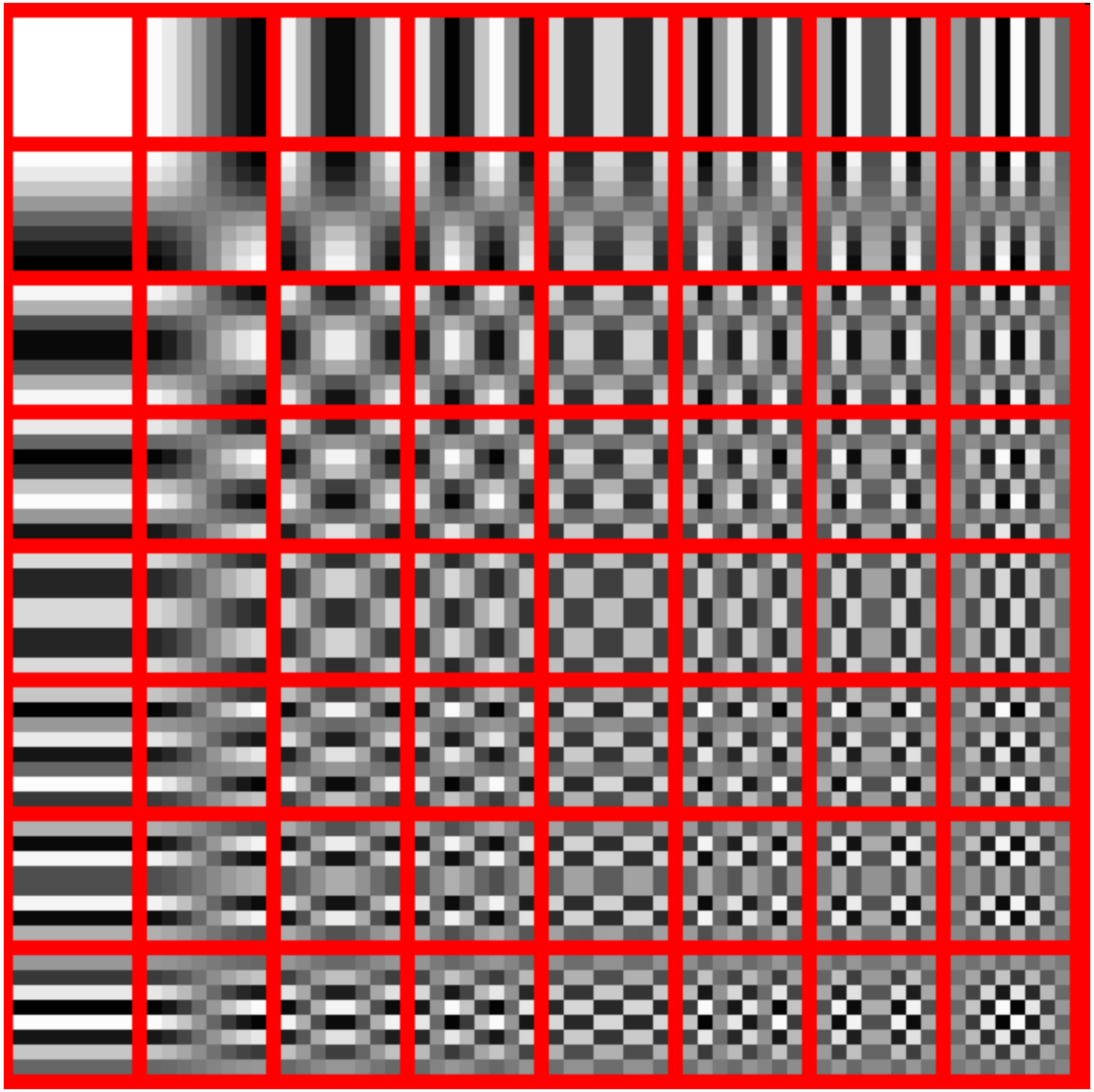


60 most important eigenvectors



Looks like the discrete cosine bases of JPG!...

2D Discrete Cosine Basis



http://en.wikipedia.org/wiki/Discrete_cosine_transform