CS373 Data Mining and Machine Learning

Lecture 12

Jean Honorio
Purdue University

Except for the first two and last two slides, the presentation was taken from
http://www.cs.cmu.edu/~bapoczos/Classes/ML10715_2015Fall/

Linear Algebra: Eigen Decomposition

- Symmetric matrix $\Sigma = U \Lambda U^T$
- Orthonormal matrix

$$U^T U = I$$

- Diagonal matrix Λ of eigenvalues

- Eigenvectors, columns of U

- Python:

```
import numpy as np
import numpy.linalg as la

Sigma = np.array([[ 5.2,  3.3,  -2],
        [ 3.3,  8.1,  4],
        [ -2,  4,  6.5]])
lam, U = la.eig(Sigma)
```

```
>>> lam
array([  0.38,   7.7,   11.71])
>>> U
array([[-0.61,  0.75,  0.25],
       [ 0.55,  0.18,  0.81],
       [-0.56, -0.64,  0.53]])

>>> np.dot(U.T,U)
array([[  1.,  0.,  0.],
       [  0.,  1.,  0.],
       [  0.,  0.,  1.]])

>>> np.dot(U,np.dot(np.diag(lam),U.T))
array([[ 5.2,  3.3, -2. ],
       [ 3.3,  8.1,  4. ],
       [-2. ,  4. ,  6.5]])
```

Linear Algebra: Singular Value Decomposition

- Matrix $X = USV^T$
- Orthonormal matrices

$$U^T U = I \quad , \quad V^T V = I$$

- Diagonal matrix S of singular values
- Python:

```
import numpy as np
import numpy.linalg as la

X = np.array([[ 0.6,  -0.7],
        [ 2.5,  1.9],
        [ -1.6,  -0.9],
        [ -2.8,  0.8]])
U, s, Vt = la.svd(X,False)
```

```
>>> U
array([[-0.09,  0.39],
       [-0.69, -0.54],
       [ 0.42,  0.2 ],
       [ 0.58, -0.72]])
>>> s
array([ 4.24,  2.13])
>>> Vt
array([[-0.96, -0.27],
       [ 0.27, -0.96]])

>>> np.dot(Vt,Vt.T)
array([[ 1.,  0.],
       [ 0.,  1.]])

# U*Diag(s)*Vt
>>> np.dot(U,np.dot(np.diag(s),Vt))
array([[ 0.6, -0.7],
       [ 2.5,  1.9],
       [-1.6, -0.9],
       [-2.8,  0.8]])
```

Motivation

PCA Applications

- Data Visualization
- Data Compression
- Noise Reduction

Data Visualization

Example:

- Given 53 blood and urine samples (features) from 65 people.

- How can we visualize the measurements?

Data Visualization

- Matrix format (65x53)

Instances

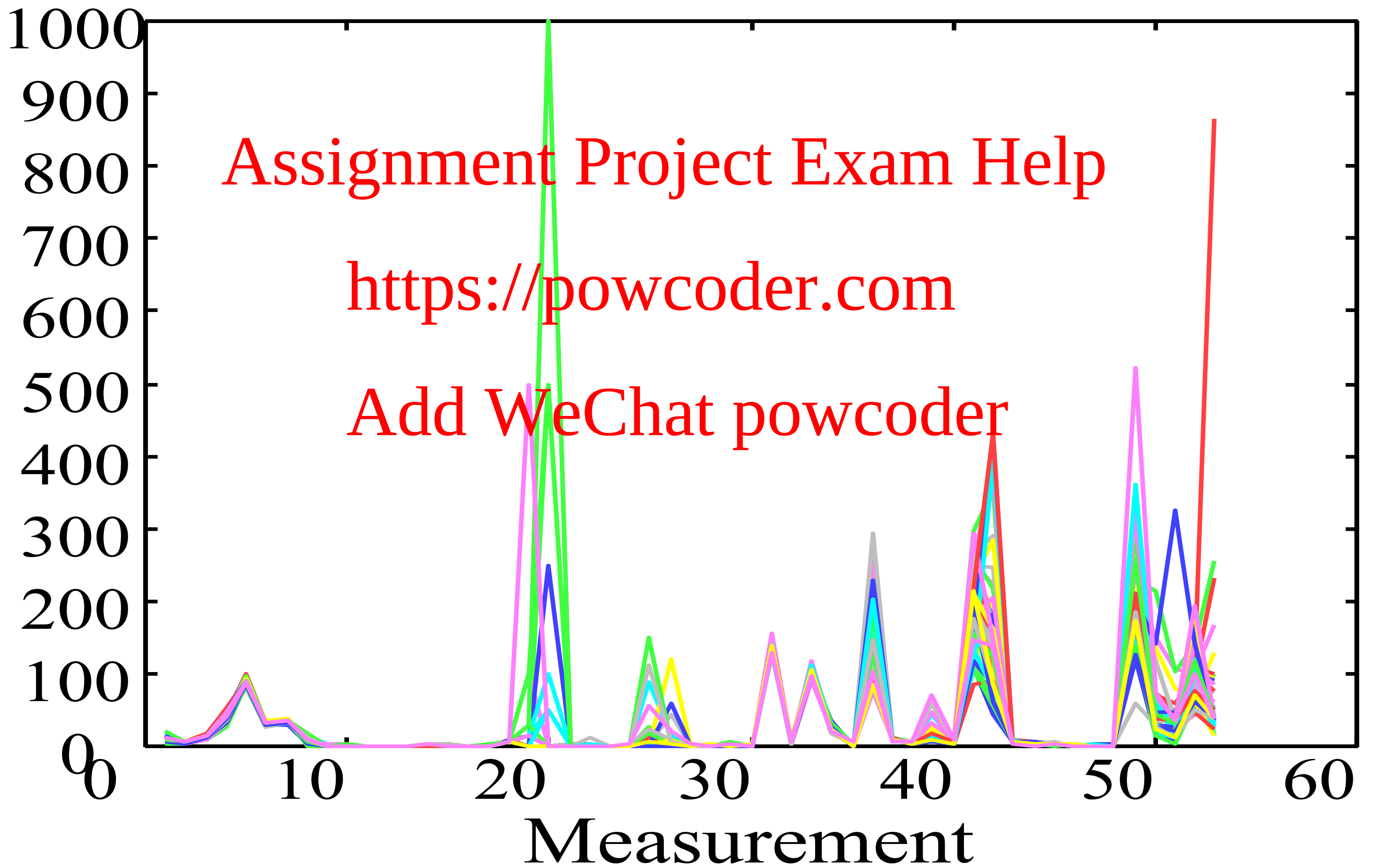
	H-WBC	H-RBC	H-Hgb	H-Hct	H-MCV	H-MCH	H-MCHC
A1	8.0000	4.8200	14.1000	41.0000	85.0000	29.0000	34.0000
A2	7.3000	5.0200	14.7000	43.0000	86.0000	29.0000	34.0000
A3	4.3000	4.4800	14.1000	41.0000	91.0000	32.0000	35.0000
A4	7.5000	4.4700	14.9000	45.0000	101.0000	33.0000	33.0000
A5	7.3000	5.5200	15.4000	46.0000	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

Features

Difficult to see the correlations between the features...

Data Visualization

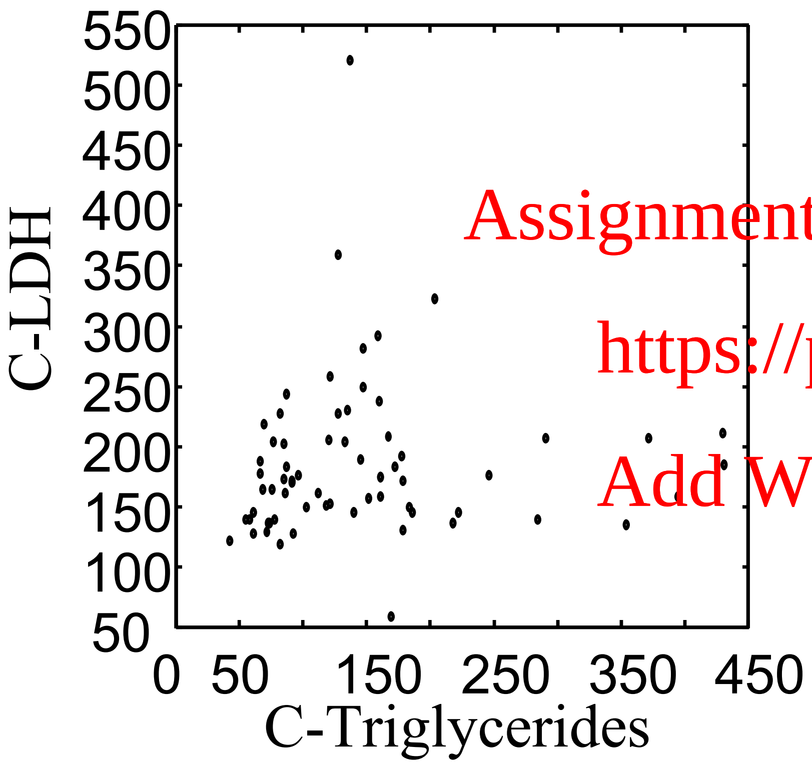
- Spectral format (65 curves, one for each person)



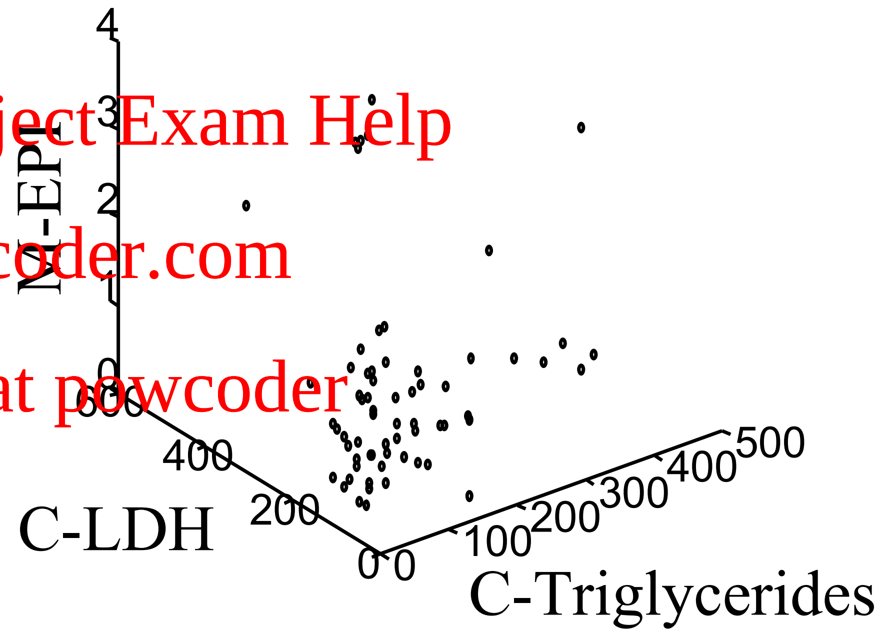
Difficult to compare the different patients...

Data Visualization

Bi-variate



Tri-variate



How can we visualize the other variables???

… difficult to see in 4 or higher dimensional spaces...

Data Visualization

- Is there a representation better than the coordinate axes?

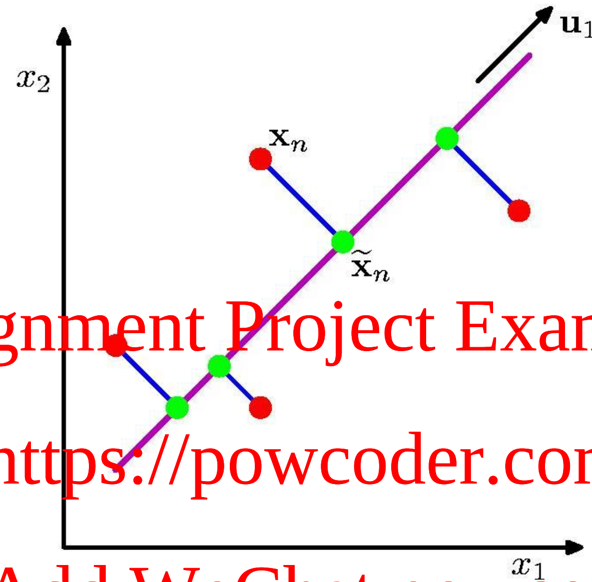
- Is it really necessary to show all the 53 dimensions?
  - … what if there are strong correlations between the features?

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

- A solution: **Principal Component Analysis**

PCA Algorithms

Principal Component Analysis



PCA:

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

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

Principal Component Analysis

Idea:

- Given data points in a d-dimensional space, project them into a lower dimensional space while preserving as much information as possible.
  - Find best planar approximation of 3D data
  - Find best 12-D approximation of 10^4-D data

- In particular, choose projection that

  minimizes *squared error*

  in reconstructing the original data.

Principal Component Analysis

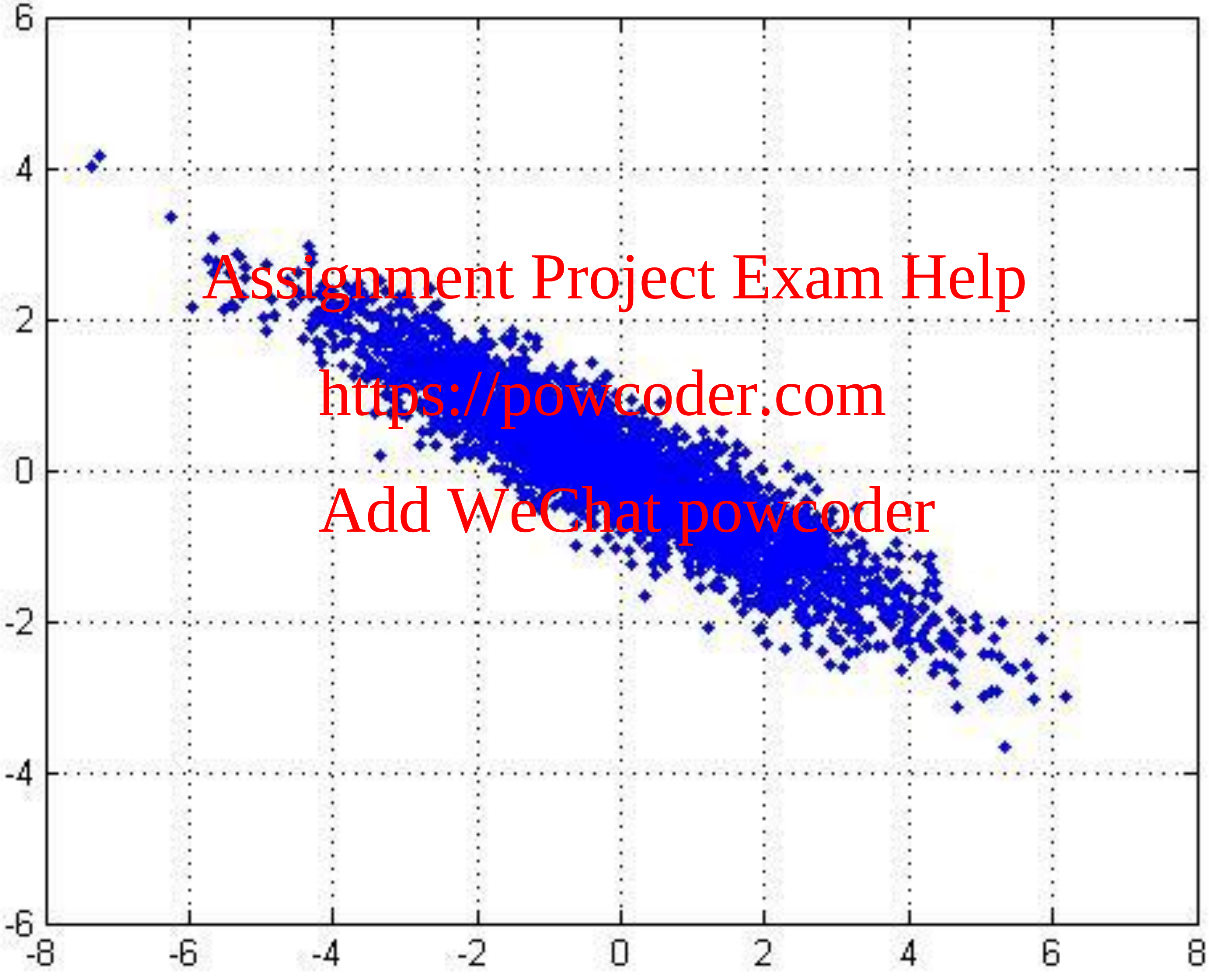
Properties:

- **PCA Vectors** originate from the center of mass.

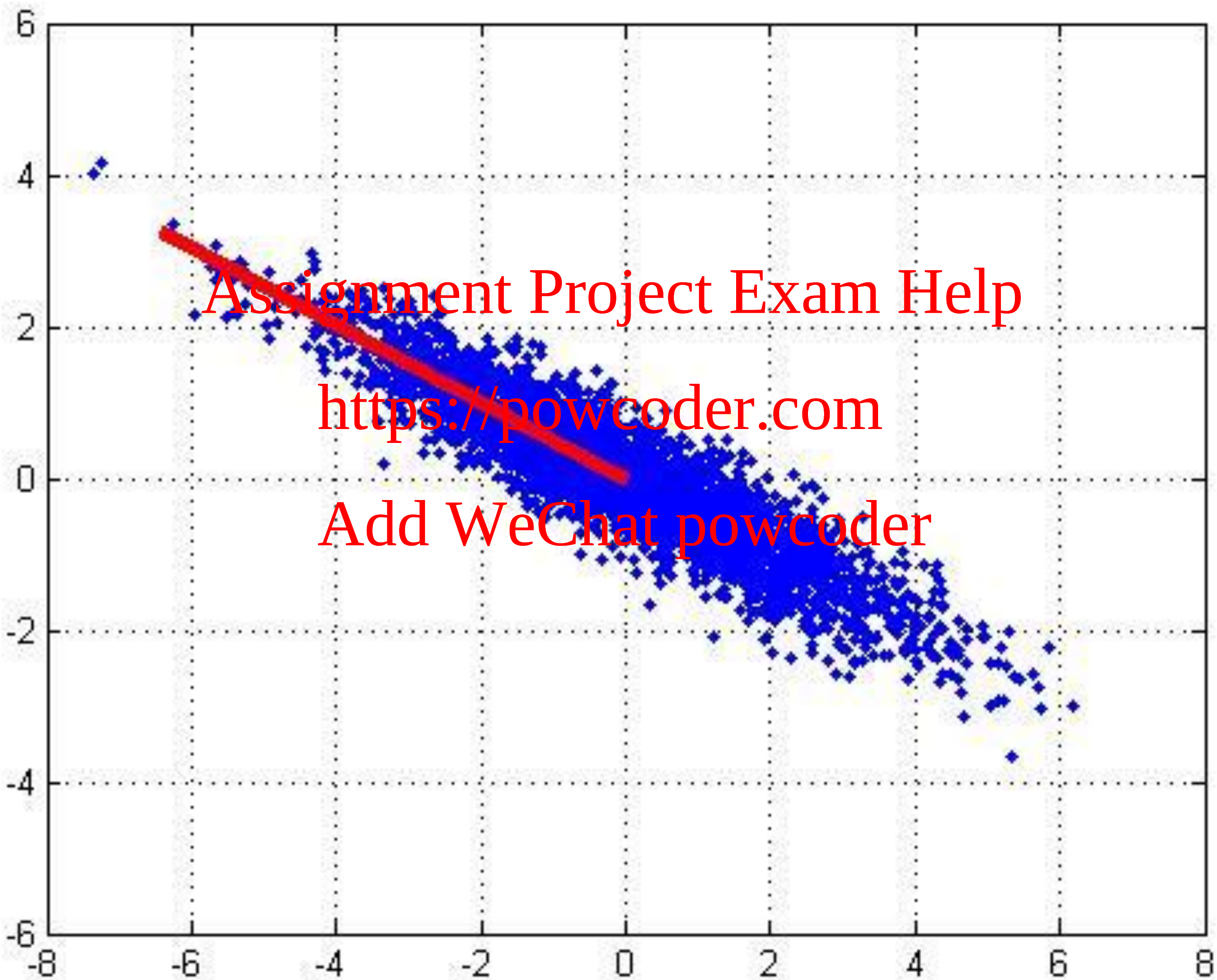
- Principal component #1: points in the direction of the **largest variance**.

- Each subsequent principal component
  - is **orthogonal** to the previous ones, and
  - points in the directions of the **largest variance of the residual subspace**

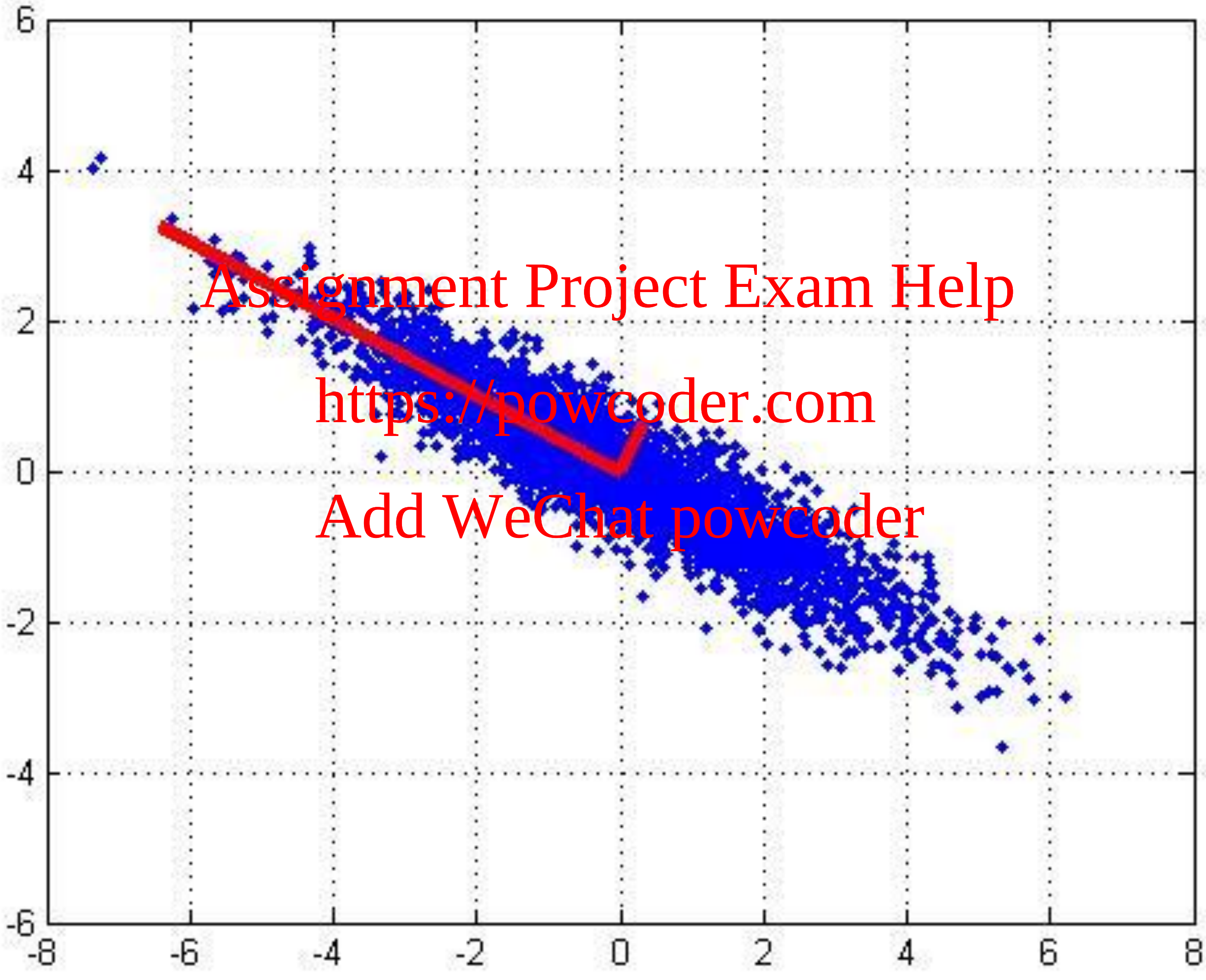
2D Gaussian dataset



1st PCA axis



2^{nd} PCA axis



PCA algorithm I (sequential)

Given the **centered** data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute the principal vectors:

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{(\mathbf{w}^T \mathbf{x}_i)^2\}$$

1st PCA vector

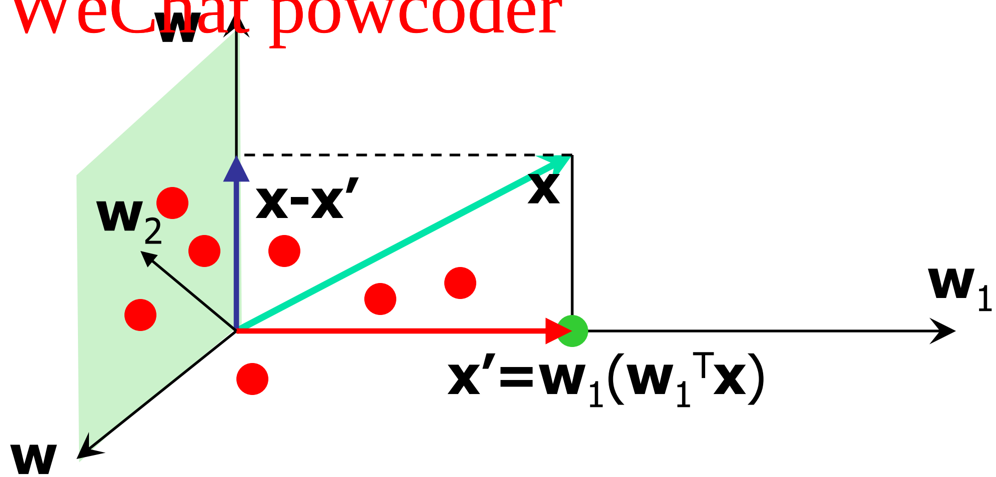
To find $\mathbf{w_1}$, maximize the variance of projection of $\mathbf{x}$

$$\mathbf{w}_2 = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{[\mathbf{w}^T (\mathbf{x}_i - \underbrace{\mathbf{w}_1 \mathbf{w}_1^T \mathbf{x}_i}_{\mathbf{x}'})]^2\}$$

2nd PCA vector

$\mathbf{x}'$ PCA reconstruction

To find $\mathbf{w_2}$, we maximize the **variance** of the projection in the **residual** subspace



PCA algorithm I (sequential)

Given $\mathbf{w}_1, \ldots, \mathbf{w}_{k-1}$, we calculate $\mathbf{w}_k$ principal vector as before:

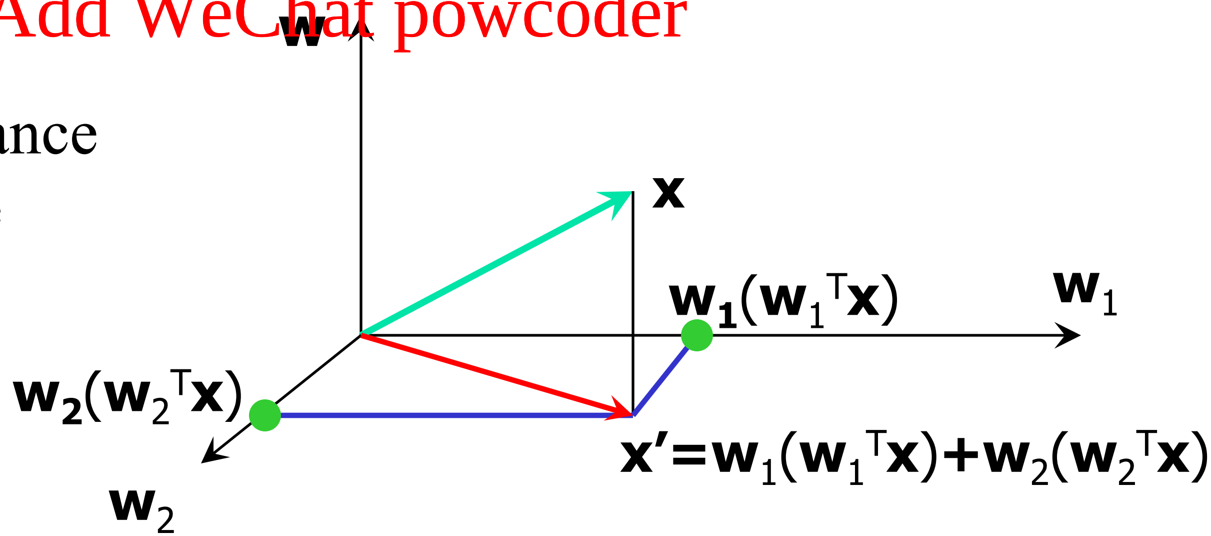
Maximize the variance of projection of $\mathbf{x}$

$$\mathbf{w}_k = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{[\mathbf{w}^T (\mathbf{x}_i - \underbrace{\sum_{j=1}^{k-1} \mathbf{w}_j \mathbf{w}_j^T \mathbf{x}_i}_{\mathbf{x}'})]^2\}$$

k^{th} PCA vector

$\mathbf{x}'$ PCA reconstruction

We maximize the variance of the projection in the residual subspace



PCA algorithm II (sample covariance matrix)

- Given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute covariance matrix Σ

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T$$

where

$$\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$$

- **PCA** basis vectors = the eigenvectors of Σ

- Larger eigenvalue $\Rightarrow$ more important eigenvectors

PCA algorithm II (sample covariance matrix)

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

% $\underline{\mathbf{X}}$ = N × m data matrix,
% … each data point $\mathbf{x}_i$ = column vector, i=1..m

- $\underline{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i$

- $\mathbf{X}$ ← subtract mean $\underline{\mathbf{x}}$ from each column vector $\mathbf{x}_i$ in $\underline{\mathbf{X}}$

- $\Sigma \leftarrow \mathbf{X}\mathbf{X}^T$ … covariance matrix of $\mathbf{X}$
- $\{ \lambda_i, \mathbf{u}_i \}_{i=1..N}$ = eigenvectors/eigenvalues of Σ
  … $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$
- Return $\{ \lambda_i, \mathbf{u}_i \}_{i=1..k}$
  % top k PCA components

PCA algorithm III
(SVD of the data matrix)

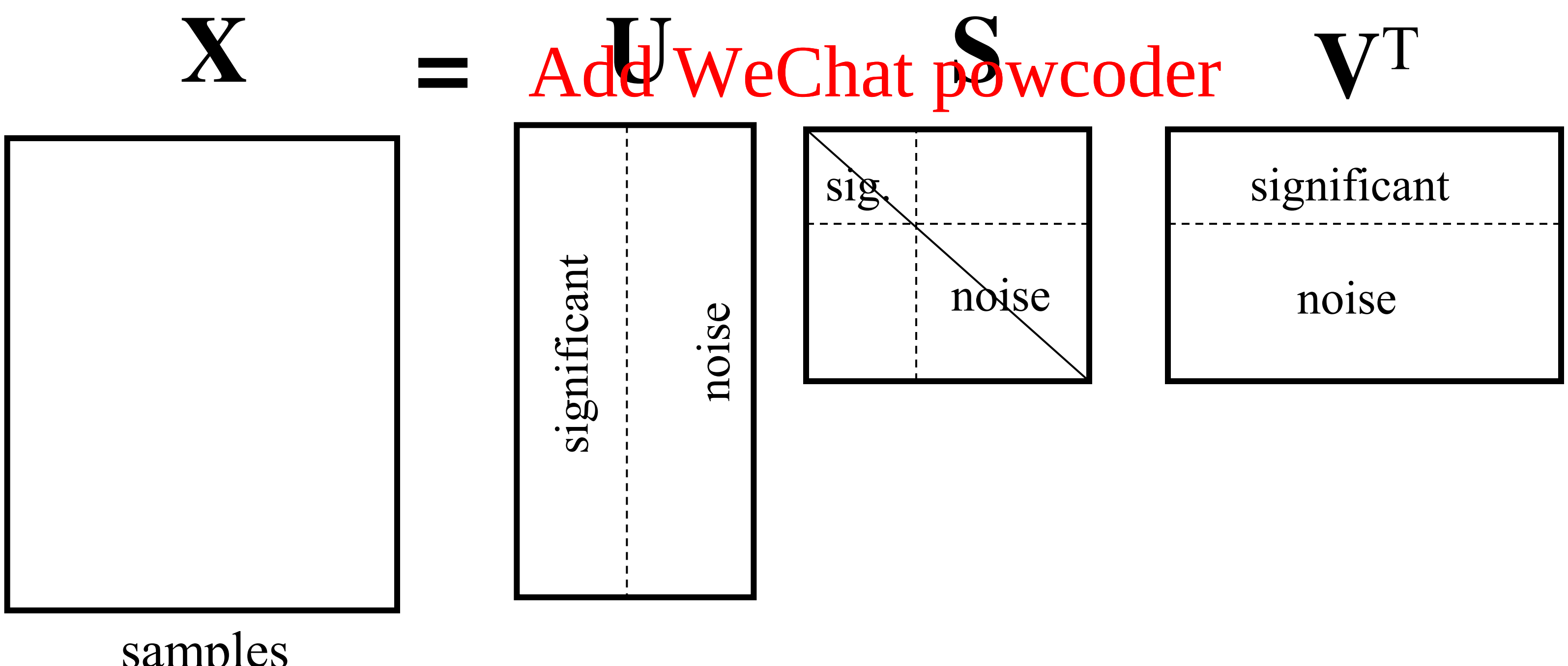
Singular Value Decomposition of the **centered** data matrix **X**.

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{N \times m},$$

m: number of instances,

N: dimension

$$\mathbf{X}_{\text{features} \times \text{samples}} = \mathbf{USV}^{\mathrm{T}}$$



PCA algorithm III

- **Columns of U**
  - the principal vectors, $\{ \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)} \}$
  - orthogonal and has unit norm – so $U^T U = I$
  - Can reconstruct the data using linear combinations of $\{ \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)} \}$

- **Matrix S**
  - Diagonal
  - Shows importance of each eigenvector

- **Columns of V^T**
  - The coefficients for reconstructing the samples

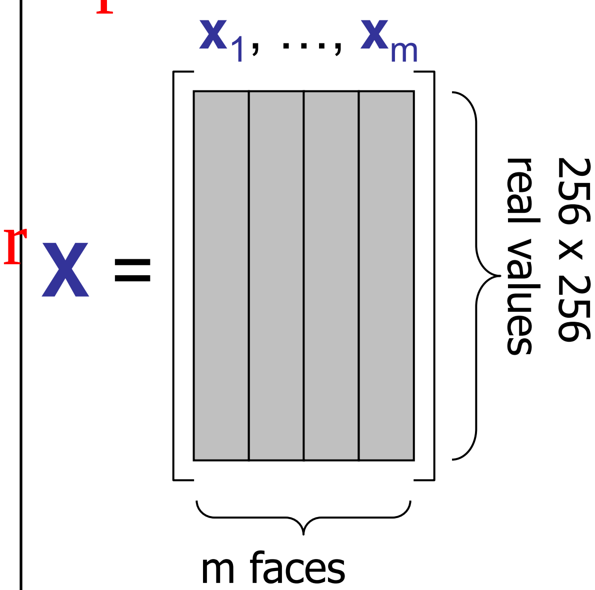
Applications

Face Recognition

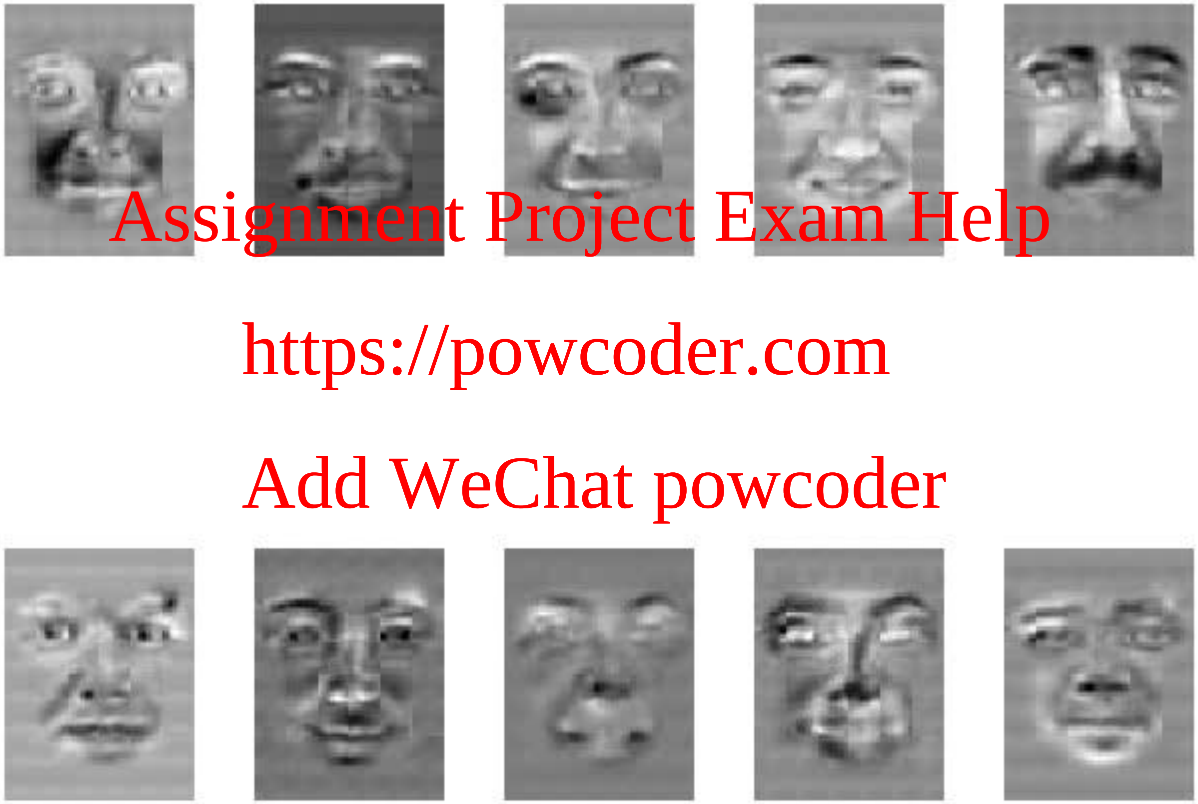
- Want to identify specific person, based on facial image
- Robust to glasses, lighting,…
  - ⇒ Can't just use the given 256 x 256 pixels

Applying PCA: Eigenfaces

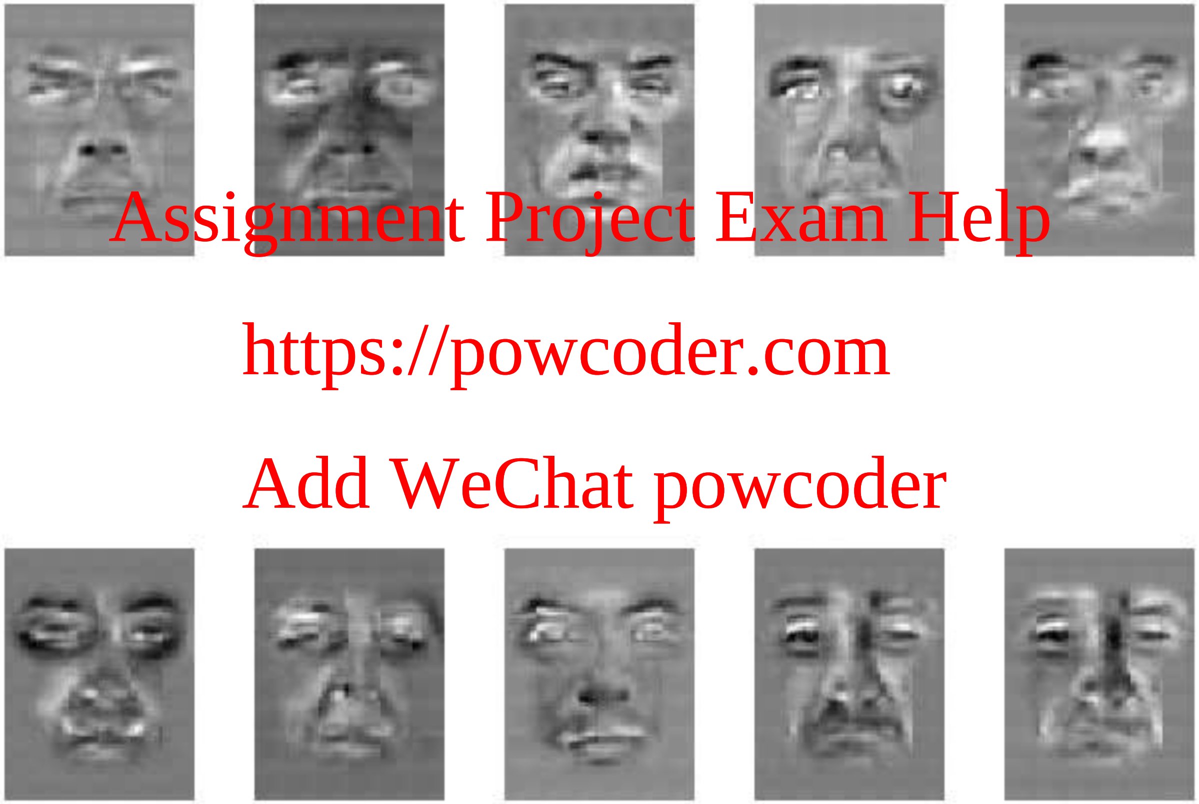
- Example data set: Images of faces
  - Eigenface approach [Turk & Pentland], [Sirovich & Kirby]
- Each face $\mathbf{x}$ is …
  - 256×256 values (luminance at location)
  - $\mathbf{x}$ in $\Re^{256\times256}$ (view as 64K dim vector)
- Form $\mathbf{X} = [\, \mathbf{x}_1 , \ldots, \mathbf{x}_m \,]$ **centered** data mtx
- Compute $\Sigma = \mathbf{X}\mathbf{X}^T$
- Problem: Σ is $64K \times 64K$ … HUGE!!!



Happiness subspace



Disgust subspace



Facial Expression Recognition Movies

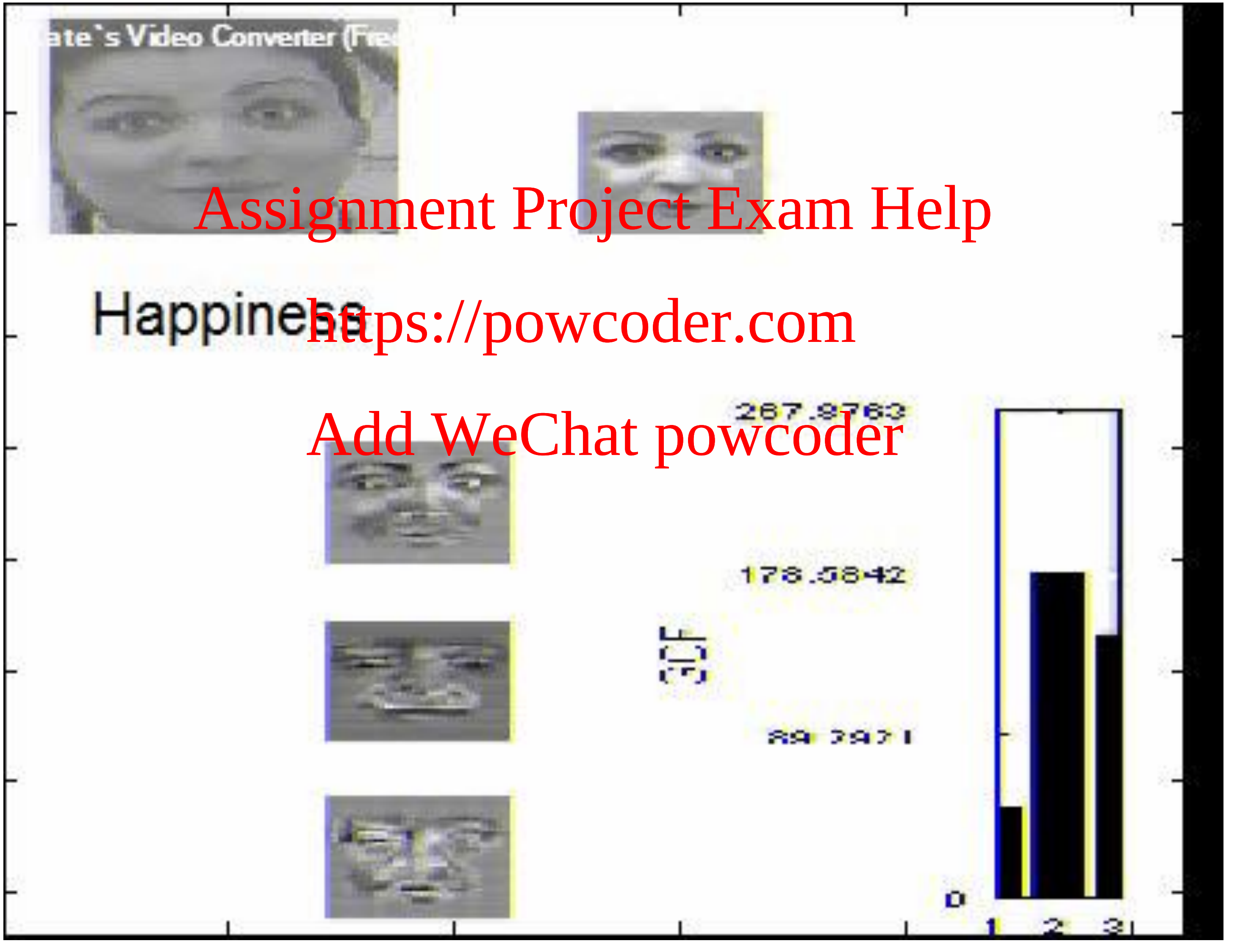
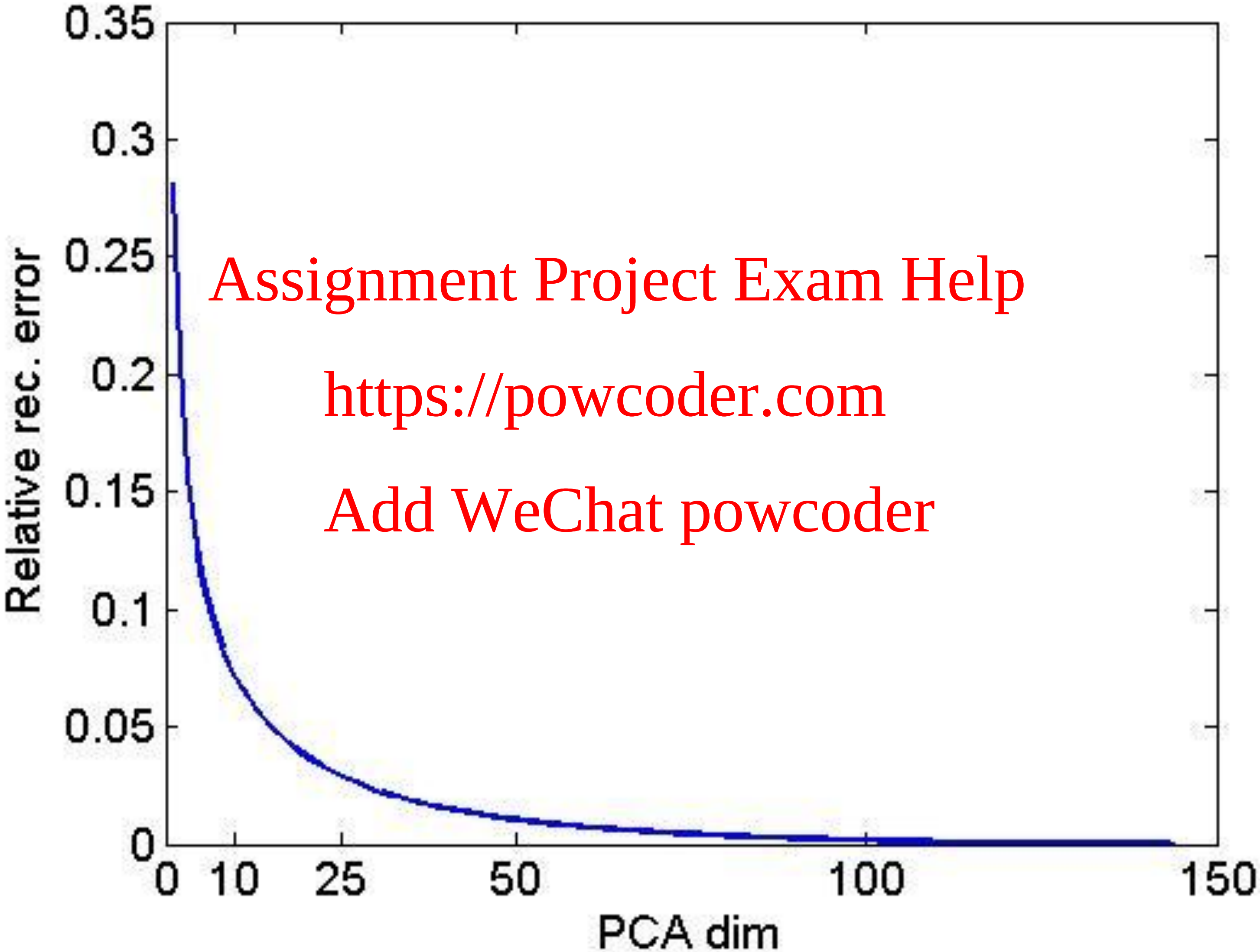


Image Compression

Original Image

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

L_2 error and PCA dim



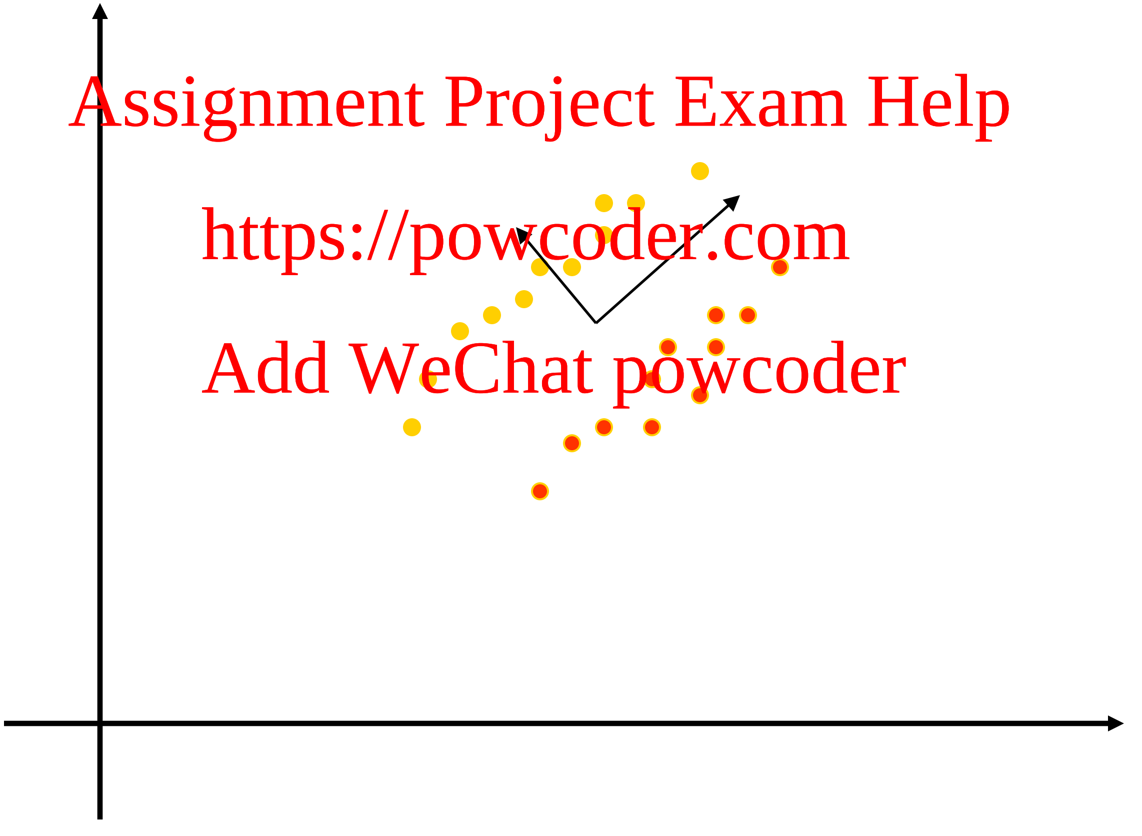
PCA compression: 144D $\Rightarrow$ 60D

60 most important eigenvectors

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

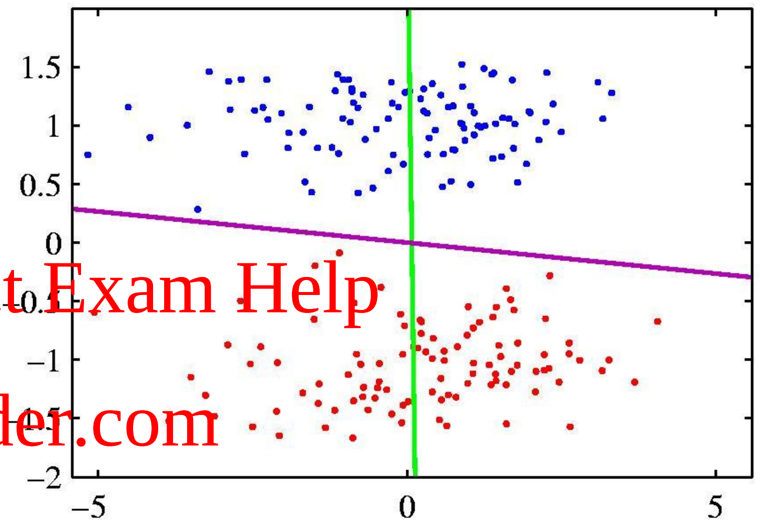
PCA Shortcomings

Problematic Data Set for PCA



PCA doesn’t know about class labels!

PCA vs Fisher Linear Discriminant



- PCA maximizes variance, *independent of class*

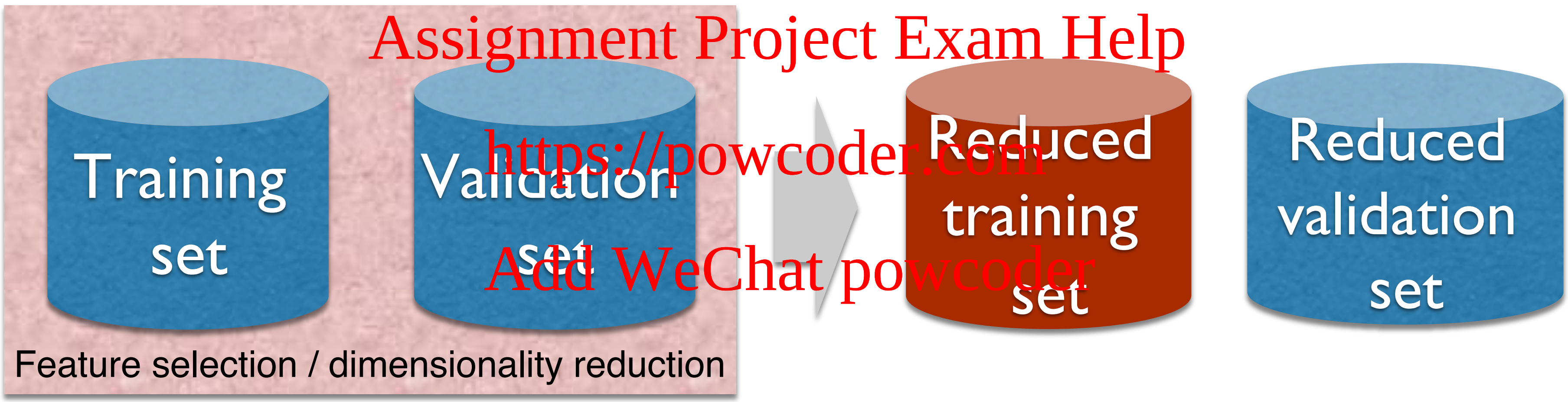
  $\Rightarrow$ magenta

- FLD attempts to separate classes

  $\Rightarrow$ green line

Feature Selection and Cross-Validation

- Incorrect way: DO NOT do feature selection (or dimensionality reduction) on the whole dataset, and then cross-validation



- Feature selection and dimensionality reduction on the whole dataset destroys cross-validation
  - *reduced training set* would depend on the validation set
  - Thus, *training is looking at the supposedly "unseen" data*

Feature Selection and Cross-Validation

- Correct way: feature selection (or dimensionality reduction) *inside cross-validation*, only applied to the training set