CS M146: Introduction to Machine Learning
Dimensionality Reduction

Aditya Grover

https://aditya-grover.github.io/ @adityagrover_

Visualizing High Dimensional Data

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

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...

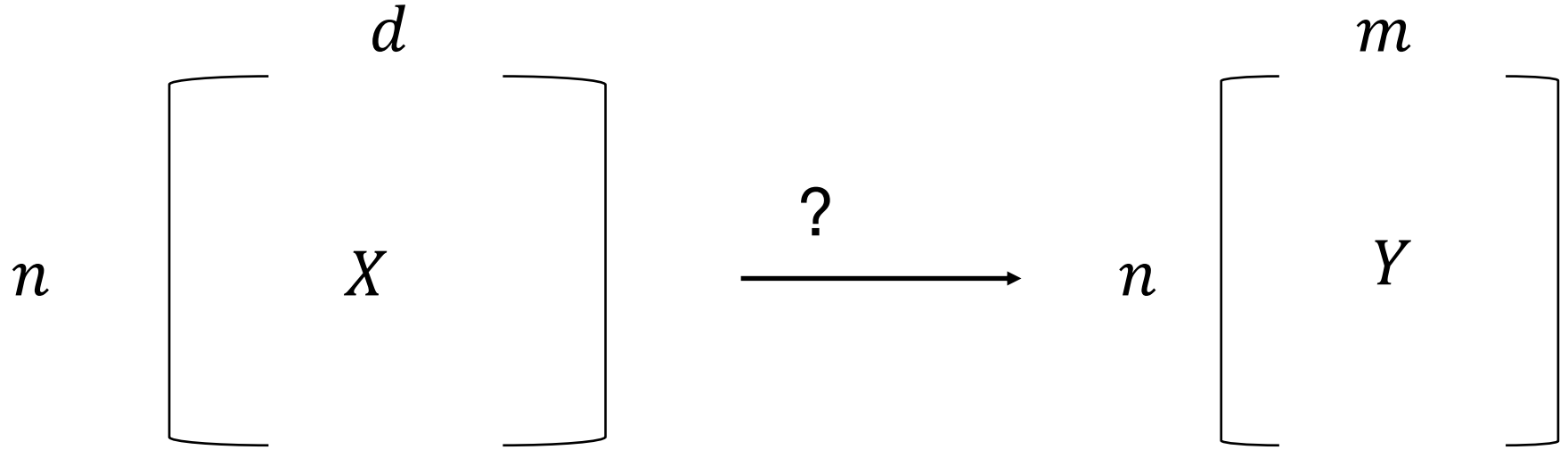
Data Visualization

- 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 a **small subspace** of the 53-D space that keeps the **most information** about the original data?

Dimensionality Reduction

- Input $X = \left[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}\right]^T \in \mathbb{R}^{n \times d}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$
- Goal: Represent X by $Y = \left[\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \cdots, \boldsymbol{y}^{(n)}\right]^T \in \mathbb{R}^{n \times m}$ where $\boldsymbol{y}^{(i)} \in \mathbb{R}^m$ such that:
  - Typically, $m \ll d$ (**dimensionality reduction** or **compression**)
  - Y captures most of the **information** in X (information preserving)



Preserving Information Via Reconstruction

- How can we measure information preservation between X, Y?
- Consider an encoding-decoding mechanism for dimensionality reduction:
  - Encoding (**Principal Component Analysis or PCA)**:
    $\boldsymbol{y}^{(i)} = W\boldsymbol{x}^{(i)}$ where $W \in \mathbb{R}^{m \times d}$ and $\boldsymbol{y}^{(i)} \in \mathbb{R}^{m}$
  - Decoding (**Reverse PCA**):
    $\tilde{\boldsymbol{x}}^{(i)} = U\boldsymbol{y}^{(i)}$ where $U \in \mathbb{R}^{d \times m}$ and $\tilde{\boldsymbol{x}}^{(i)} \in \mathbb{R}^{d}$
- To learn PCA encoding-decoding, we minimize reconstruction loss:

$$\min_{U,W} \sum_{i=1}^{n} \|\boldsymbol{x}^{(i)} - \tilde{\boldsymbol{x}}^{(i)}\|_2^2 = \sum_{i=1}^{n} \|\boldsymbol{x}^{(i)} - UW\boldsymbol{x}^{(i)}\|_2^2$$

$$= \|X - XW^T U^T\|_F^2 \text{ (via vectorization)}$$

(where $\|M\|_F^2 = \sum_i \sum_j M_{i,j}^2$ denotes the squared Frobenius norm)

Solving PCA

- **Lemma 1 (without proof):** For the PCA objective, the optimal solution for U is such that the columns of U are orthogonal i.e. $U^T U = I_m$ and $W = U^T$
- **Implication:** Can rewrite PCA objective as:

$$\min_U \sum_{i=1}^{n} \|\boldsymbol{x}^{(i)} - UU^T \boldsymbol{x}^{(i)}\|_2^2$$

$$\text{s.t. } UU^T = I_d$$

Constrained optimization problem

Solving PCA

- **Lemma 2 (without proof):** The solution to the PCA learning objective is given by the first m eigenvectors of the empirical covariance matrix
  - Let $A = \frac{1}{(n-1)} \sum_{i=1}^{n} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^T$ be the empirical covariance matrix. Here, $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}^{(i)}$ is the mean vector.
  - Let $\boldsymbol{v}_i \in \mathbb{R}^d$ denote the eigenvector for the i-th **largest** eigenvalue of A. Let $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d]$ be the matrix of eigenvectors.
  - The solution to the PCA learning objective is given by the first m columns of V i.e., $U = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m]$
- **Implication:** Can solve for PCA on X by performing an eigendecomposition of A
  - $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$ are also known as the **principal components directions**
  - $\boldsymbol{v}_1^T \boldsymbol{x}$, $\boldsymbol{v}_2^T \boldsymbol{x}$, **…**, $\boldsymbol{v}_m^T \boldsymbol{x}$ are also known as the **principal components scores**

Pseudocode

- **Step 1:** Compute covariance matrix $A \in \mathbb{R}^{d \times d}$

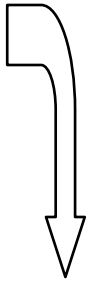
$$A = \frac{1}{n-1}(X - \boldsymbol{\mu})^T(X - \boldsymbol{\mu}) = \frac{1}{(n-1)}\sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^T$$

- **Step 2:** Compute the eigenvectors of A
  - V, Λ = numpy.linalg.eig(A)
  - $\{\boldsymbol{v}_i, \lambda_i\}_{i=1,\ldots,d}$ are the eigenvectors/eigenvalues of A sorted in descending order of eigenvalues
- **Step 3:** Return the top m eigenvectors $U = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m]$ as the principal component directions
- For any point $\boldsymbol{x}$, its reduced representation is $\boldsymbol{y} = W\boldsymbol{x} = U^T\boldsymbol{x}$

PCA

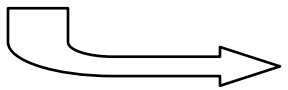
$$X = \begin{bmatrix} 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ldots \\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ldots \\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ldots \\ \vdots \\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ldots \end{bmatrix}$$

X has d columns



V is the eigenvectors of covariance matrix A;
columns are ordered by importance!

V is $d \times d$

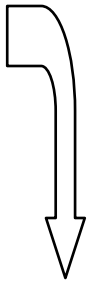


$$V = \begin{bmatrix} 0.34 & 0.23 & -0.30 & -0.23 & \ldots \\ 0.04 & 0.13 & -0.40 & 0.21 & \ldots \\ -0.64 & 0.93 & 0.61 & 0.28 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ -0.20 & -0.83 & 0.78 & -0.93 & \ldots \end{bmatrix}$$

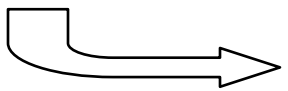
Slide by Eric Eaton

PCA

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



Each row of V corresponds to a feature; keep only first m columns of V to get U



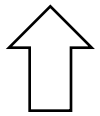
$$U = \begin{bmatrix} 0.34 & 0.23 & -0.30 & -0.23 & \ldots \\ 0.04 & 0.13 & -0.40 & 0.21 & \ldots \\ -0.64 & 0.93 & 0.61 & 0.28 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ -0.20 & -0.83 & 0.78 & -0.93 & \ldots \end{bmatrix}$$

U is $d \times m$

Slide by Eric Eaton

PCA

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



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

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

Slide by Eric Eaton

PCA

- We can apply these formulas to get the new representation for each instance in X. E.g., if $m = 2$,

$$X = \begin{bmatrix} 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ldots \\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ldots \\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ldots \\ \vdots \\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ldots \end{bmatrix} \qquad U = \begin{bmatrix} 0.34 & 0.23 \\ 0.04 & 0.13 \\ -0.64 & 0.93 \\ \vdots & \vdots \\ -0.20 & -0.83 \end{bmatrix}$$

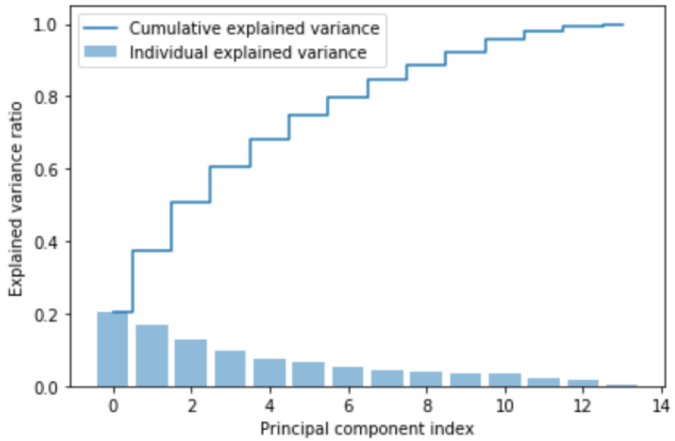
- The new 2D representation for $\boldsymbol{x}^{(3)}$ is given by:

$$\hat{x}_1^{(3)} = 0.34(0) + 0.04(0) - 0.64(1) + \ldots$$

$$\hat{x}_2^{(3)} = 0.23(0) + 0.13(0) + 0.93(1) + \ldots$$

- Vectorization: The re-projected data matrix is given by $\hat{X} = U^T X$

Explained Variance

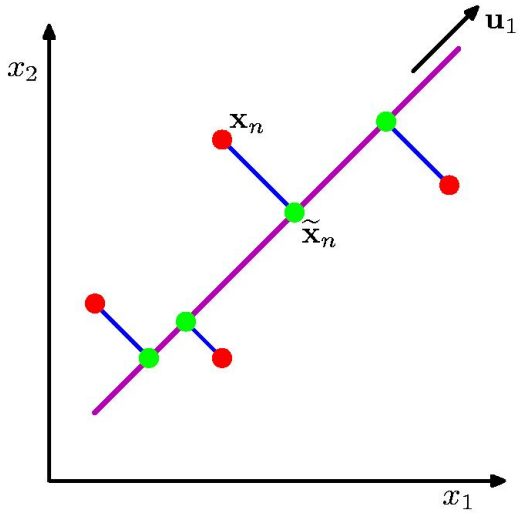


- **Explained variance** measures relative importance of each eigenvector
- For component i, its explained variance is given as:

$$\mathrm{EV}_i = \frac{\lambda_i}{\lambda_1+\lambda_2+\ldots+\lambda_d}$$

- PCA ignores the $d - m$ components with least importance
  - Larger eigenvalue $\Rightarrow$ more important eigenvectors
- PCA loses some information, but not much if the dropped eigenvalues are small

Principal Component Directions

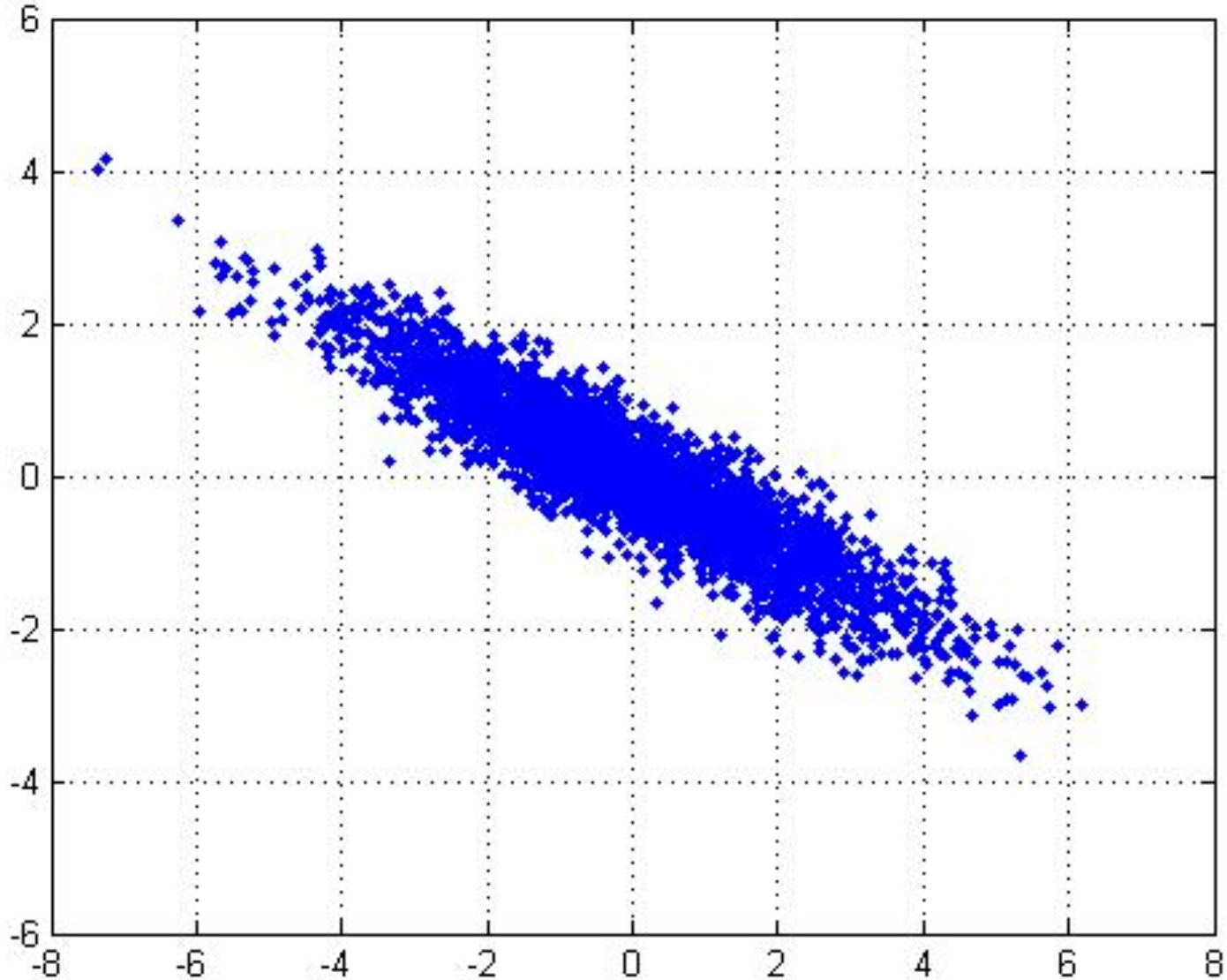


Geometric Interpretation (stated without proof): Principal component directions are orthogonal projection of data onto lower-dimension linear space that:

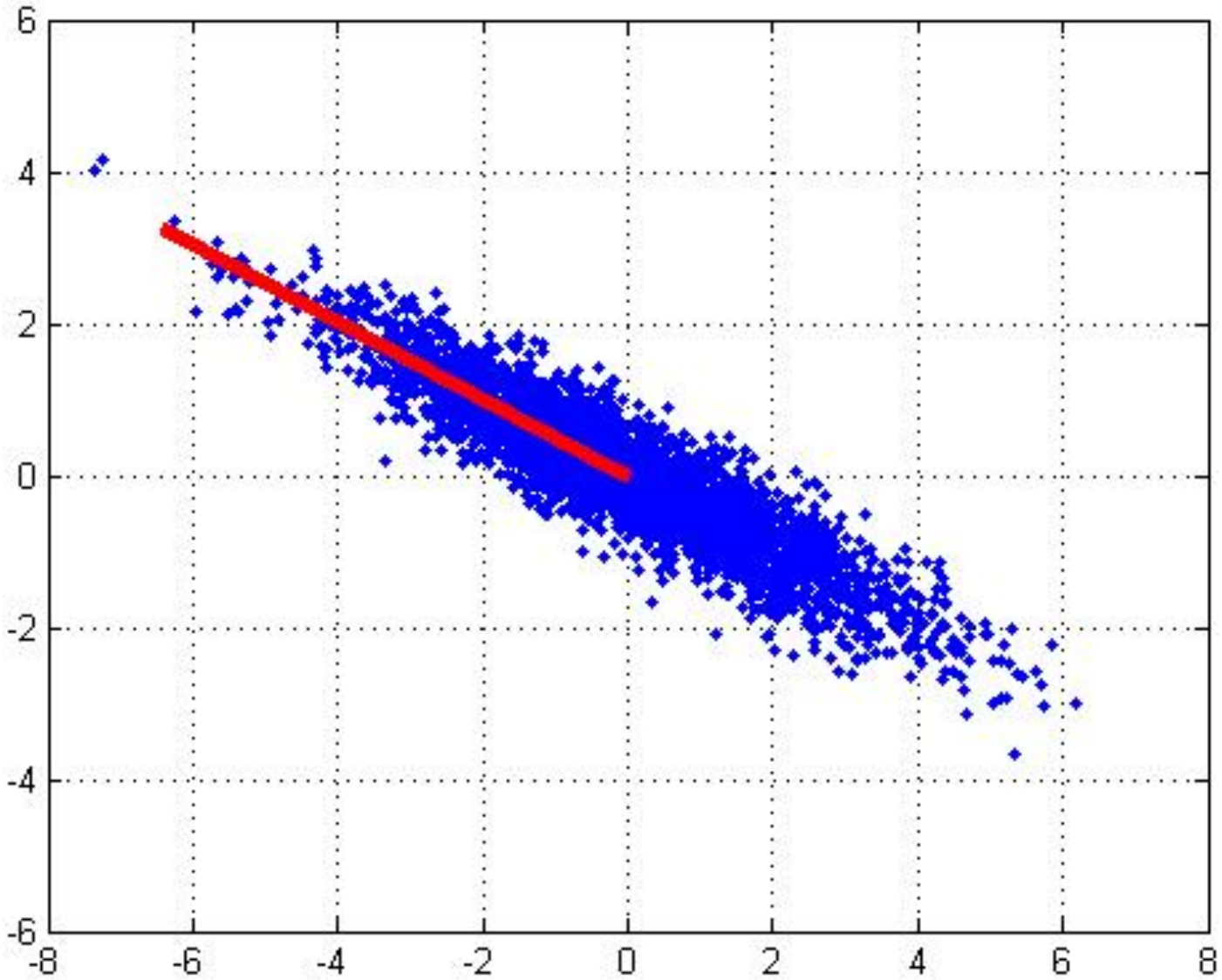
- maximizes variance of projected data (purple line) i.e., spread of data along the largest eigenvector
- minimizes mean squared distance between data point and projections (sum of blue lines)

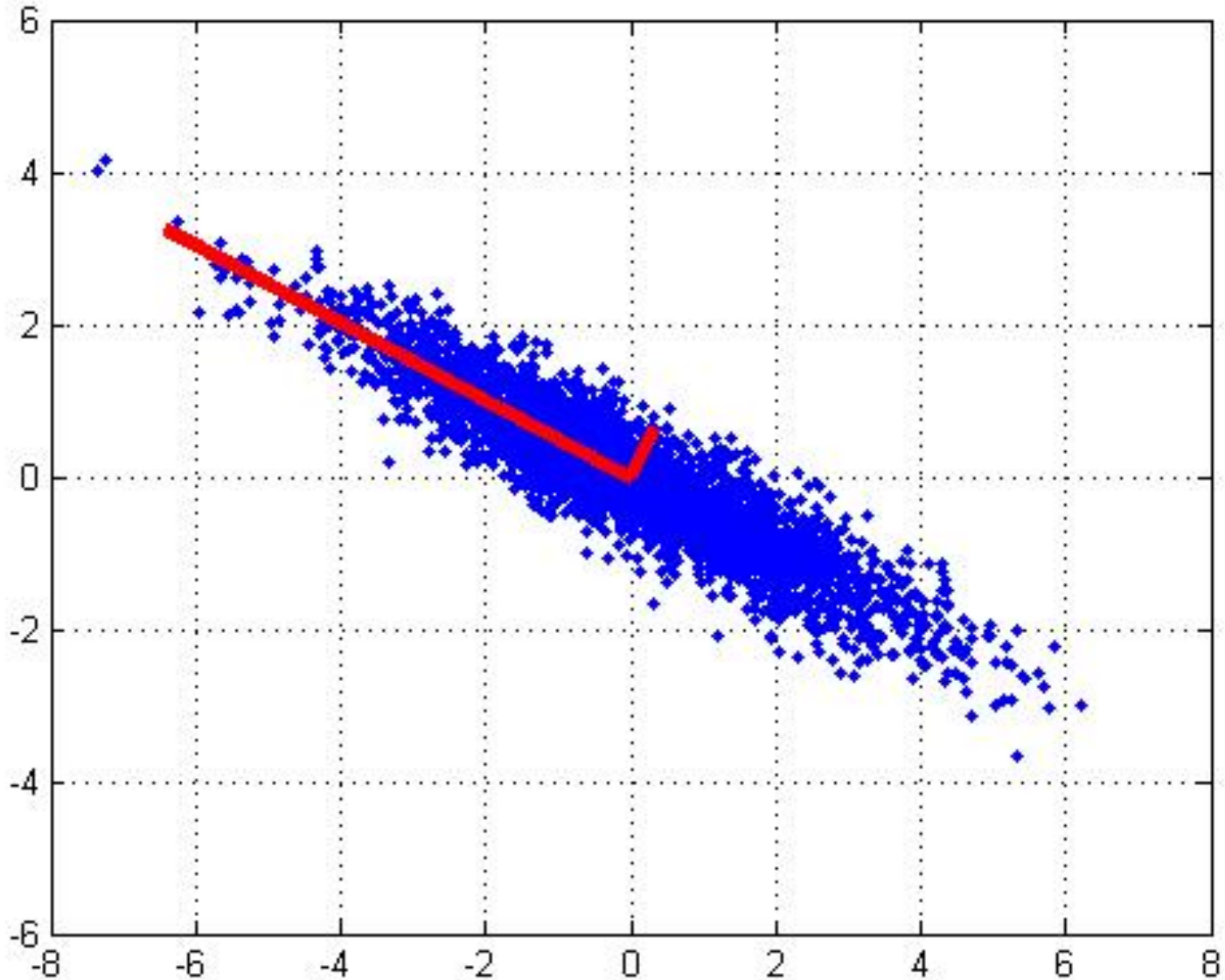
Principal Component Directions

- **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**

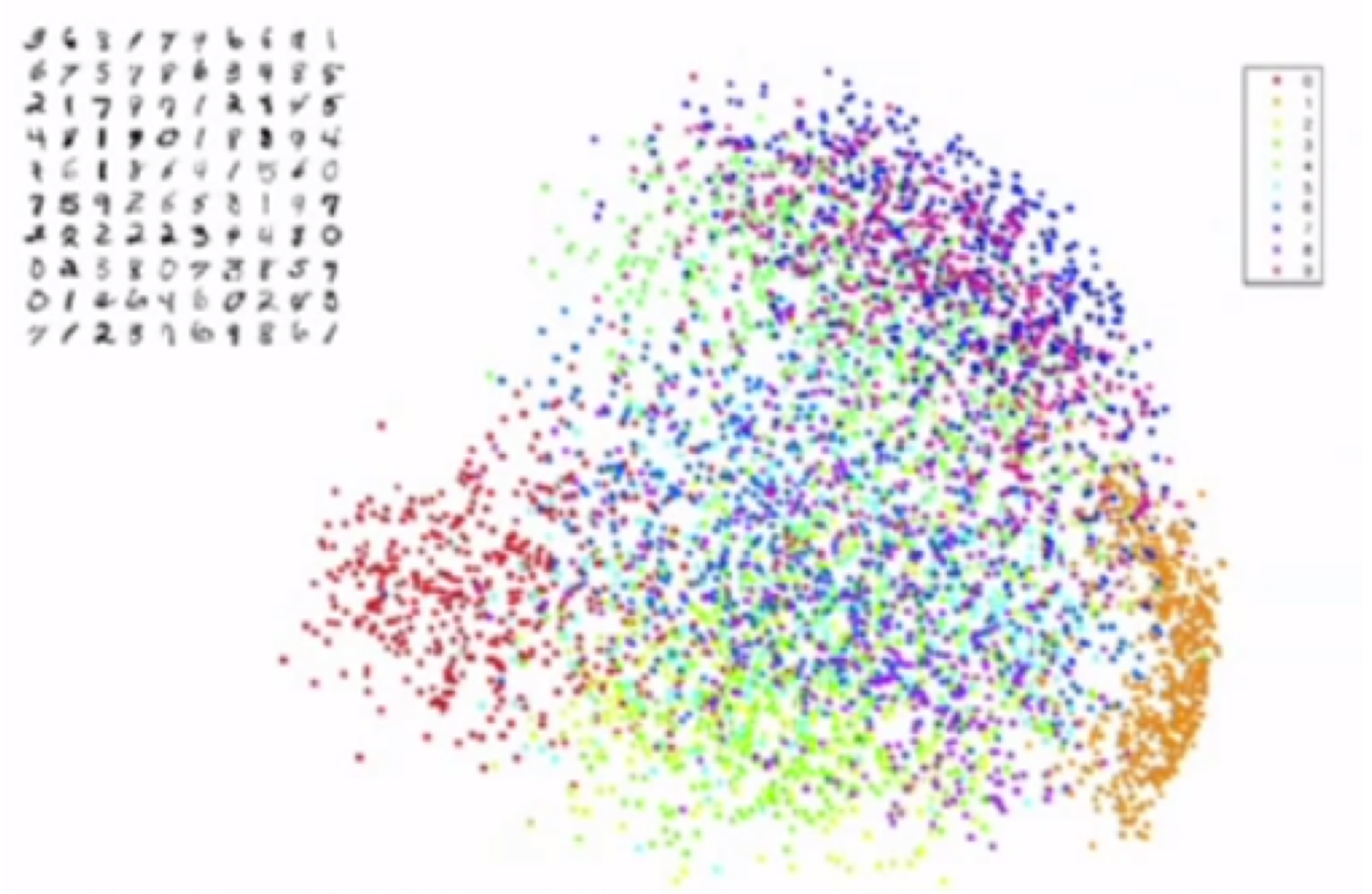

6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8

1st PCA axis




6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8

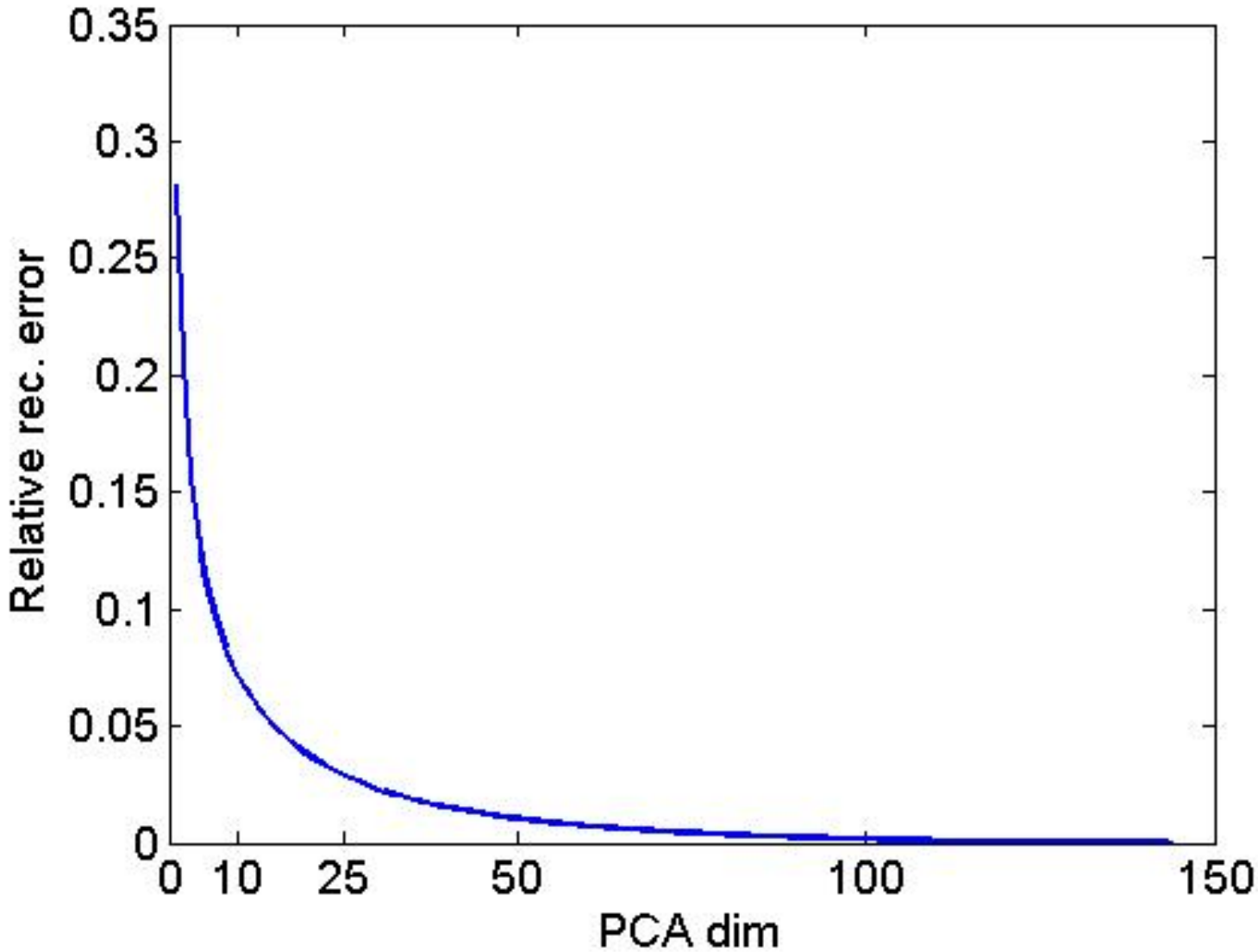
PCA Visualization of MNIST Digits



Application of PCA: Image Compression

Original Image

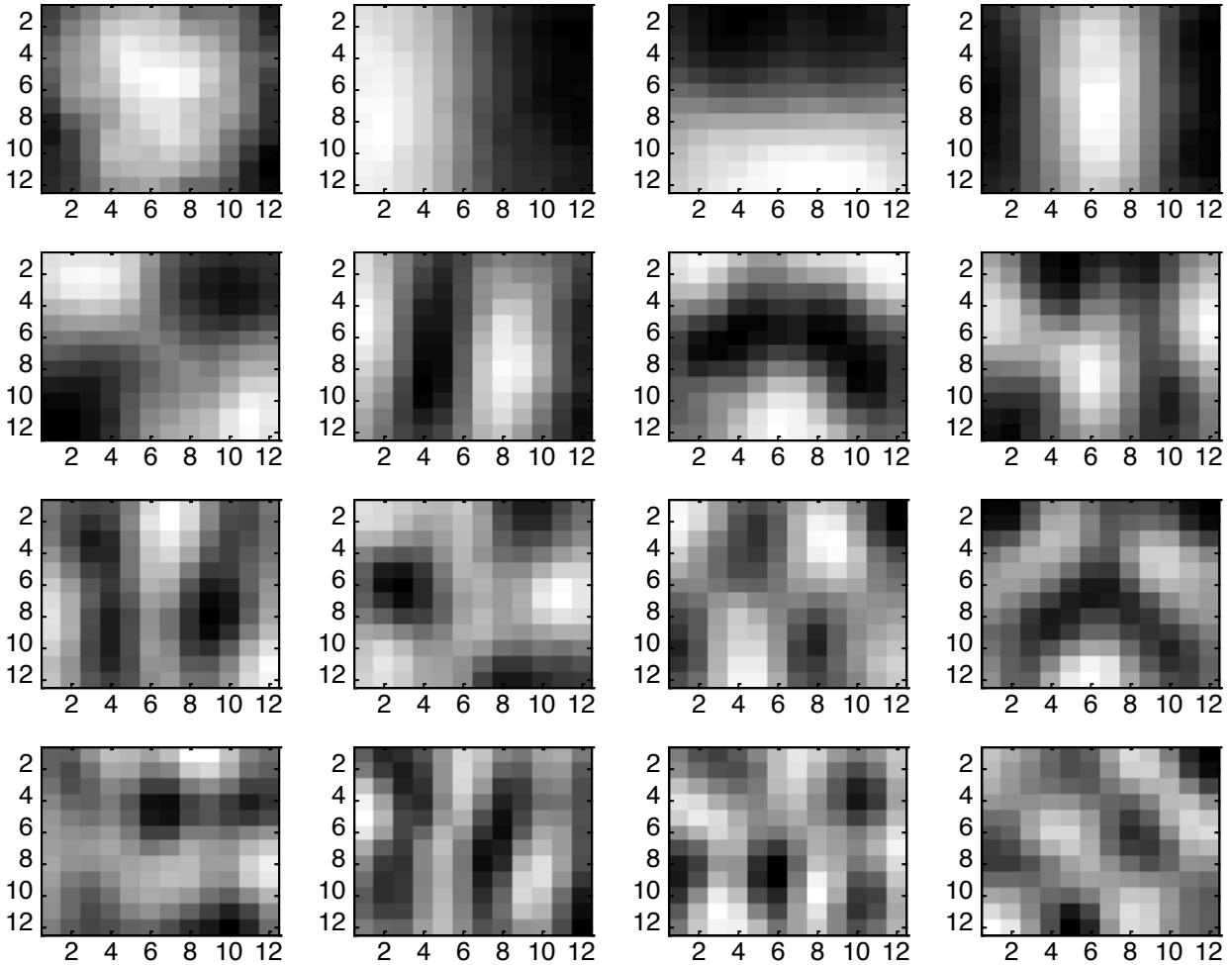
- Divide a single original 372x492 image into multiple patches
- Each patch is an instance that contains 12x12 pixels on a grid
- View each patch as a 144 dimension vector instance i.e., a training example $\boldsymbol{x}^{(i)}$
- Apply PCA on the dataset of patches


0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Relative rec. error
0
10
25
50
100
150
PCA dim

PCA compression: 144D → 60D

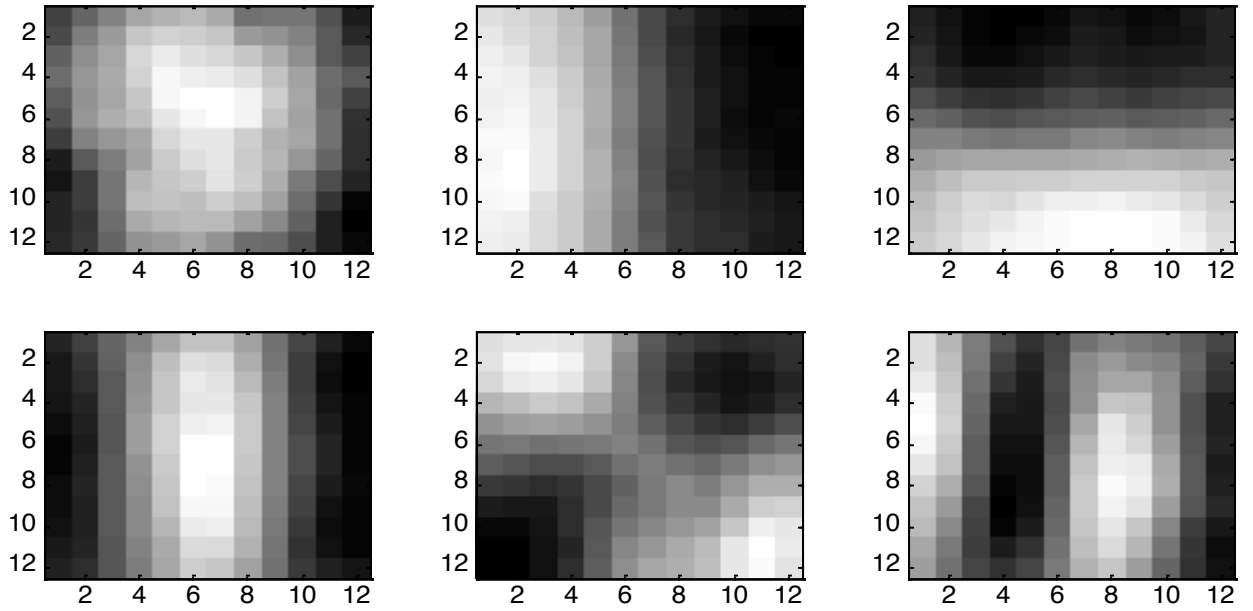
PCA compression: 144D → 16D

16 most important eigenvectors



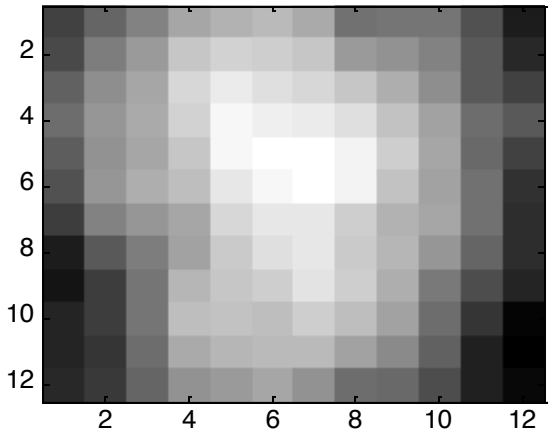
PCA compression: 144D → 6D

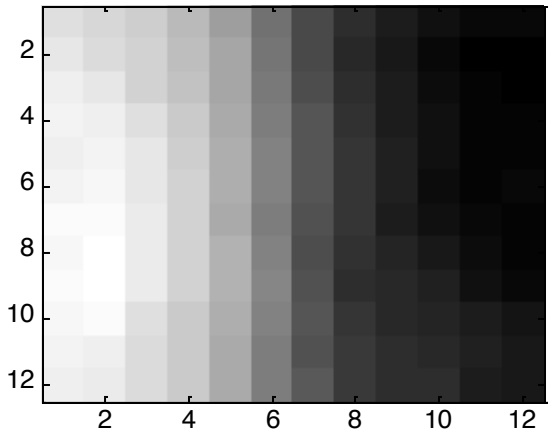
6 most important eigenvectors

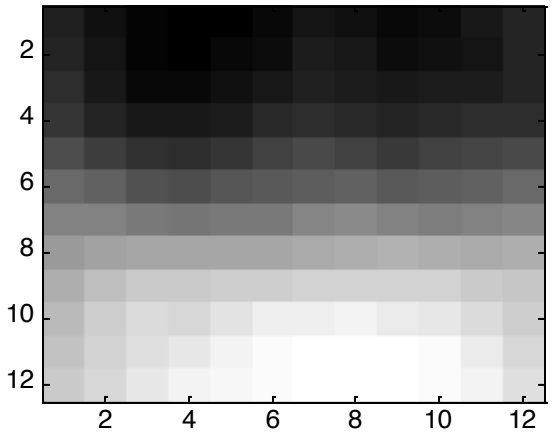


PCA compression: 144D → 3D

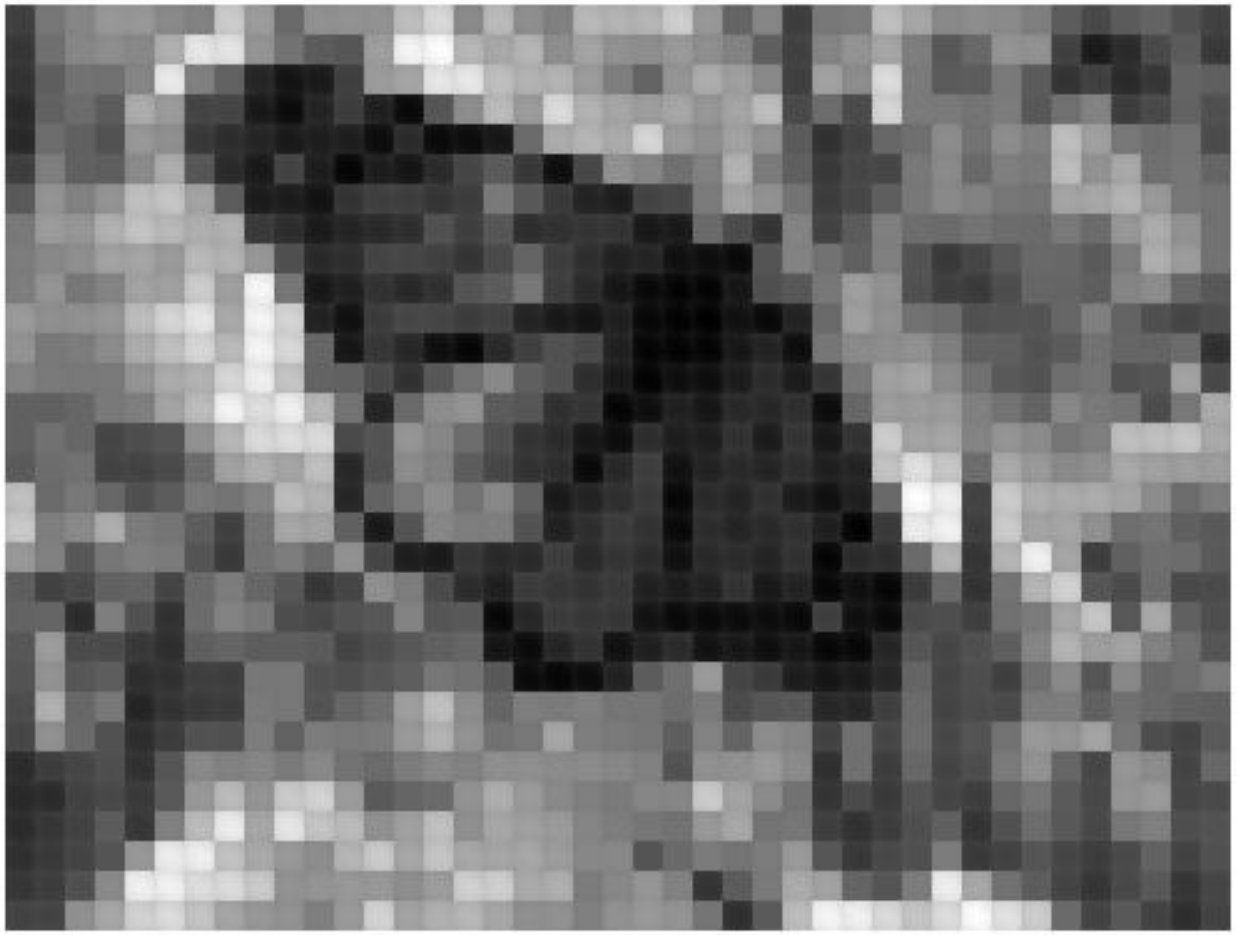
3 most important eigenvectors







PCA compression: 144D → 1D



Summary

- Principal Component Analysis (PCA)
  - Method for dimensionality reduction
  - Preserves maximum information by minimizing reconstruction error
  - Analytical solution can be obtained via eigendecomposition of covariance matrix
- Applications in data visualization, data compression