



Dimensionality Reduction

PCA

Machine Learning – CSE446

David Wadden (slides provided by Carlos Guestrin)

University of Washington

Feb 22, 2017

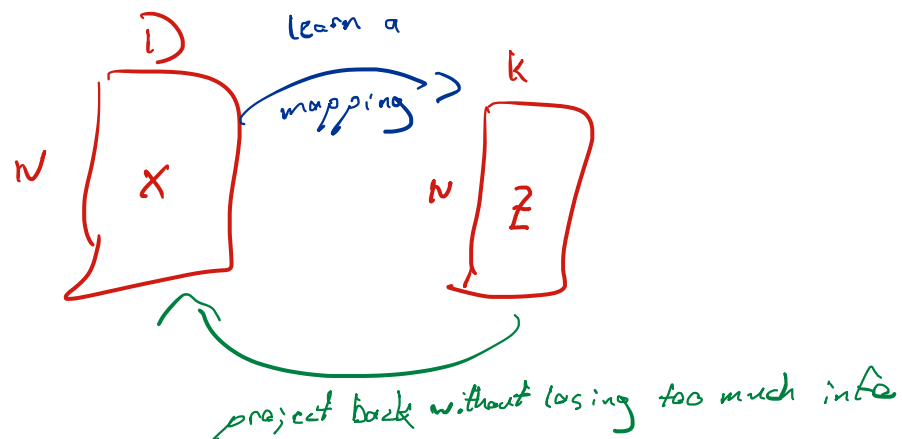
Dimensionality reduction

- Input data may have thousands or millions of dimensions!
 - e.g., text data has *x with 10,000 - 10,000,000 dims*
- **Dimensionality reduction:** represent data with fewer dimensions
 - easier learning – fewer parameters
 - visualization – hard to visualize more than 3D or 4D
 - discover “intrinsic dimensionality” of data
 - high dimensional data that is truly lower dimensional

Lower dimensional projections

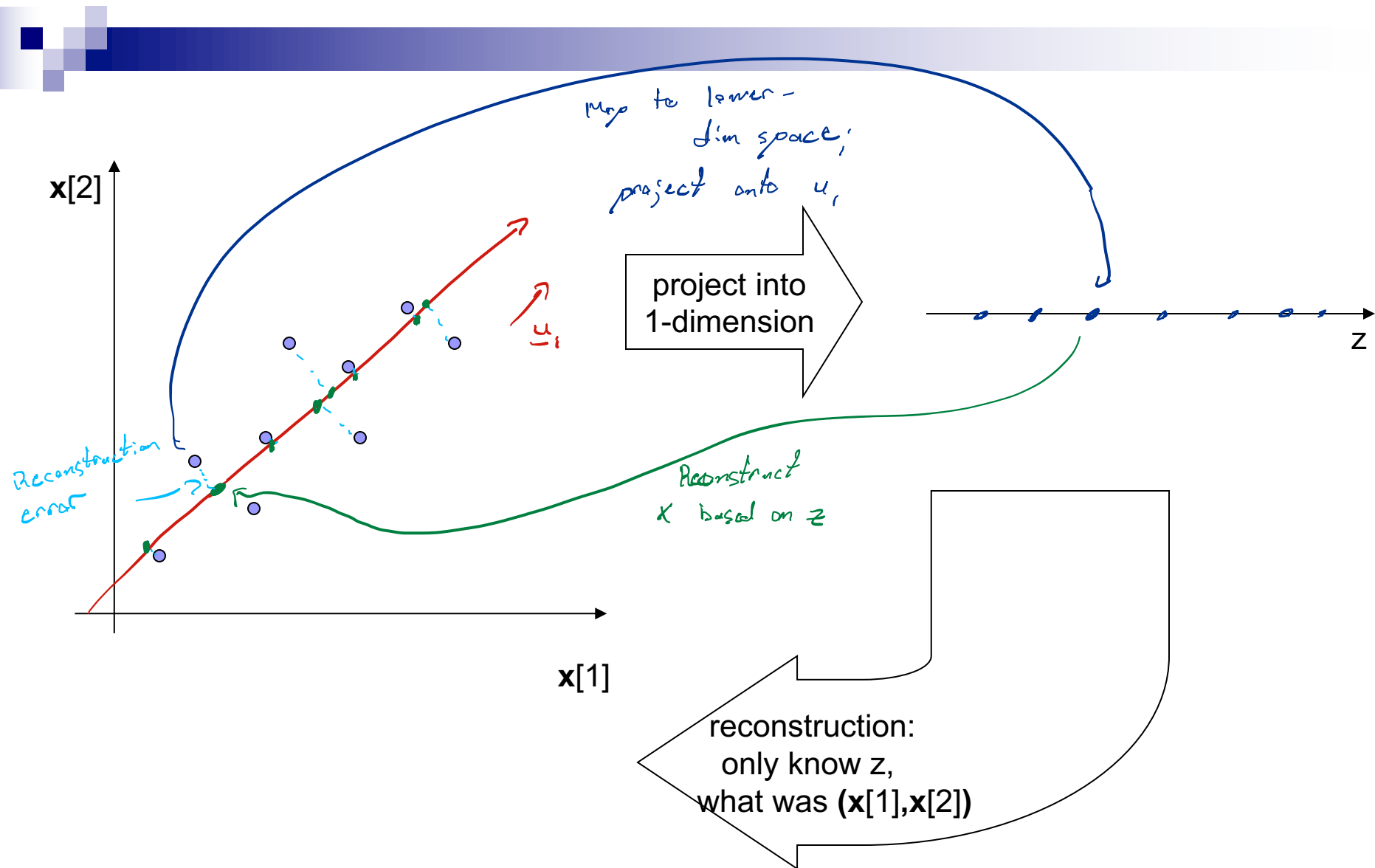
- Rather than picking a subset of the features, we can create new features that are combinations of existing features

$$z[2] = 2.5 \times [1] + 8.3 \times [2] - 3.6 \times [4] + \dots$$



- Let's see this in the unsupervised setting
 - just **X**, but no **Y**

Linear projection and reconstruction



Principal component analysis – basic idea

- Project d -dimensional data into k -dimensional space while preserving information:
 - e.g., project space of 10000 words into 3-dimensions
 - e.g., project 3-d into 2-d

- Choose projection with minimum reconstruction error

Linear projections, a review

- Project a point into a (lower dimensional) space:

- point: $\mathbf{x}_i = (\mathbf{x}_i[1], \dots, \mathbf{x}_i[D])$ *D is big.*

each u_i has length 1.

- select a basis – set of basis vectors – $(\mathbf{u}_1, \dots, \mathbf{u}_K)$

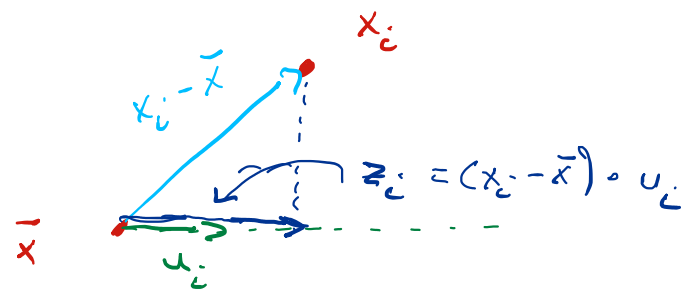
- we consider orthonormal basis:

- $\mathbf{u}_i \cdot \mathbf{u}_i = 1$, and $\mathbf{u}_i \cdot \mathbf{u}_j = 0$ for $i \neq j$ *mean of training data*

Goal: learn these from data.

- select a center – $\bar{\mathbf{x}}$, defines offset of space

- best coordinates in lower dimensional space defined by dot-products: $(\mathbf{z}_i[1], \dots, \mathbf{z}_i[K])$, $\mathbf{z}_i[j] = (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$



just another way of writing $(x_i - \bar{x})^T u_j$

PCA finds projection that minimizes reconstruction error

- Given N data points: $\mathbf{x}_i = (\mathbf{x}_i[1], \dots, \mathbf{x}_i[D])$, $i=1 \dots N$
- Will represent each point as a projection:

$$\square \hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^K \mathbf{z}_i[j] \mathbf{u}_j$$
 where: $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$ and $\mathbf{z}_i[j] = (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$

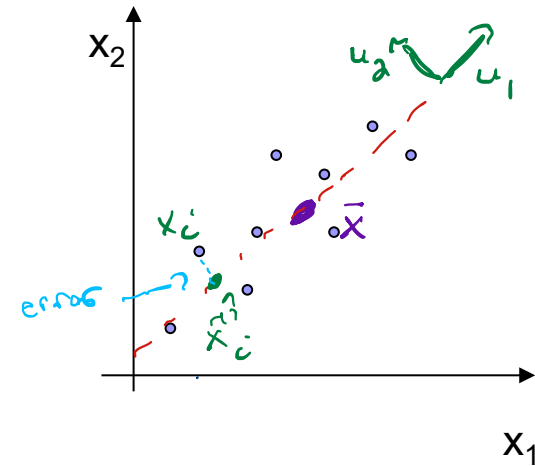
Handwritten notes:
 - $\hat{\mathbf{x}}_i$: reconstructed \mathbf{x}_i
 - $\mathbf{z}_i[j]$: each point has own $\mathbf{z}_i[j]$
 - \mathbf{u}_j : shared by all data

- PCA:

- Given $K < D$, find $(\mathbf{u}_1, \dots, \mathbf{u}_K)$ minimizing reconstruction error:

$$\text{error}_K = \sum_{i=1}^N (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2$$

Handwritten notes:
 - \mathbf{x}_i : truth
 - $\hat{\mathbf{x}}_i$: project down, then reconstruct



Understanding the reconstruction error

error

Note that \mathbf{x}_i can be represented exactly by d-dimensional projection:

$$\mathbf{x}_i = \bar{\mathbf{x}} + \sum_{j=1}^D \mathbf{z}_i[j] \mathbf{u}_j$$

Rewriting error:

$$\begin{aligned} \text{error}_K &= \sum_{i=1}^N (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2 = \sum_{i=1}^N \left[\bar{\mathbf{x}} + \sum_{j=1}^D \mathbf{z}_i[j] \mathbf{u}_j - \left(\bar{\mathbf{x}} + \sum_{j=1}^K \mathbf{z}_i[j] \mathbf{u}_j \right) \right]^2 \\ &= \sum_{i=1}^N \left[\sum_{j=K+1}^D \mathbf{z}_i[j] \mathbf{u}_j \right]^2 = \sum_{i=1}^N \left[\sum_{j=K+1}^D \underbrace{\mathbf{z}_i[j] \mathbf{u}_j \cdot \mathbf{u}_j \mathbf{z}_i[j]}_{=1} + 2 \sum_{j=K+1}^D \sum_{\ell > j} \underbrace{\mathbf{z}_i[j] \mathbf{u}_j \cdot \mathbf{u}_\ell \mathbf{z}_i[\ell]}_{=0} \right] \\ &= \sum_{i=1}^N \sum_{j=K+1}^D (\mathbf{z}_i[j])^2 \end{aligned}$$

Minimize the projection into the dimensions that are ignored

(orthogonal)

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^K \mathbf{z}_i[j] \mathbf{u}_j$$

want to learn

$$\mathbf{z}_i[j] = (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$$

□ Given $K < D$, find $(\mathbf{u}_1, \dots, \mathbf{u}_K)$ minimizing reconstruction error:

$$\text{error}_K = \sum_{i=1}^N (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2$$

Reconstruction error and covariance matrix

The empirical covariance matrix

Plugging in the definition for $z_i[j]$

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

$$\text{error}_K = \sum_{i=1}^N \sum_{j=K+1}^D [\mathbf{u}_j \cdot (\mathbf{x}_i - \bar{\mathbf{x}})]^2$$

$$\sigma_{ml} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i[m] - \bar{\mathbf{x}}[m])(\mathbf{x}_i[l] - \bar{\mathbf{x}}[l])$$

$$= \sum_{i=1}^N \sum_{j=K+1}^D \mathbf{u}_j^T (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \mathbf{u}_j$$

For vectors a, b : $(a \cdot b)^2 = (a^T b)^2 = (a^T b)^T (a^T b) = b^T a a^T b$

$$= \sum_{j=K+1}^D \mathbf{u}_j^T \left[\sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \right] \mathbf{u}_j$$

$$= N \sum_{j=K+1}^D \mathbf{u}_j^T \Sigma \mathbf{u}_j$$

Minimizing reconstruction error and eigen vectors

- Minimizing reconstruction error equivalent to picking (ordered) orthonormal basis $(\mathbf{u}_1, \dots, \mathbf{u}_D)$ minimizing:

$$\text{error}_K = N \sum_{j=k+1}^D \mathbf{u}_j^T \Sigma \mathbf{u}_j$$

Sum of the (D-k) smallest eigenvalues of Σ

- Eigen vector:

$$\Sigma \mathbf{u} = \lambda \mathbf{u}$$

$\mathbf{u}^T \Sigma \mathbf{u} = \lambda \mathbf{u}^T \mathbf{u} = \lambda$

- Minimizing reconstruction error equivalent to picking $(\mathbf{u}_{K+1}, \dots, \mathbf{u}_D)$ to be eigen vectors with smallest eigen values

min error \Rightarrow throw out the (D-k) eigenvalues with smallest eigenvalues \Rightarrow Keep top K eigenvalues of Σ

Basic PCA algorithm

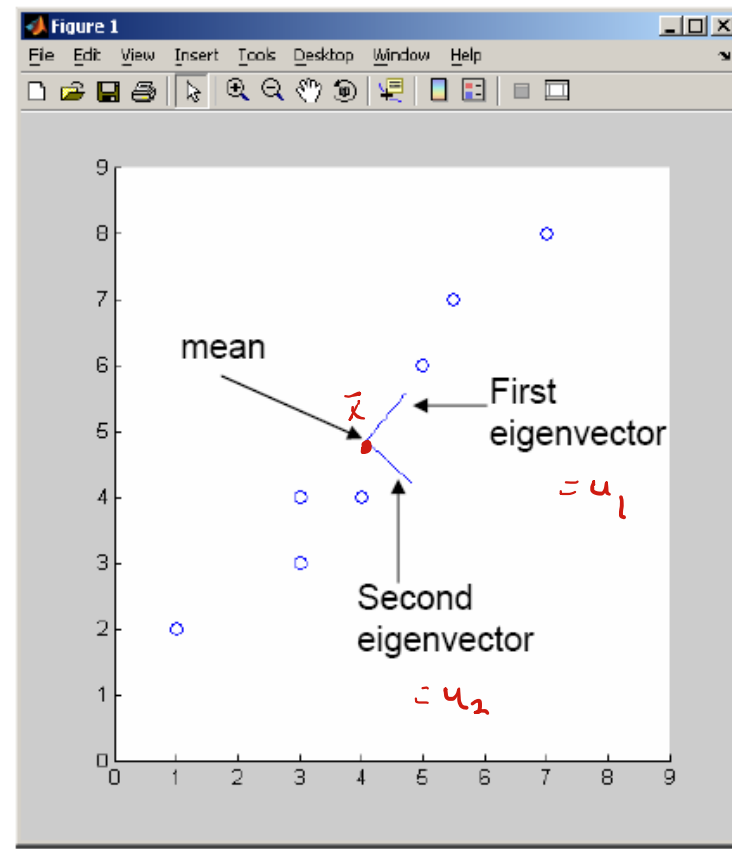
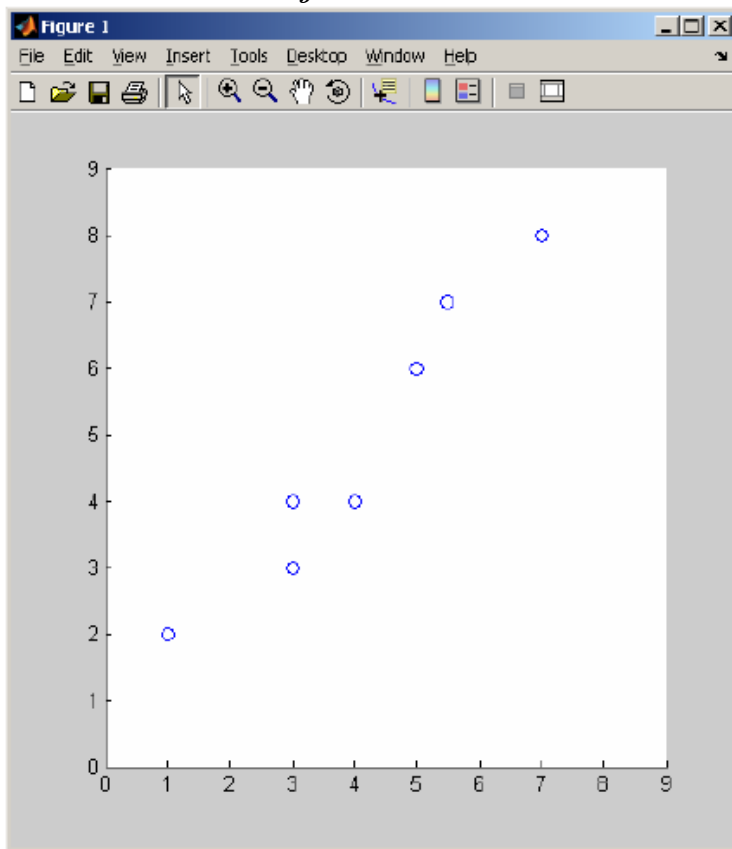
- Start from m by n data matrix \mathbf{X}
- **Recenter**: subtract mean from each row of \mathbf{X}
 - $\mathbf{X}_c \leftarrow \mathbf{X} - \bar{\mathbf{X}}$
- **Compute covariance matrix**:
 - $\Sigma \leftarrow 1/N \mathbf{X}_c^T \mathbf{X}_c$
- Find **eigen vectors and values** of Σ
- **Principal components**: k eigen vectors with highest eigen values

$$x_i \begin{pmatrix} x \\ \dots \\ \dots \end{pmatrix}$$

numpy.linalg.eig

PCA example

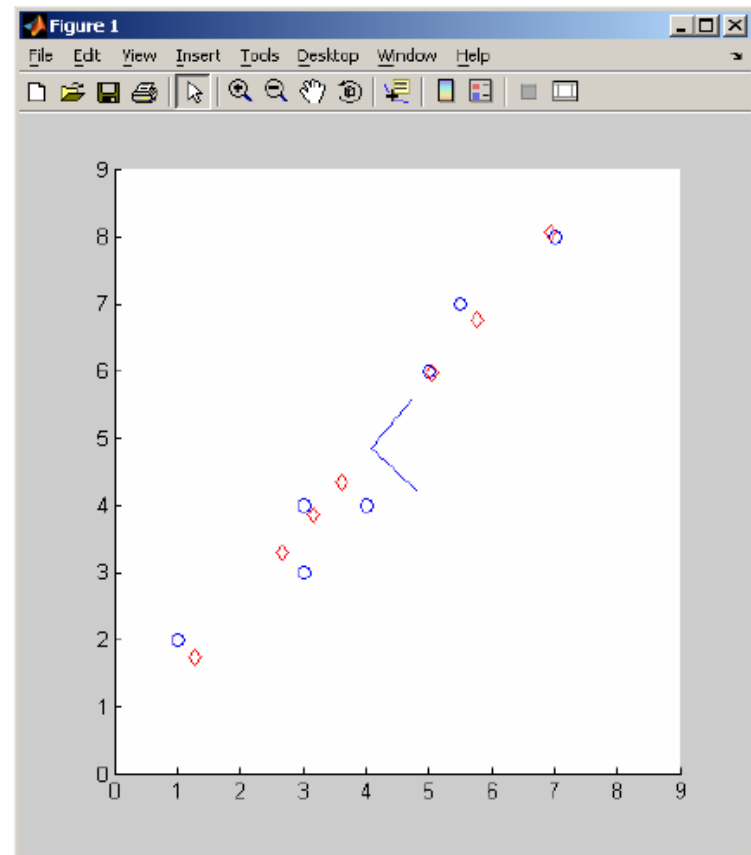
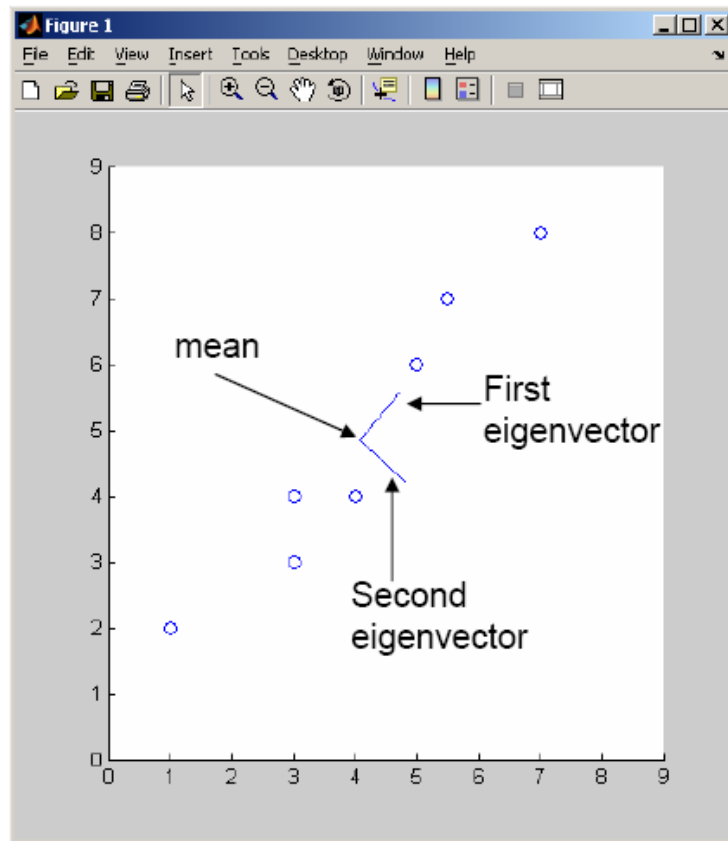
$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^K \mathbf{z}_i[j] \mathbf{u}_j$$



PCA example – reconstruction

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^K \mathbf{z}_i[j] \mathbf{u}_j$$

only used first principal component



Eigenfaces [Turk, Pentland '91]

Input images:



Principal components:



Avg. face
 \bar{x}

glasses

long hair

Eigenfaces reconstruction

- Each image corresponds to adding 8 principal components:



↑
pretty good reconstruction
with $K=8$.

Scaling up

- Covariance matrix can be really big!
 - Σ is D by D
 - Say, only 10000 features
 - finding eigenvectors is very slow...
- Use singular value decomposition (SVD)
 - finds to K eigenvectors
 - great implementations available, e.g., `scipy.linalg.svd`

What you need to know

- Dimensionality reduction
 - why and when it's important
- Simple feature selection
- Principal component analysis
 - minimizing reconstruction error
 - relationship to covariance matrix and eigenvectors
 - using SVD