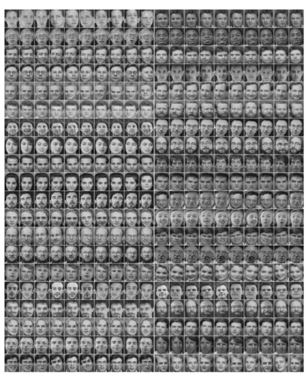
Dimensionality Reduction
Principal Component Analysis (PCA)

CS229: Machine Learning
Carlos Guestrin
Stanford University
Slides include content developed by and co-developed with Emily Fox

Embedding

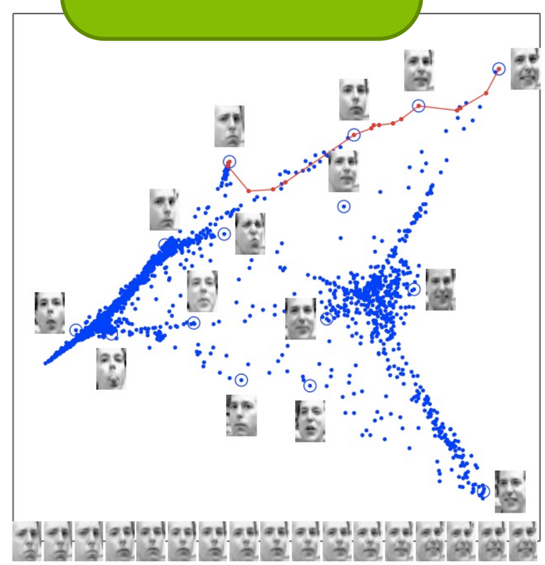
Example: Embedding images to visualize data

Data → PCA → Intelligence



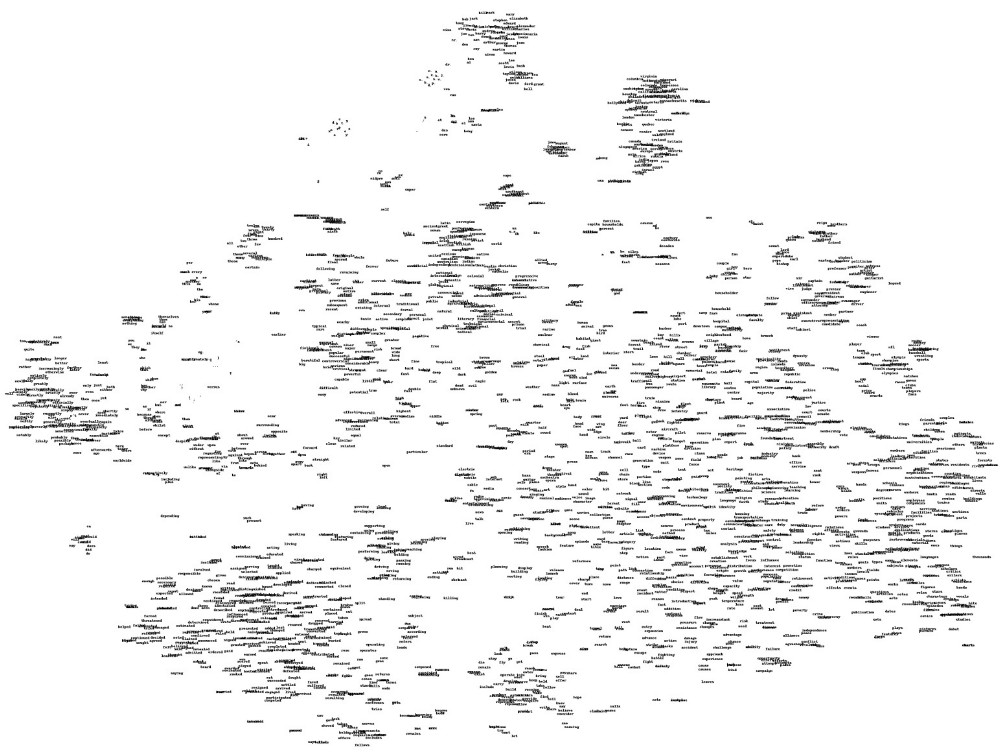
Images with thousands or millions of pixels

Can we give each image a coordinate, such that similar images are near each other?



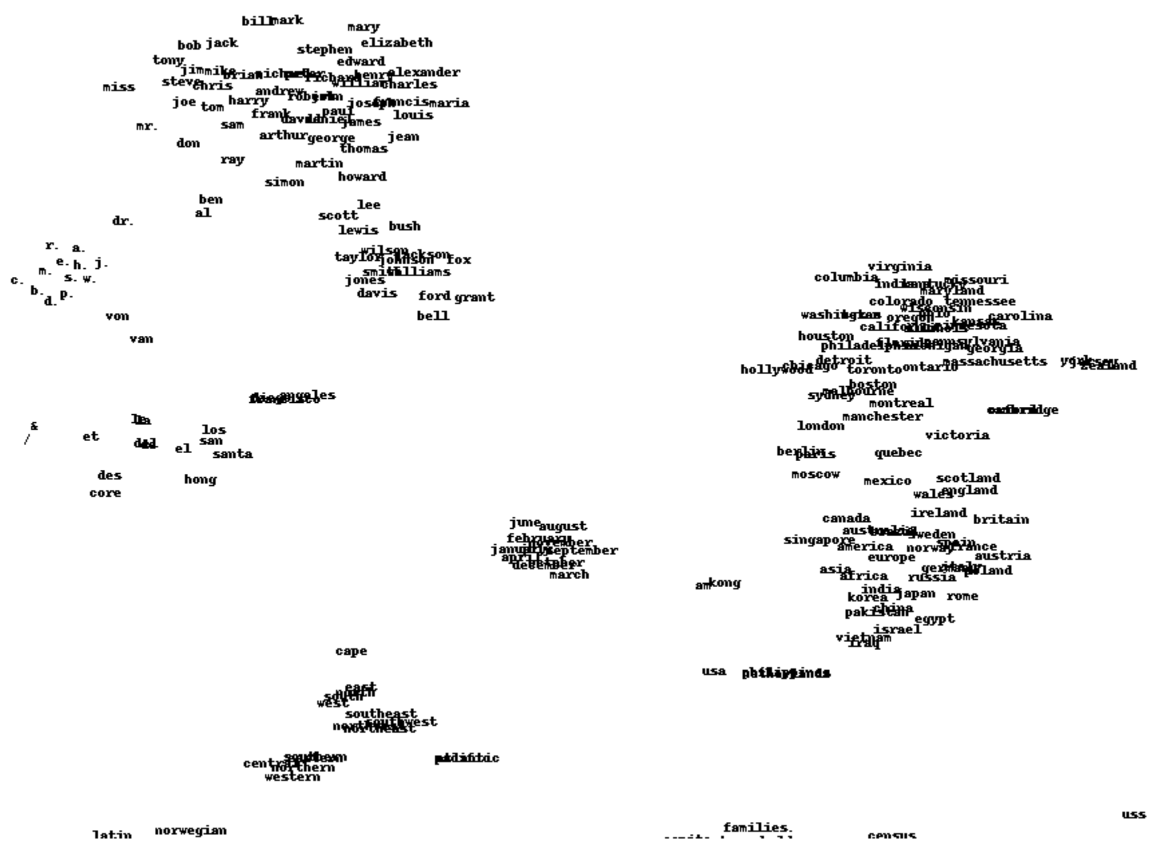
[Saul & Roweis '03]

Embedding words



[Joseph Turian 2008]

Embedding words (zoom in)



[Joseph Turian 2008]

Dimensionality reduction

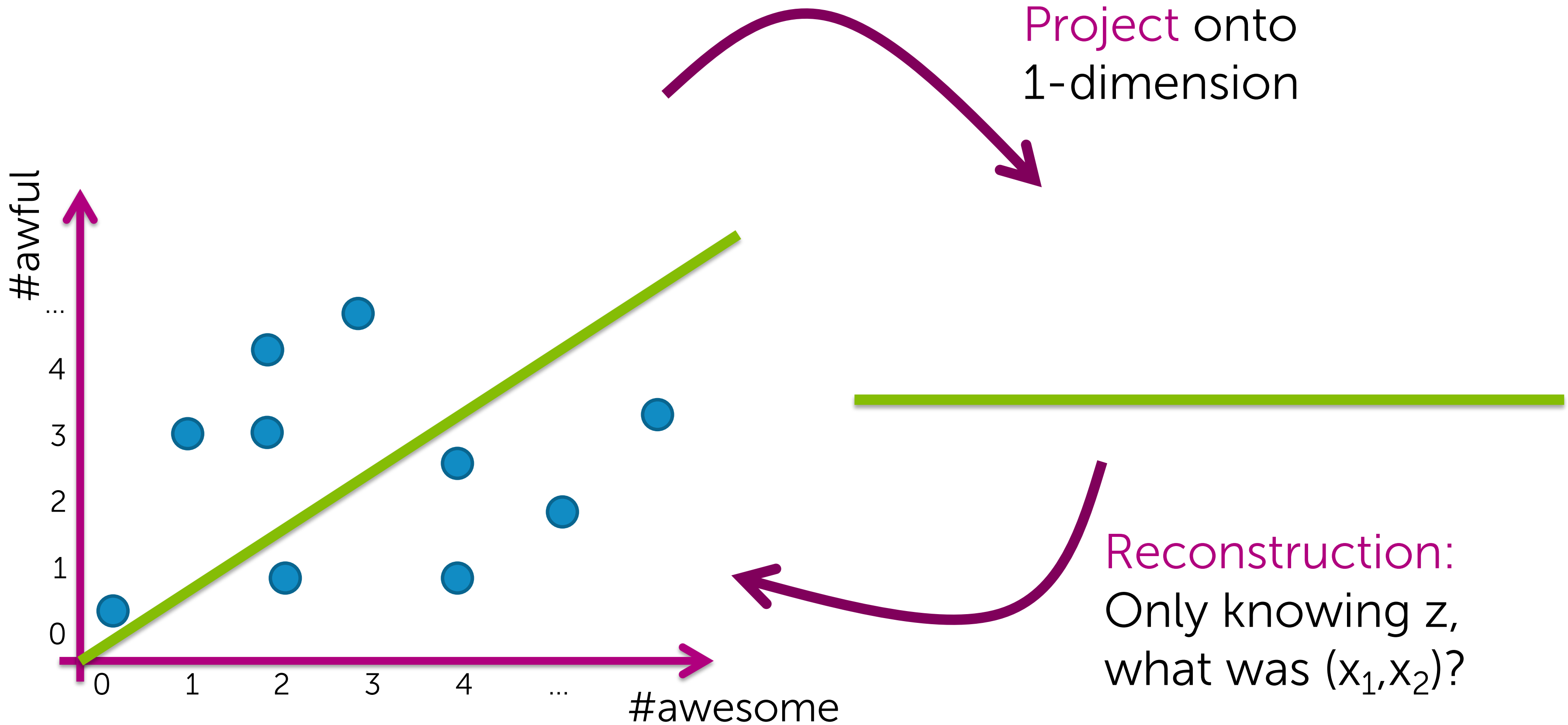
- Input data may have thousands or millions of dimensions!
  - e.g., text data
- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D or 4D
  - discover "intrinsic dimensionality" of data
    - high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

Lower dimensional projections

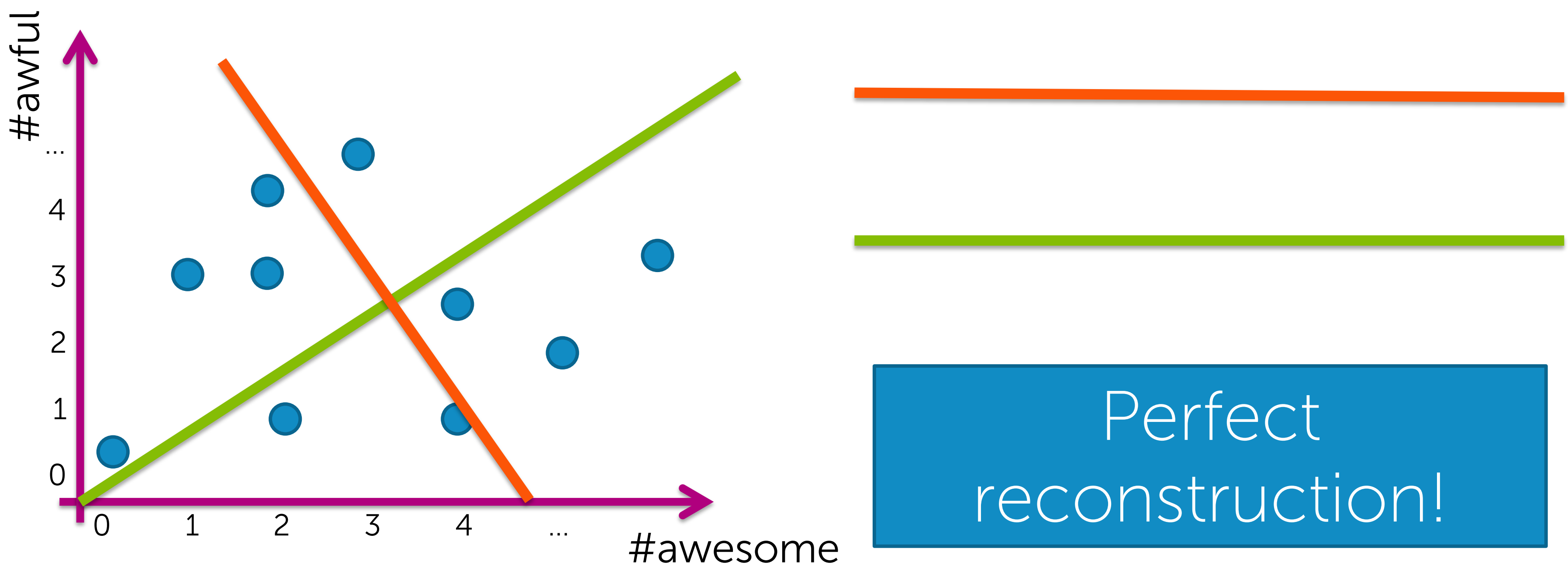
- Rather than picking a subset of the features, we can create new features that are combinations of existing features

- Let's see this in the unsupervised setting
  - just **x**, but no y

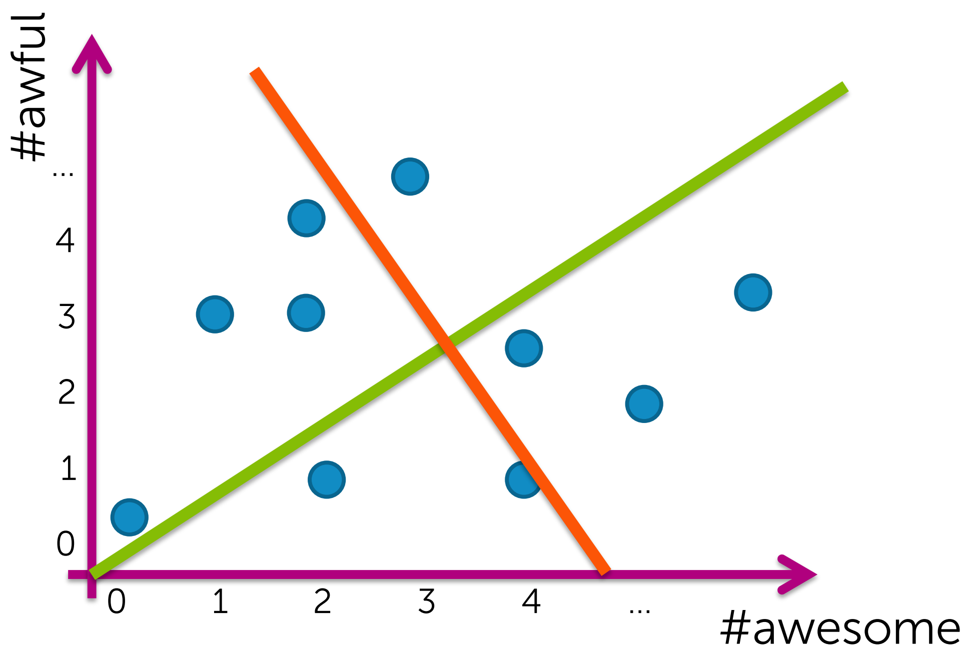
Linear projection and reconstruction



What if we project onto d vectors?



If I had to choose one of these vectors, which do I prefer?



Principal component analysis (PCA) – Basic idea

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

- Choose projection with minimum reconstruction error

“PCA explained visually”

http://setosa.io/ev/principal-component-analysis/

Linear projections, a review

- Project a point into a (lower dimensional) space:
  - **point**: $\mathbf{x} = (x_1,\ldots,x_d)$
  - **select a basis** – set of basis vectors – $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$
    - we consider orthonormal basis:
      - $\mathbf{u}_i \bullet \mathbf{u}_i = 1$, and $\mathbf{u}_i \bullet \mathbf{u}_j = 0$ for $i \neq j$
  - **select a center** – $\bar{\mathbf{x}}$, defines offset of space
  - **best coordinates** in lower dimensional space defined by dot-products: $(z_1,\ldots,z_k)$, $z_i = (\bar{\mathbf{x}}-\mathbf{x}) \bullet \mathbf{u}_i$
    - minimum squared error

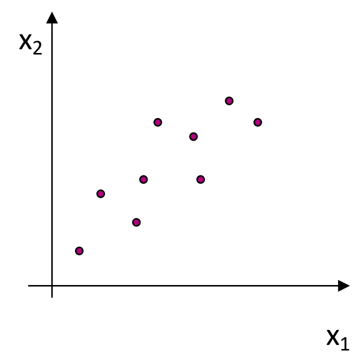
PCA finds projection that minimizes reconstruction error

- Given N data points: $\mathbf{x}^i = (x_1^i,\ldots,x_d^i)$, i=1…N
- Will represent each point as a projection:

$$\widehat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j \quad \text{and} \quad \bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^i \qquad z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$$

- PCA:
  - Given k<<d, find $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$ minimizing reconstruction error:

$$error_k = \sum_{i=1}^{N} (\mathbf{x}^i - \widehat{\mathbf{x}}^i)^2$$



Understanding the reconstruction error

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

$$\mathbf{x}^i = \bar{\mathbf{x}} + \sum_{j=1}^{d} z_j^i \mathbf{u}_j$$

- Rewriting error:

$$\widehat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

$$z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$$

- Given k<<d, find ($\mathbf{u}_1$,…,$\mathbf{u}_k$) minimizing reconstruction error:

$$error_k = \sum_{i=1}^{N} (\mathbf{x}^i - \widehat{\mathbf{x}}^i)^2$$

Reconstruction error and covariance matrix

$$error_k = \sum_{i=1}^{N} \sum_{j=k+1}^{d} [\mathbf{u}_j \cdot (\mathbf{x}^i - \bar{\mathbf{x}})]^2$$

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

Minimizing reconstruction error and eigen vectors

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

$$error_k = N \sum_{j=k+1}^{d} \mathbf{u}_j^T \Sigma \mathbf{u}_j$$

- Eigen vector:

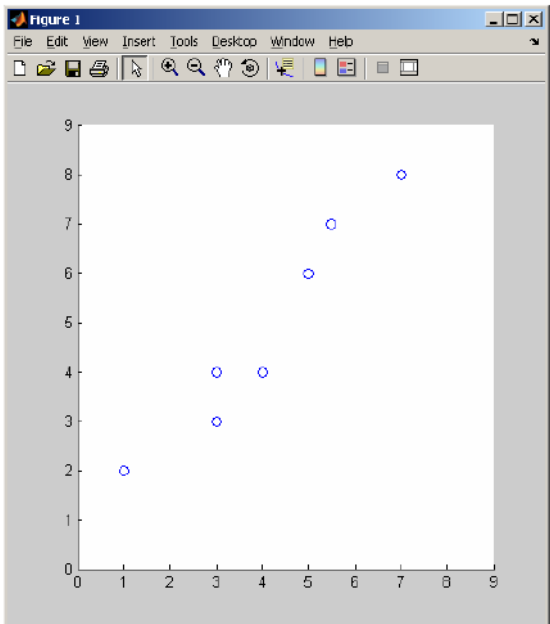
- Minimizing reconstruction error equivalent to picking ($\mathbf{u}_{k+1},\ldots,\mathbf{u}_d$) to be eigen vectors with smallest eigen values

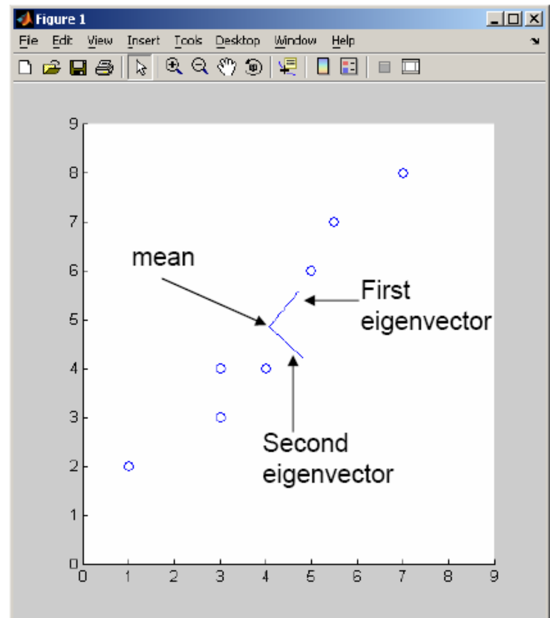
Basic PCA algoritm

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

PCA example

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

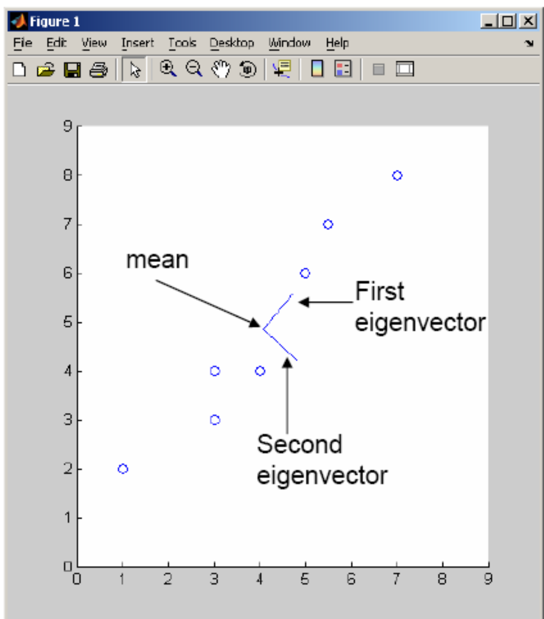


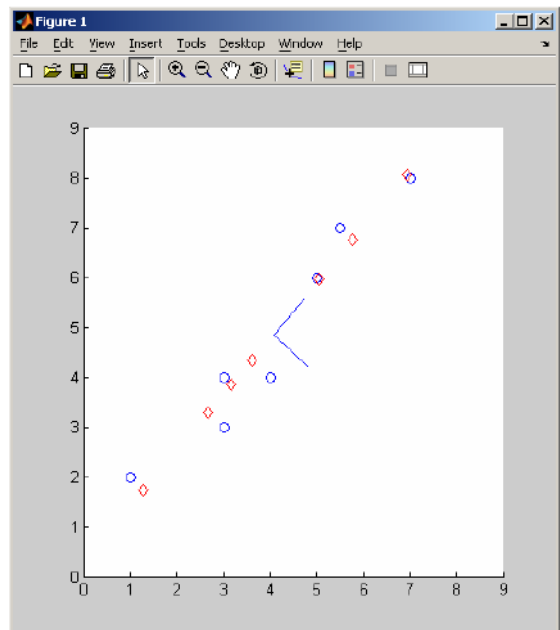


PCA example – reconstruction

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

only used first principal component

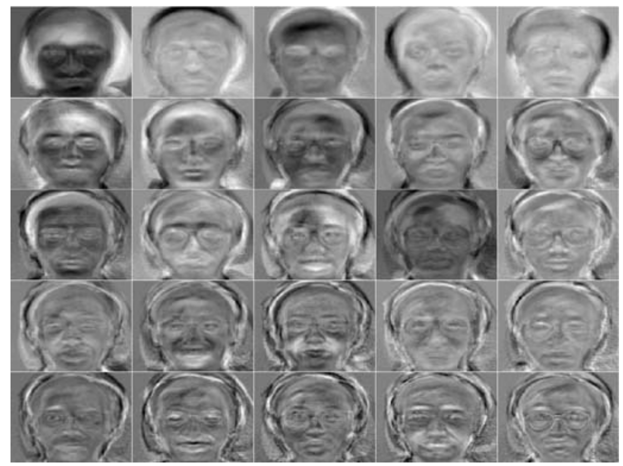




Eigenfaces [Turk, Pentland ’91]

- Input

- Principal components:



Eigenfaces reconstruction

- Each image corresponds to adding 8 principal components:

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., python, R, Matlab svd

SVD

- Write $X = W S V^T$
  - $X \leftarrow$ data matrix, one row per datapoint
  - $W \leftarrow$ weight matrix, one row per datapoint – coordinate of $\mathbf{x}^i$ in eigenspace
  - $S \leftarrow$ singular value matrix, diagonal matrix
    - in our setting each entry is eigenvalue λ_j
  - $V^T \leftarrow$ singular vector matrix
    - in our setting each row is eigenvector $\mathbf{v}_j$

PCA using SVD algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $X_c \leftarrow X - \bar{X}$
- Call SVD algorithm on X_c – ask for k singular vectors
- **Principal components:** k singular vectors with highest singular values (rows of V^T)
  - **Coefficients** become:

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