EE3731C Pattern Recognition 1

BT Thomas Yeo

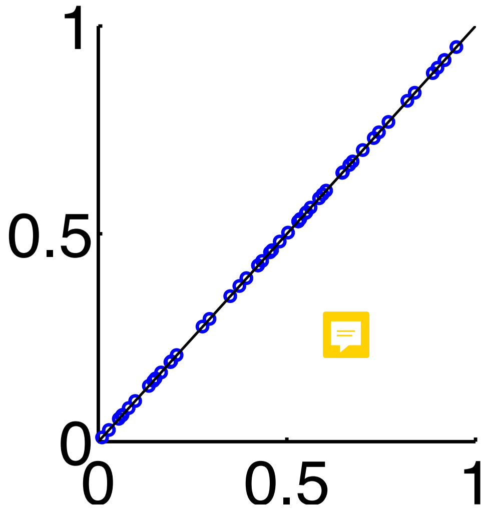
ECE, CIRC, Sinapse, Duke-NUS, HMS

Curse of Dimensionality

- Data dimensionality = number of "free" parameters in data / model
  - $1M$-pixel image $\implies$ $1M$ dimensions
  - Text document has N words $\implies$ N dimensions
- Dimensionality depends on problem definition
  - Word analysis of text document $\implies$ dimensionality = # words
  - Character analysis of text document $\implies$ dimensionality = # characters
- Curse of dimensionality: pattern recognition harder for big dimensions
  - Joint distribution of binary variables $p(x_1, \cdots, x_N)$ has $2^N - 1$ parameters
  - Need a lot of data to estimate so many parameters

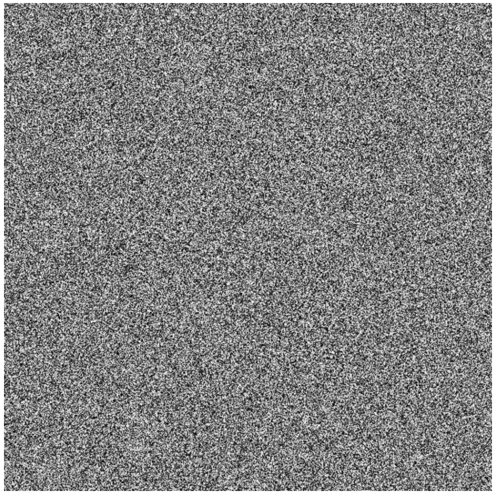
Intrinsic Dimensions

- Given data has D dimensions, hope "intrinsic" data dimensions d less than D
- "Random" images do not look like "real" images $\implies$ image pixels not independent $\implies$ true dimensions less than number of pixels



2-dimensional Points (D = 2)
with 1"intrinsic" dimension (d = 1)

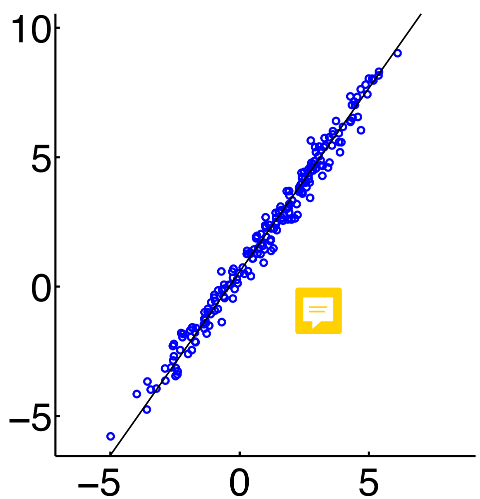
"Real" Image



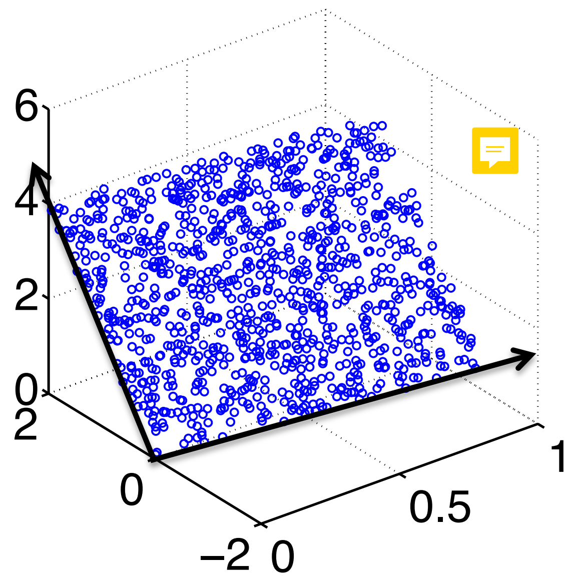
"Random" Image
(Each pixel independent with uniform U[0, 1] distribution)

Dimensionality Reduction

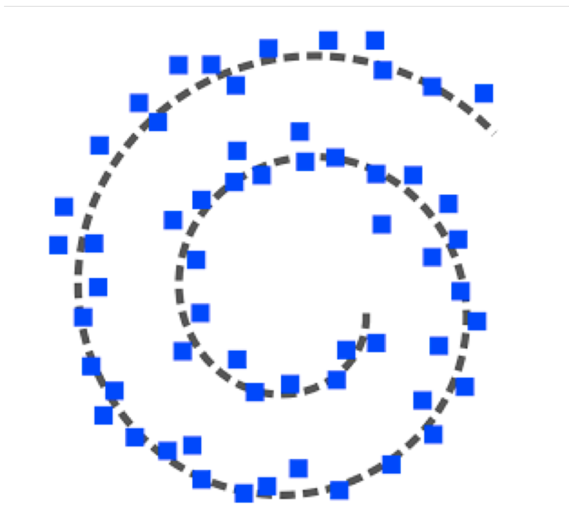
- Dimensionality reduction: map high-dimensional data into lower dimensions
  - Also known as feature extraction or feature reduction
  - Different assumptions lead to different mapping
- Principal component analysis (PCA) assumes data lies on linear subspace



1-dimensional linear subspace
(i.e., can be represented by straight line)



2-dimensional linear subspace
(i.e., can be represented by flat plane)



1-dimensional nonlinear space
representable by 1-D curve
(PCA cannot handle this)

spiral image from http://edcabalfin.blogspot.sg/2011/04/manifold-learning-part-3.html

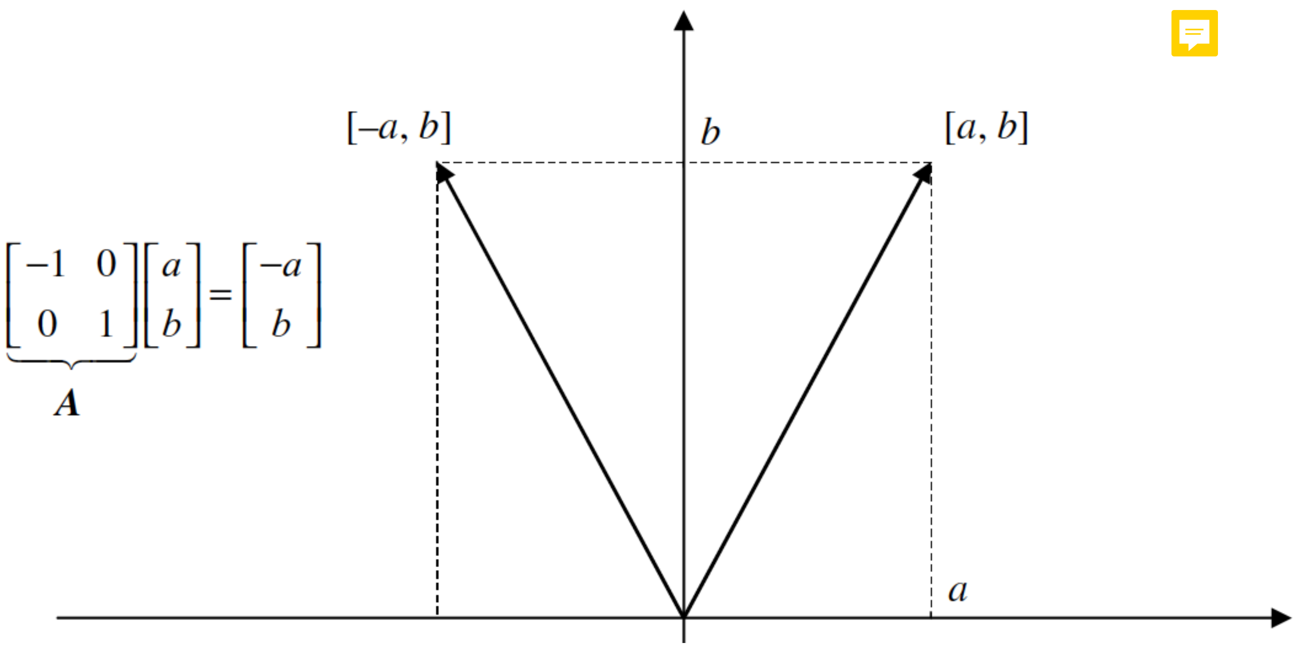
Linear System Revision

Linear System (or Transformation)

- Suppose system T maps input $x \in \mathbb{R}^n$ to output $y \in \mathbb{R}^m$
  - If system is linear, then $y_1 = T(x_1)$ and $y_2 = T(x_2)$ implies $T(ax_1 + bx_2) = aT(x_1) + bT(x_2) = ay_1 + by_2$
  - All linear systems can be written in matrix format, i.e., $y = Ax$ for some $m \times n$ matrix
- Example: $Ax = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \\ a_{31}x_1 + a_{32}x_2 \end{bmatrix}$
- Example: $Ax = [\vec{a}_1 \ \ \vec{a}_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1\vec{a}_1 + x_2\vec{a}_2$, where $\vec{a}_i$ is i-th column of A
  - Output is linear combination of columns of A

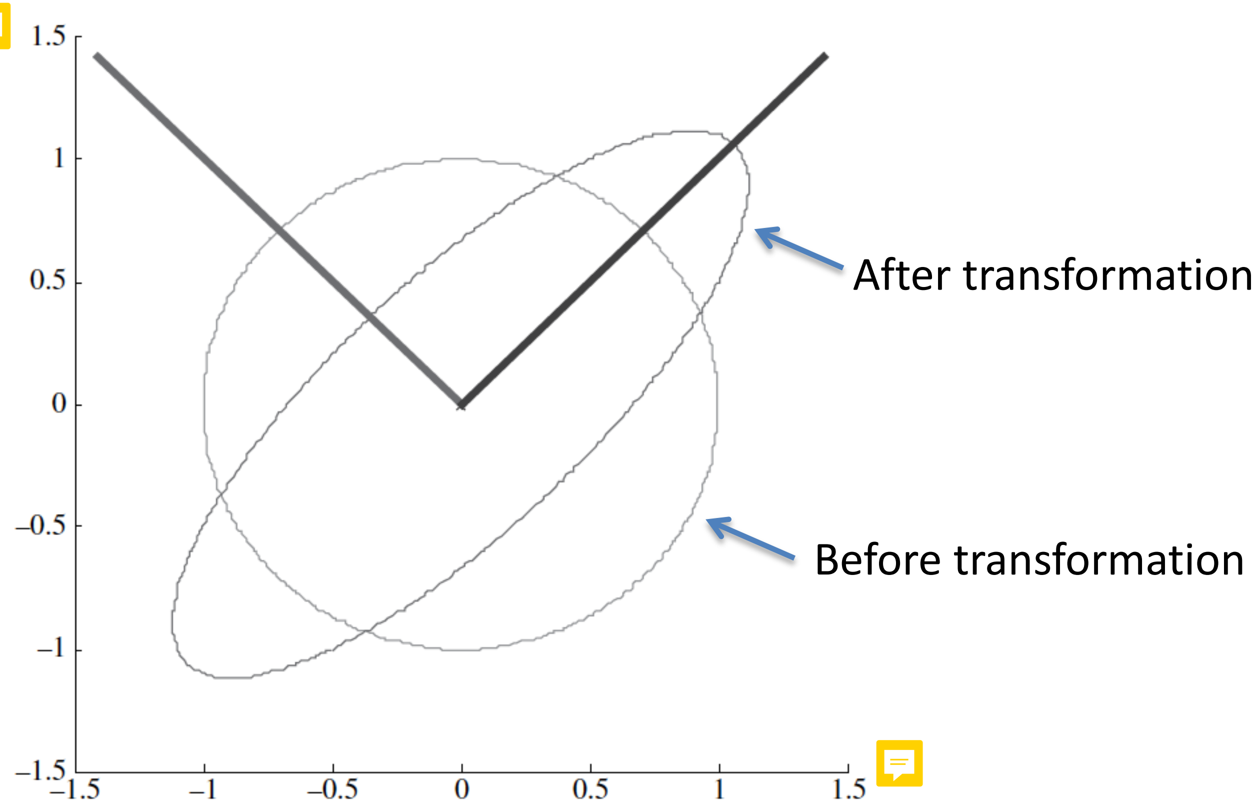
Linear Transformation: Example 1

- $A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ flips vector about vertical axis



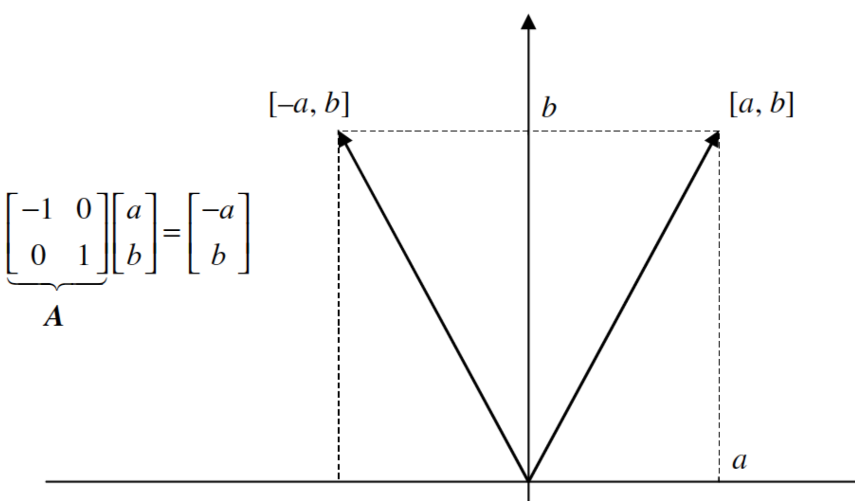
Linear Transformation: Example 2

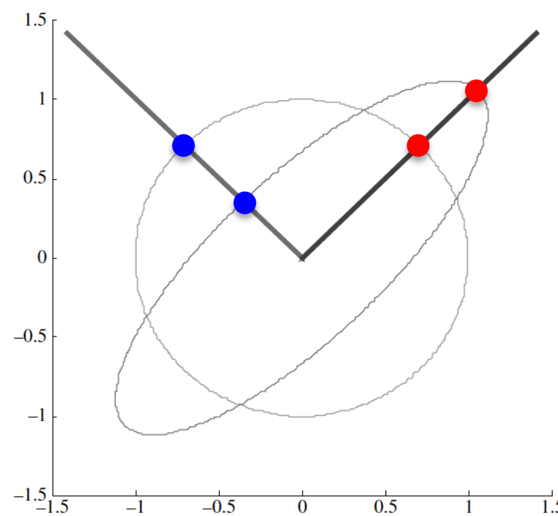
- $A = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \implies \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 + 0.5x_2 \\ 0.5x_1 + x_2 \end{bmatrix}$
- Circle of points (below) mapped by A onto an ellipse (below)



Eigenvectors & Eigenvalues

- Consider linear system $y = Ax$, where $A = M \times M$ matrix
- Suppose x is eigenvector of A
  - By definition of eigenvectors, $Ax = \lambda x$ (for some number λ)
  - Therefore direction of x preserved under transformation A





$x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ is eigenvector with $\lambda = 1$

$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ & $\begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ eigenvectors with eigenvalues 1.5 and 0.5

Computing Eigenvectors/Eigenvalues

- $u \neq 0$ is an eigenvector of $M \times M$ matrix A if $Au = \lambda u$ for some λ

$$
\begin{aligned}
Au &= \lambda u \\
(A - \lambda I)u &= 0 \\
\det(A - \lambda I) &= 0
\end{aligned}
$$

u is non-zero => A – λI is not full rank

- For example, $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, then

$$
\begin{aligned}
\det \begin{pmatrix} a - \lambda & b \\ c & d - \lambda \end{pmatrix} &= 0 \\
(a - \lambda)(d - \lambda) - bc &= 0 \\
\lambda^2 - (a + d)\lambda + (ad - bc) &= 0
\end{aligned}
$$

  - Quadratic equations $\Longrightarrow$ up to two λs (in general up to M eigenvalues for $M \times M$ matrix A)
  - Solve for λs, and then eigenvectors

- In real applications, use software to solve (e.g., ‘eig’ function in matlab)

Matrix Eigen-Decomposition

- For $M \times M$ matrix A with M eigenvalues $\lambda_1, \cdots, \lambda_M$ and corresponding eigenvectors $u_1, \cdots, u_M$, where $u_m^T u_m = 1$ (unit length)

$$A[u_1 \;\; u_2 \;\; \cdots \;\; u_M] = [\lambda_1 u_1 \;\; \lambda_2 u_2 \;\; \cdots \;\; \lambda_M u_M]$$

$$= [u_1 \;\; u_2 \;\; \cdots \;\; u_M] \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_M \end{bmatrix}$$

$$AU = U\Lambda,$$

where Λ is a diagonal matrix which is all 0 except the diagonals which correspond to eigenvalues and U is a matrix whose columns are eigenvectors

- A symmetric $\implies$ M (not necessarily unique) eigenvalues, and M orthogonal eigenvectors, i.e., $u_i^T u_j = \delta(i-j) \implies U^{-1} = U^T$

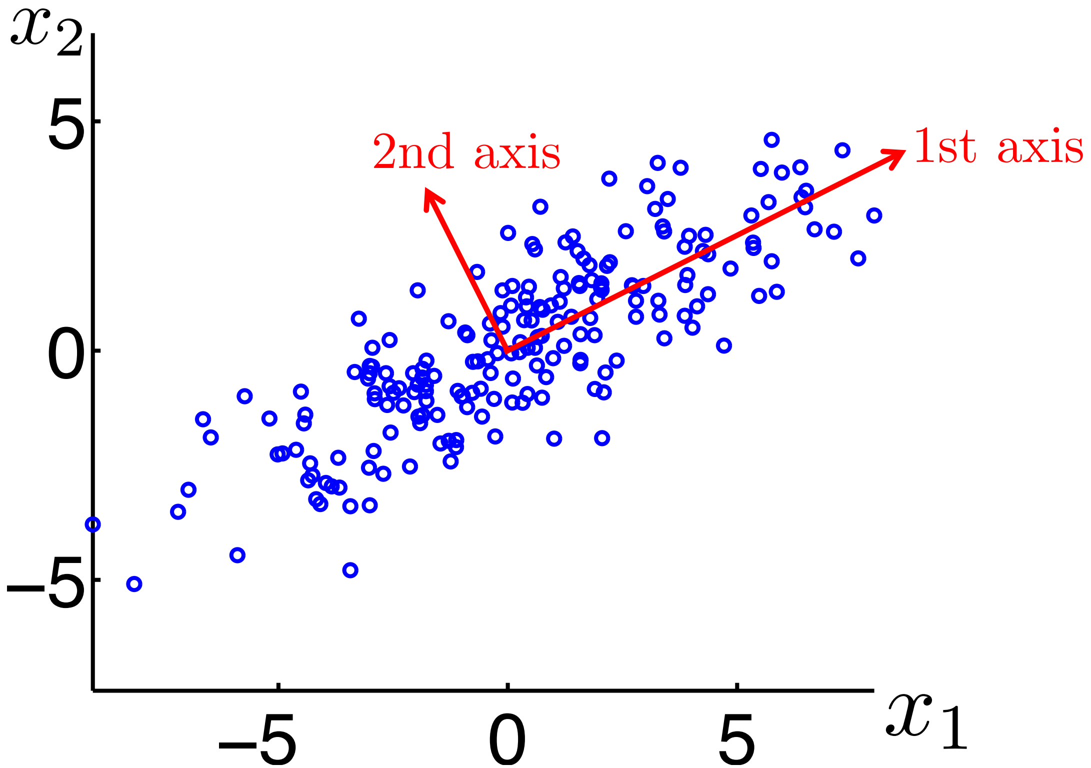
$$AU = U\Lambda \implies A = U\Lambda U^T = \lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T \cdots + \lambda_M u_M u_M^T = \sum_m \lambda_m u_m u_m^T$$

- A positive semidefinite, i.e., $x^T A x \geq 0$ for all $x \implies \lambda_m \geq 0$ for all m.

Principal Component Analysis (PCA)

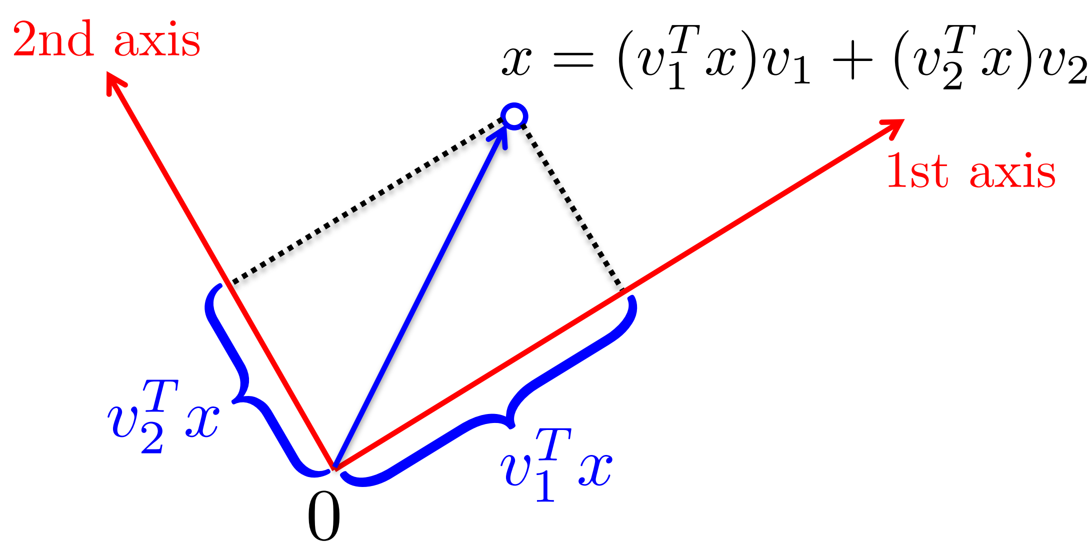
Principal Component Analysis (PCA)

- PCA = new coordinate system with orthogonal axes
  - Assume mean of input data = $\vec{0}$
  - 1st axis direction of largest variability in data
  - 2nd axis direction of largest variability orthogonal to 1st axis
  - 3rd axis direction of largest variability orthogonal to 1st and 2nd axes
  - ...



Projecting Onto Principal Axes

- Coordinates with respective to new axes given by projection onto axes
  - For D-dimensional point x, new coordinates $= (v_1^T x, v_2^T x, \cdots, v_K^T x)$, where v_i is unit vector in same direction as i-th axis
  - x now K-dimensional vector: we have achieved dimensionality reduction for $K < D$
  - $v_i^T x$ called i-th principal component (PC)



1st Principal Axis is Direction of Largest Variability

- Consider $x^{(1)}, \cdots, x^{(N)}$ (N data points, each dimension D)
- Assume data centered at origin, i.e., $\frac{1}{N}\sum_n x^{(n)} = \vec{0}$
- Let v_1 = unit vector in same direction as 1st principal axis, so resulting projected data are $[v_1^T x^{(1)}, \cdots, v_1^T x^{(N)}]$
- Note that projected data has $\frac{1}{N}\sum_n v_1^T x^{(n)} = 0$ mean, so (empirical) variance of projected data are

$$\frac{1}{N}\sum_{n=1}^{N}\left(v_1^T x^{(n)}\right)^2 = \frac{1}{N}\sum_{n=1}^{N} v_1^T x^{(n)} x^{(n)T} v_1 = v_1^T \left(\frac{1}{N}\sum_{n=1}^{N} x^{(n)} x^{(n)T}\right) v_1 = v_1^T \Sigma v_1$$

where Σ is covariance of data samples

  - $\Sigma = \frac{1}{N} X X^T$, where X is the $D \times N$ matrix $[x^{(1)}, \cdots, x^{(N)}]$

- Therefore 1st principal axis (direction of data with largest variance):

$$v_1 = \operatorname*{argmax}_v v^T \Sigma v \quad \text{where} \quad v^T v = 1$$

Finding Principal Axes

- 1st principal axis (direction of data with largest variance):

$$v_1 = \underset{v}{\operatorname{argmax}}\, v^T \Sigma v \quad \text{where} \quad v^T v = 1$$

- Σ is symmetric & positive semidefinite, so has D eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$ with eigenvectors $u_1, \cdots, u_D$, where $u_i^T u_j = \delta(i-j)$:

$$\Sigma = \lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T + \cdots + \lambda_D u_D u_D^T$$

- Therefore

$$\begin{aligned} v^T \Sigma v &= v^T \left(\lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T + \cdots + \lambda_D u_D u_D^T\right) v \\ &= \lambda_1 (v^T u_1)^2 + \lambda_2 (v^T u_2)^2 + \cdots + \lambda_D (v^T u_D)^2 \end{aligned}$$

- Since λ_1 largest and $v_1^T v_1 = 1$, therefore $v_1 = u_1$
  - $v_2 = \operatorname{argmax}_v v^T \Sigma v \quad$ where $\quad v^T v = 1$ and $v_1^T v_2 = 0 \implies v_2 = u_2$
  - In general, $v_n = u_n$

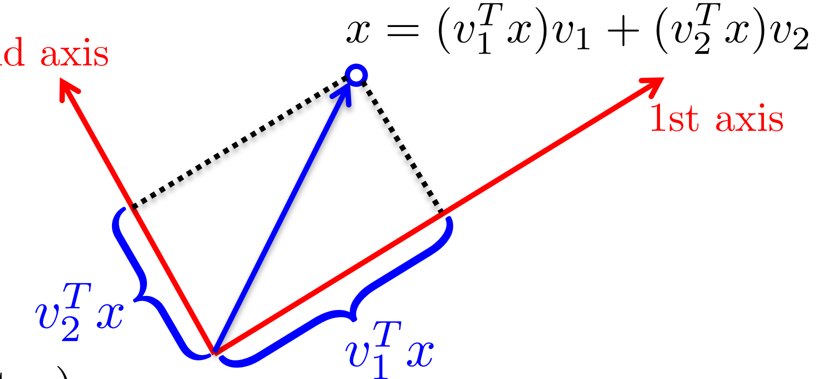
Principal Component Analysis (PCA)

- Consider $x^{(1)}, \cdots, x^{(N)}$ (N data points, each dimension D)
- Compute mean $\mu = \frac{1}{N}\sum_n x^{(n)}$
- Subtract mean from each sample point $\bar{x}^{(n)} = x^{(n)} - \mu$ (this ensures resulting data are centered at the origin) – procedure called "demean"
- Form $D \times N$ matrix $X = [\bar{x}^{(1)}, \cdots, \bar{x}^{(N)}]$
- Compute sample covariance matrix $\Sigma = \frac{1}{N} X X^T$
- Find top K eigenvalues and corresponding eigenvectors $v_1, \cdots, v_K$
- Given new (or old) datapoint x, we can represent x by new coordinates $[v_1^T(x-\mu) \;\; v_2^T(x-\mu) \;\; \cdots \;\; v_K^T(x-\mu)]$
- Why top K eigenvalues and not bottom K?

Principal Component Analysis (PCA) Representation

What does PCA representation buy us?

- Consider $x^{(1)}, \cdots, x^{(N)}$ (N data points, each dimension D) with 0 mean
- Perform PCA and represent old datapoint $x^{(n)}$ by $\left[v_1^T x^{(n)} \;\; v_2^T x^{(n)} \;\; \cdots \;\; v_K^T x^{(n)}\right]$
- Can "reconstruct" $x^{(n)}$ by $\hat{x}^{(n)} = \sum_{k=1}^{K} (v_k^T x^{(n)}) v_k$
- Consider reconstruction error



$$||x^{(n)} - \hat{x}^{(n)}||^2 \triangleq \sum_{i=1}^{D} (x_i^{(n)} - \hat{x}_i^{(n)})^2$$

- Total reconstruction error (proof at end of notes):

$$\sum_{n=1}^{N} ||x^{(n)} - \hat{x}^{(n)}||^2 = \sum_{n=1}^{N} x^{(n)T} x^{(n)} - N \sum_{i=1}^{K} v_i^T \Sigma v_i$$

- First term in reconstruction error always positive $\implies$ to minimize reconstruction error, want second term $\sum_{i=1}^{K} v_i^T \Sigma v_i$ as big as possible, which is how PCA was defined $\implies$ PCA lets us represent our original data with the least amount of error with K coefficients

Remarks

- Previous argument still works when $x^{(1)}, \cdots, x^{(N)}$ has non-zero mean μ
- Just need to subtract μ from data before PCA
- When projecting datapoint x onto principal axes, new coordinates are $[v_1^T(x-\mu), v_2^T(x-\mu), \cdots, v_K^T(x-\mu)]$
- Reconstructed $\hat{x} = \mu + \sum_{k=1}^{K} \left(v_k^T(x-\mu)\right) v_k$
- No magic way to select K. One way is the following criterion

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{D} \lambda_i} \geq \text{threshold (e.g., 0.95)}$$

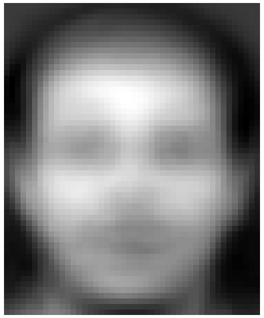
- For $M \times M$ images, can convert into a vector (i.e., $D = M^2$) by stacking columns of pixel values into one single column

Eigenfaces Example

Eigenfaces

- PCA on 200 face images

16 of 200 faces

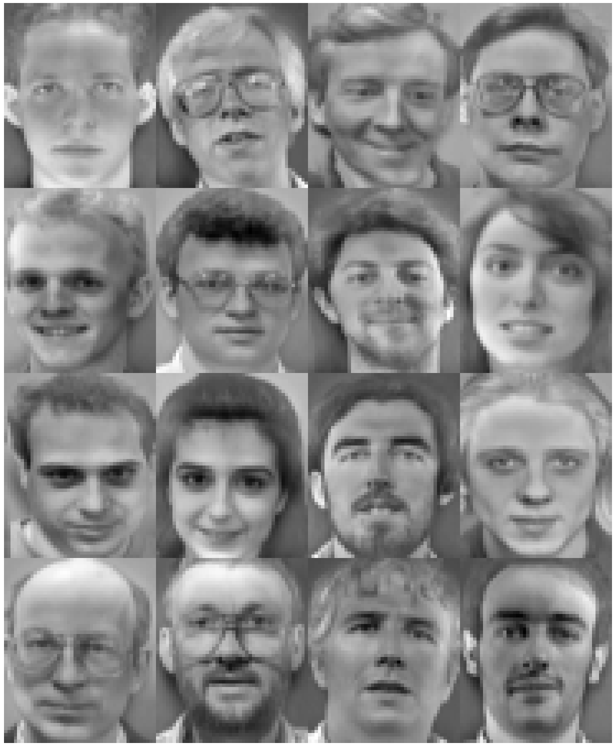


Mean of 200 faces

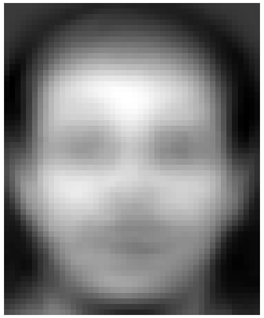
See eigenfaces.m on IVLE

Eigenfaces

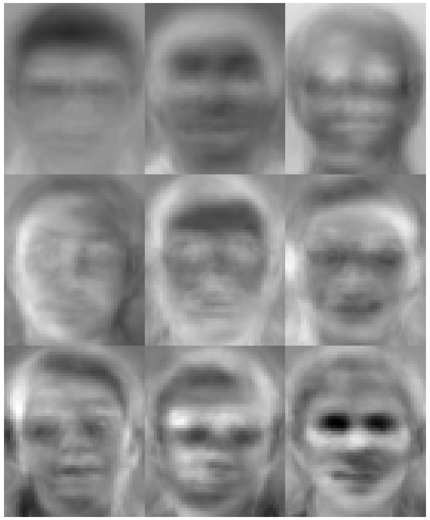
- PCA on 200 face images



16 of 200 faces
after de-meaning



Mean of 200 faces
(origin of coordinate
System)



Top 9 Eigenvectors
(Eigenfaces)

See eigenfaces.m on IVLE

Eigenfaces

- PCA on 200 face images

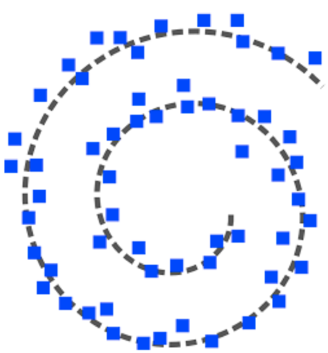
16 of 200 faces

Reconstructed Faces
From Top 50 eigenfaces

See eigenfaces.m on IVLE

Summary

- Curse of dimensionality: pattern recognition harder for high dimensions $\Longrightarrow$ useful to reduce dimensions
- PCA finds orthonormal basis (coordinate system) for data
- Principal axes corresponds to eigenvectors of data covariance matrix
- Principal axes sorted in order of importance (based on eigenvalues)
- Can discard axes with lower eigenvalues (hoping they are noise or unimportant for application)
- Reduce data dimensions by projecting onto remaining axes
- PCA optimal in the sense that it minimizes reconstruction error as measured by Euclidean distance
- PCA does not work for nonlinear space



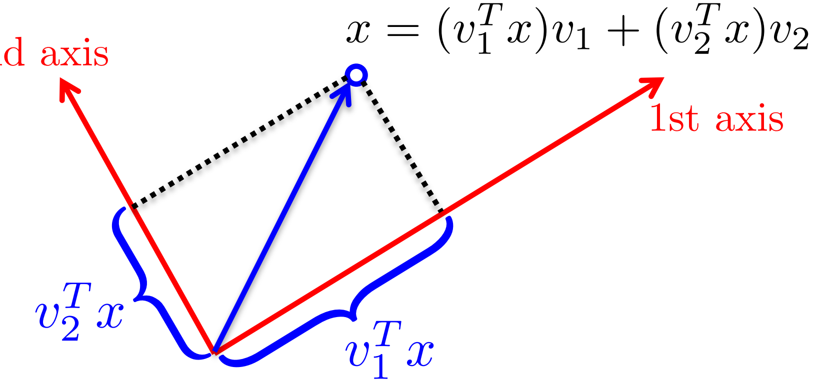
Optional Reading

- Duda and Hart: Pattern Recognition, Chapter 3.8.1

Additional Material: PCA minimizes reconstruction error proof

What does PCA representation buy us?

- Consider $x^{(1)}, \cdots, x^{(N)}$ (N data points, each dimension D) with 0 mean
- Perform PCA and represent old datapoint $x^{(n)}$ by $\begin{bmatrix} v_1^T x^{(n)} & v_2^T x^{(n)} & \cdots & v_K^T x^{(n)} \end{bmatrix}$
- Can "reconstruct" $x^{(n)}$ by $\hat{x}^{(n)} = \sum_{k=1}^{K} (v_k^T x^{(n)}) v_k$
- Consider reconstruction error



$$
\begin{aligned}
||x^{(n)} - \hat{x}^{(n)}||^2 &\triangleq \sum_{i=1}^{D} (x_i^{(n)} - \hat{x}_i^{(n)})^2 \\
&= (x^{(n)} - \hat{x}^{(n)})^T (x^{(n)} - \hat{x}^{(n)}) \\
&= \left(x^{(n)} - \sum_{k=1}^{K} (v_k^T x^{(n)}) v_k \right)^T \left(x^{(n)} - \sum_{k=1}^{K} (v_k^T x^{(n)}) v_k \right) \\
&= x^{(n)T} x^{(n)} - 2 \sum_{k=1}^{K} (v_k^T x^{(n)})^2 + \sum_{k=1}^{K} (v_k^T x^{(n)})^2 \\
&= x^{(n)T} x^{(n)} - \sum_{k=1}^{K} (v_k^T x^{(n)})^2
\end{aligned}
$$

What does PCA representation buy us?

- From previous slide: $||x^{(n)} - \hat{x}^{(n)}||^2 = x^{(n)T} x^{(n)} - \sum_{i=1}^{K} (v_i^T x^{(n)})^2$
- Total reconstruction error:

$$\begin{aligned}
\sum_{n=1}^{N} ||x^{(n)} - \hat{x}^{(n)}||^2 &= \sum_{n=1}^{N} x^{(n)T} x^{(n)} - \sum_{n=1}^{N} \sum_{i=1}^{K} (v_i^T x^{(n)})^2 \\
&= \sum_{n=1}^{N} x^{(n)T} x^{(n)} - \sum_{n=1}^{N} \sum_{i=1}^{K} v_i^T x^{(n)} x^{(n)T} v_i \\
&= \sum_{n=1}^{N} x^{(n)T} x^{(n)} - \sum_{i=1}^{K} v_i^T \left(\sum_{n=1}^{N} x^{(n)} x^{(n)T} \right) v_i \\
&= \sum_{n=1}^{N} x^{(n)T} x^{(n)} - N \sum_{i=1}^{K} v_i^T \Sigma v_i
\end{aligned}$$

- First term in reconstruction error always positive $\implies$ to minimize reconstruction error, want second term $\sum_{i=1}^{K} v_i^T \Sigma v_i$ as big as possible, which is how PCA was defined $\implies$ PCA lets us represent our original data with the least amount of error with K coefficients