PCA, Eigenfaces, and Face Detection

Salvador Dalí, “Galatea of the Spheres”

Many slides from
Noah Snavely, Derek Hoeim, Robert Collins

CSC320: Introduction to Visual Computing
Michael Guerzhoy

What makes face detection hard?

*Face detection: given an image, find the coordinates of the faces

Variation in appearance: can’t match a single face template and expect it to work.
(Note: it doesn’t work that great for eyes either)

slide by Derek Hoiem

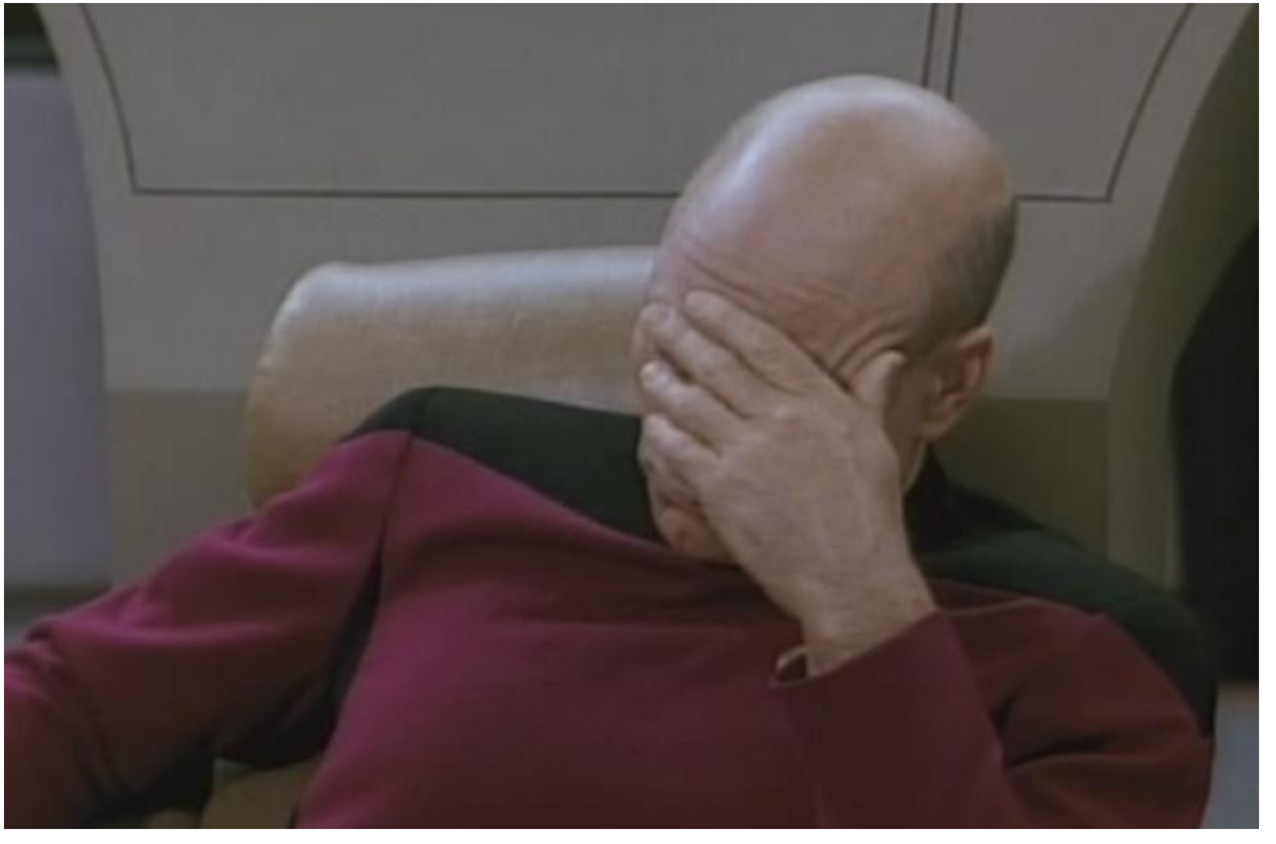
What makes face detection hard?

Lighting

slide by Derek Hoiem

What makes face detection hard?

Occlusion



What makes face recognition hard?

Viewpoint

slide by Derek Hoiem

Face detection

- Do these images contain faces? Where?

Simple Idea for Face Detection

1. Treat each window in the image like a vector

$\mathbf{x}$

2. Test whether x matches some y_j in the database

SSD: $(y_j - x)^2$

Cross-correlation: $y_j \cdot x$

NCC, zero-mean NCC…

slide by Derek Hoiem

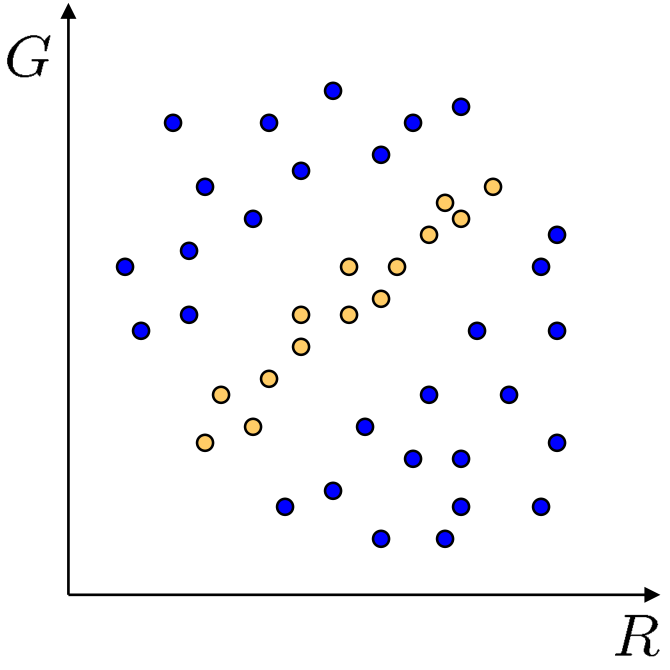
The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
  - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

slide by Derek Hoiem

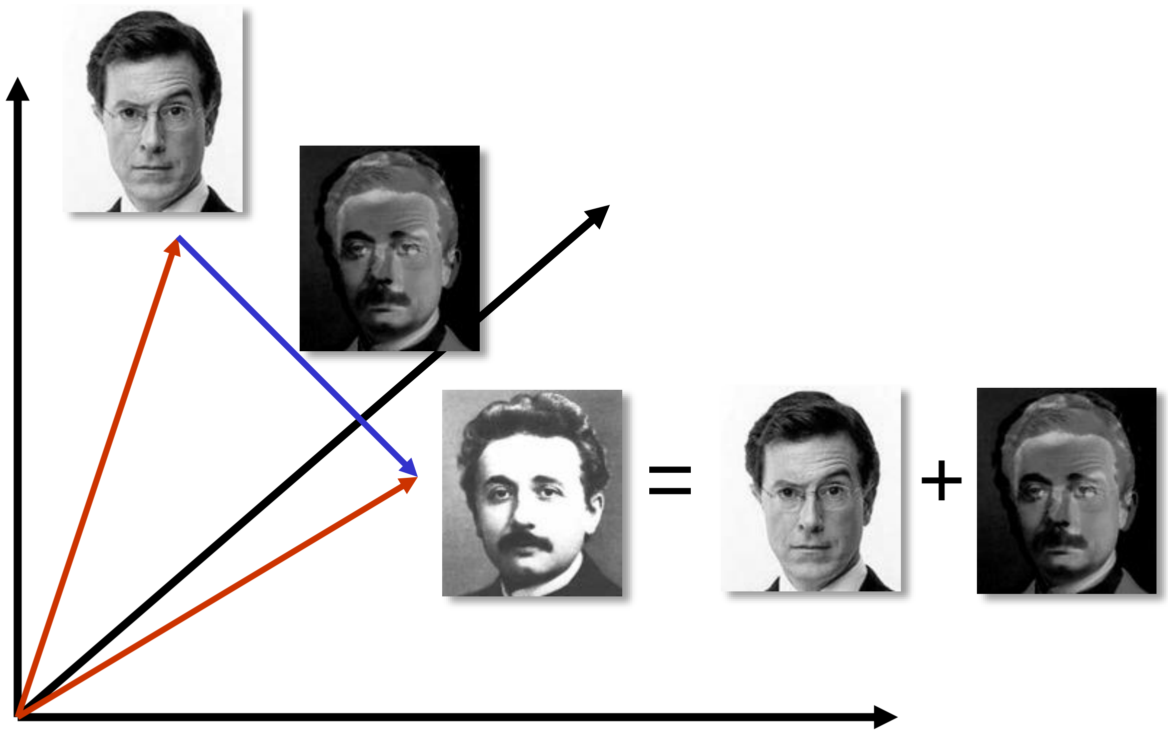
The space of all face images

- Eigenface idea: construct a low-dimensional linear subspace that contains most of the face images possible (possibly with small errors)



- Here: a 1D subspace arguably suffices

The space of faces



An image is a point in a high dimensional space

- An W x H intensity image is a point in R^{WH}
- We can define vectors in this space as we did in the 2D case

Reconstruction

- For a subspace with the orthonormal basis of size k $\mathrm{V_k} = \{v_0, v_1, v_2, \dots v_k\}$, the best reconstruction of *x* in that subspace is:

$$\hat{x}_k = (x \cdot v_0)v_0 + (x \cdot v_1)v_1 + \cdots + (x \cdot v_k)v_k$$

  – If x is in the span of V_k, this is an exact reconstruction

  – If not, this is the projection of *x* on *V*

- Squared reconstruction error: $(\hat{x}_k - x)^2$

Reconstruction cont'd

- $\hat{x}_k = (x \cdot v_0)v_0 + (x \cdot v_1)v_1 + \cdots + (x \cdot v_k)v_k$

- Note: in $(x \cdot v_0)v_0$,
  - $(x \cdot v_0)$ is a measure of how similar x is to v_0
  - The more similar x is to v_0, the larger the contribution from v_0 is to the sum

Representation and reconstruction

- Face **x** in “face space” coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x} - \mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \mu)]$$
$$= \quad w_1, \ldots, w_k$$

- Reconstruction:

 =  + 

$$\hat{x} = \mu + w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$$

slide by Derek Hoiem

Reconstruction

P = 4

P = 200

P = 400

After computing eigenfaces using 400 face images from ORL face database

slide by Derek Hoiem

Principal Component Analysis

- Suppose the columns of a matrix $X_{N \times K}$ are the datapoints (N is the size of each image, K is the size of the dataset), and we would like to obtain an orthonormal basis of size k that produces the smallest sum of squared reconstruction errors for all the columns of $X - \bar{X}$
  - $\bar{X}$ is the average column of X
- Answer: the basis we are looking for is the *k* eigenvectors of $(X - \bar{X})(X - \bar{X})^T$ that correspond to the *k* largest eigenvalues

PCA – cont'd

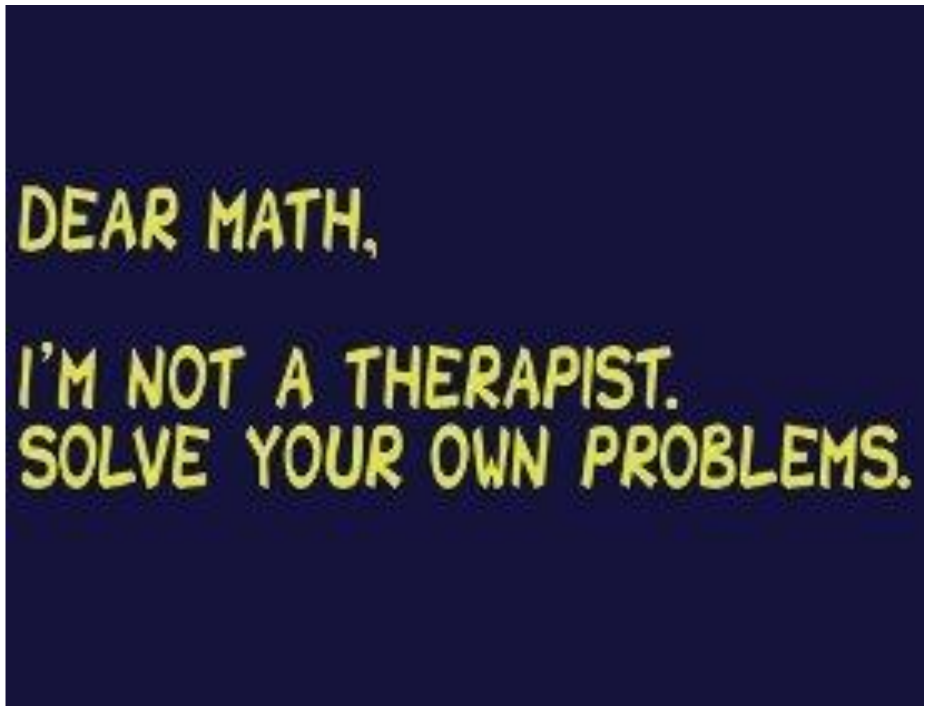
- $(X - \bar{X})(X - \bar{X})^T$ is called the *covariance matrix*
- If x is the datapoint (obtained after subtracting the mean), and V an orthonormal basis, $V^T x$ is a column of the dot products of *x* and the elements of *x*
- So the reconstruction for the **centered** x is $\hat{x} = V(V^T x))$
- PCA is the procedure of obtaining the k eigenvectors V_k

NOTE: centering

- If the image x is *not centred* (i.e., $\bar{X}$ was not subtracted), the reconstruction is:

$$\hat{x} = \bar{X} + V(V^T(x - \bar{X}))$$

Proof that PCA produces the best reconstruction

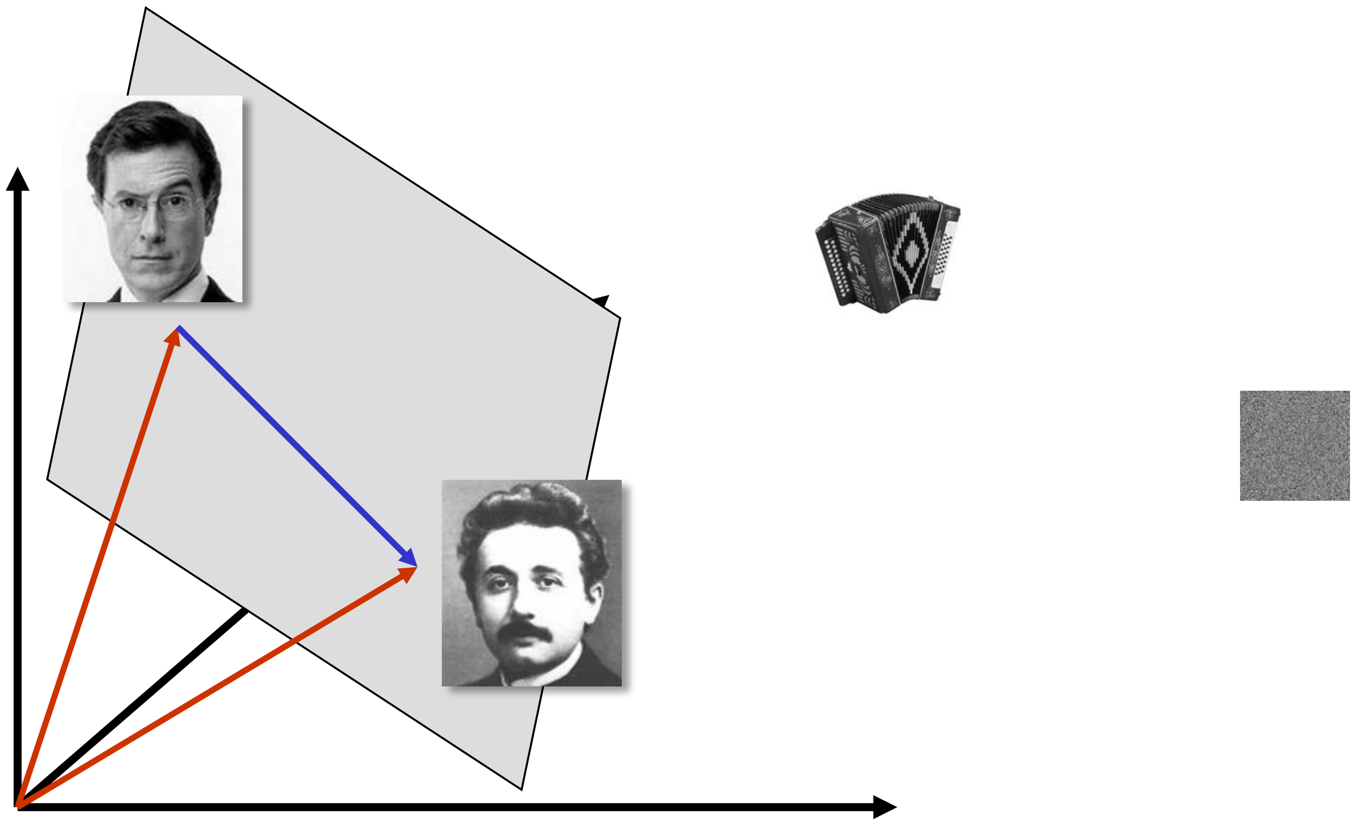


- (*Fairly* easy calculus – look it up, or we can talk in office hours, or possibly we'll do it next week)

Obtaining the Principal Components

- XX^T can be *huge*
- There are tricks to still compute the EVs

PCA as dimensionality reduction



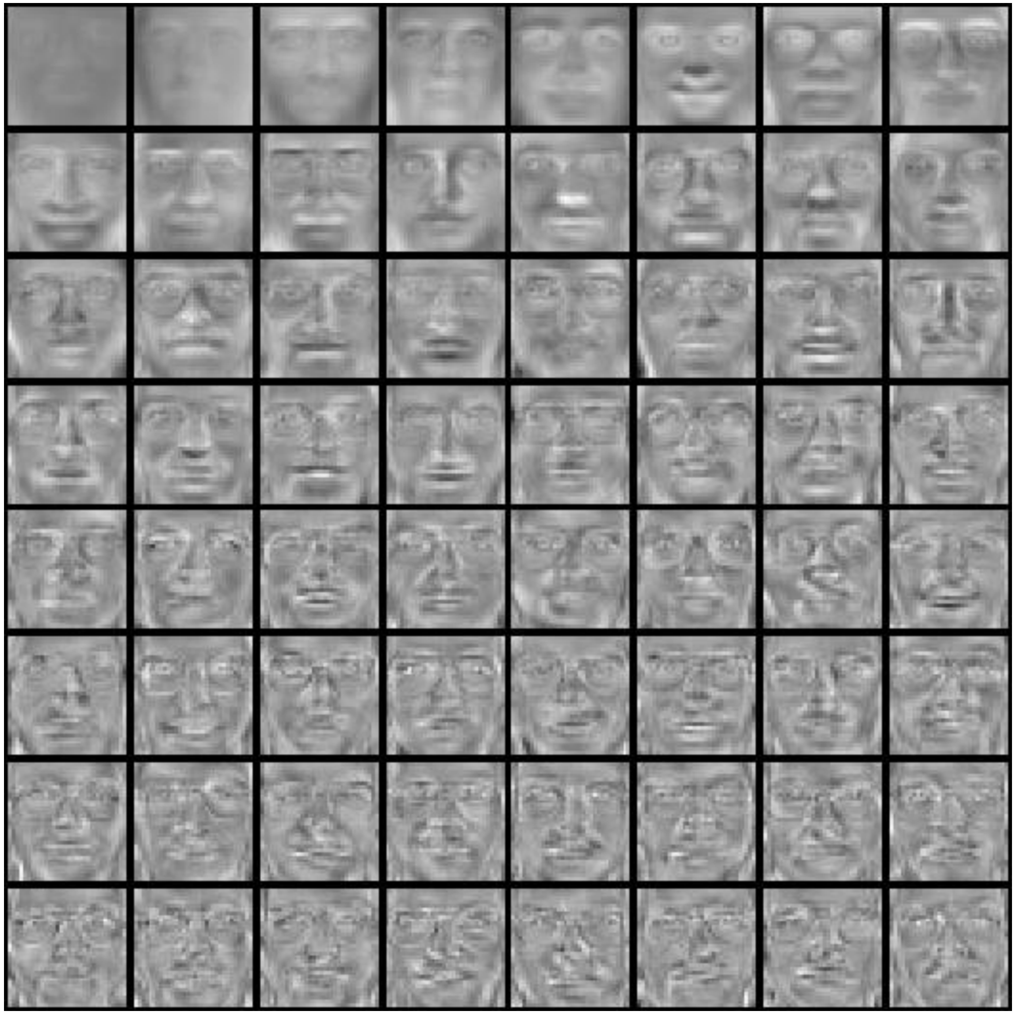
The set of faces is a "subspace" of the set of images

- Suppose it is K dimensional
- We can find the best subspace using PCA
- This is like fitting a "hyper-plane" to the set of faces
  - spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_K}$
  - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

Eigenfaces example

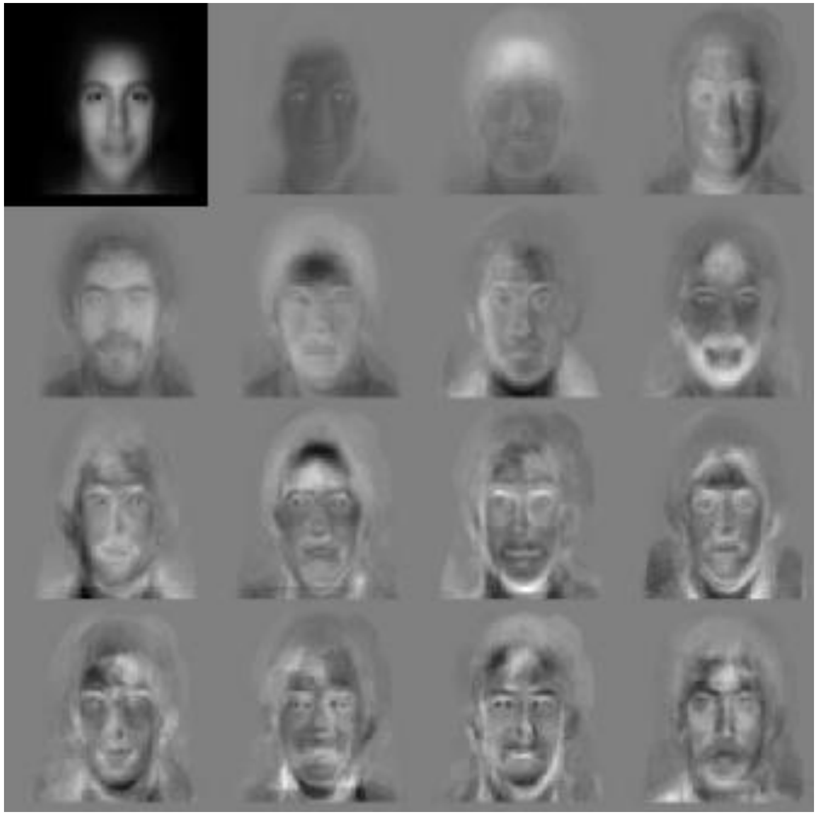
Mean: μ

Top eigenvectors: $u_1, \ldots u_k$

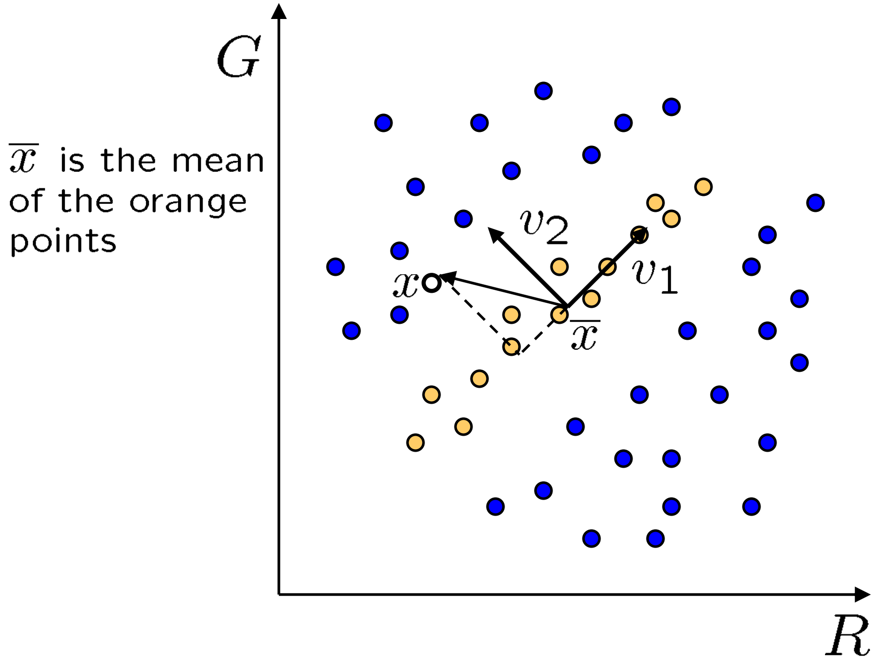


slide by Derek Hoiem

Another Eigenface set



Linear subspaces



convert **x** into $\mathbf{v_1}$, $\mathbf{v_2}$ coordinates

$$\mathbf{x} \rightarrow ((\mathbf{x} - \overline{x}) \cdot \mathbf{v_1}, (\mathbf{x} - \overline{x}) \cdot \mathbf{v_2})$$

What does the $\mathbf{v_2}$ coordinate measure?

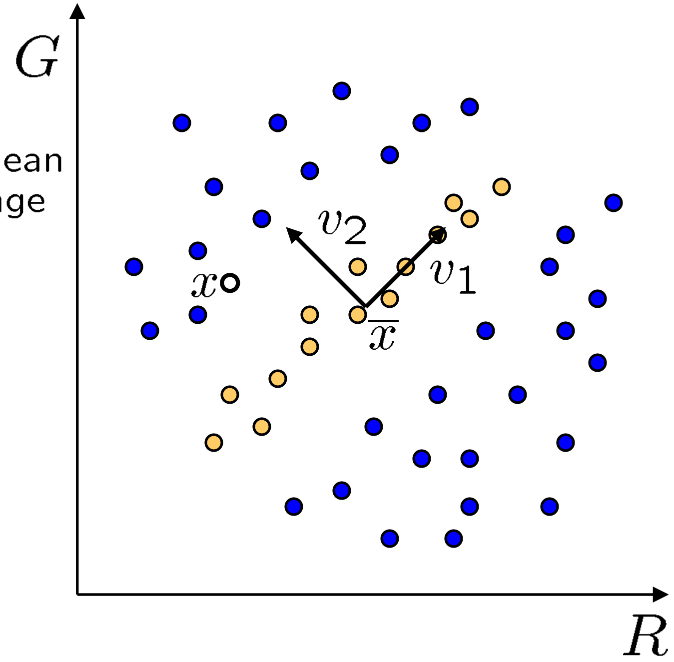
- distance to line
- use it for classification—near 0 for orange pts

What does the $\mathbf{v_1}$ coordinate measure?

- position along line
- use it to specify which orange point it is

Dimensionality reduction

$\overline{x}$ is the mean of the orange points



How to find $\mathbf{v_1}$ and $\mathbf{v_2}$?

Dimensionality reduction

- We can represent the orange points with *only* their $\mathbf{v_1}$ coordinates
  - since $\mathbf{v_2}$ coordinates are all essentially 0
- This makes it much cheaper to store and compare points
- A bigger deal for higher dimensional problems

Another Interpretation of PCA

The eigenvectors of the covariance matrix define a new coordinate system

- eigenvector with largest eigenvalue captures the most variation among training vectors **x**
- eigenvector with smallest eigenvalue has least variation
- The eigenvectors are known as principal components

Data Compression using PCA

- For each data point x, store $V_k^T x$ (a k-dimensional vector). The reconstruction error would be the smallest for a set of k numbers

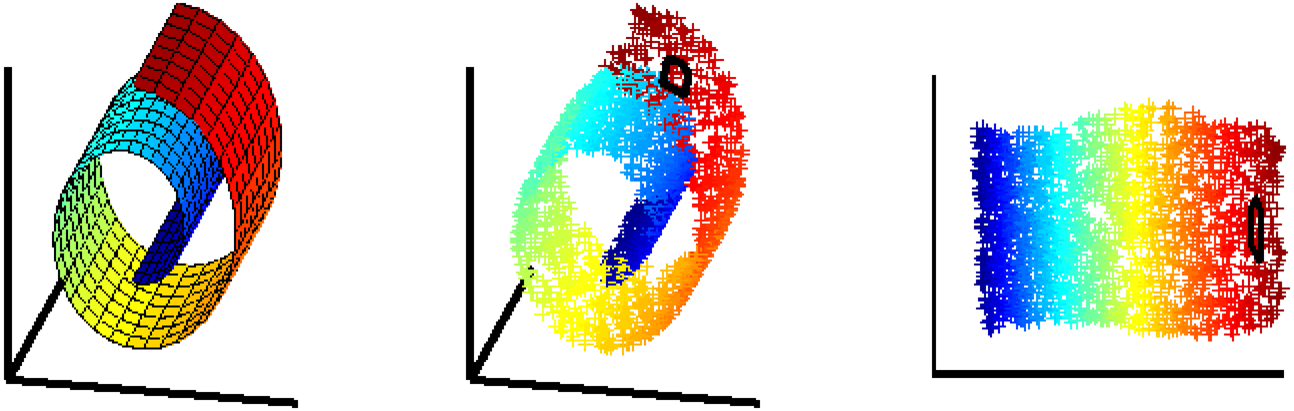
Face Detection using PCA

- For each (centered) window x and for a set of principal components V, compute the Euclidean distance $|VV^T x - x|$
- That is the distance between the reconstruction of *x* and x. The reconstruction of x is similar to x if x lies in the face subspace
  - Note: the reconstruction is *always* in the face subspace

Issues: dimensionality

What if your space isn't *flat*?

- PCA may not help



Nonlinear methods
LLE, MDS, etc.

Moving forward

- Faces are pretty well-behaved
  - Mostly the same basic shape
  - Lie close to a low-dimensional subspace

- Not all objects are as nice

Different appearance, similar parts