

Clustering and Component Analysis

Credit: most of these slides borrow heavily from material prepared by Yon Visell, Jim Clark, and Tal Arbel

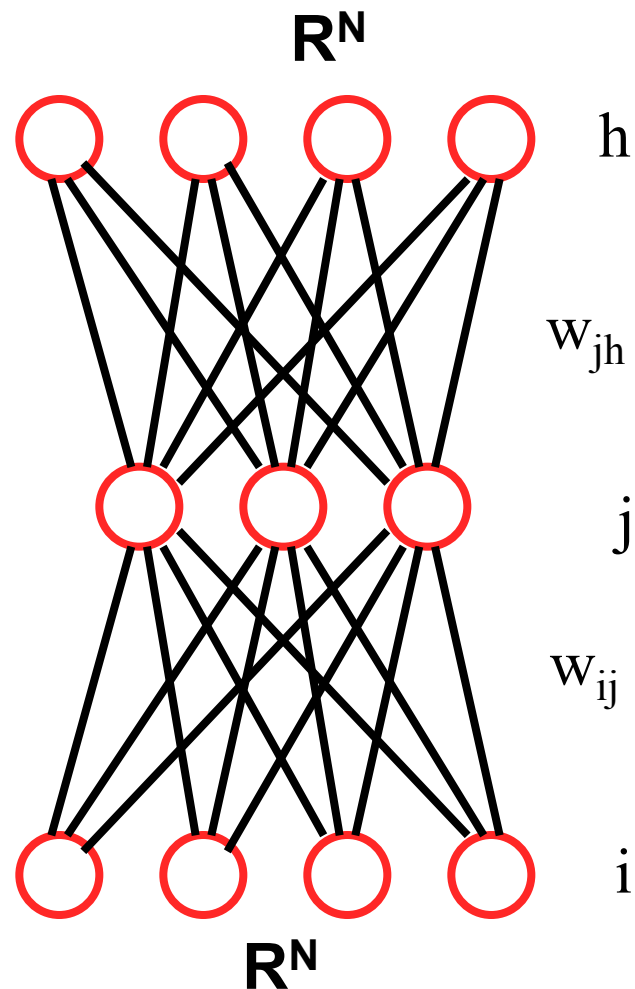
Introduction

- each data source (image, sound clip, etc.) can be represented by a single vector \mathbf{d} of length M
- let $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N\}$ denote the set of N such data in the database
- if M is the dimension in which the data lies, the goal is to find an optimal subspace of dimension S , where $S \ll M$

Clustering with k -means

- partitions the data points into k disjoint subsets based on a distance measure between instances
- advantage: easy to implement
- input: a set of n -dimensional vectors $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$
- output: a mapping $\mathbf{C} = \{C_1, C_2, \dots, C_k\}$ of the vectors into k disjoint clusters

Autoencoder Network Equivalence



- the function quantizer and approximation network provides a bottleneck in the hidden layer that serves to effect a data reduction

***k*-means algorithm**

- initialize C randomly
- repeat
 - compute centroid of each cluster
 - assign each vector to the closest centroid using Euclidean distance
- until C unchanged

Termination of k -means clustering

- for given data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ and clustering C , consider the sum of the squared Euclidean distance between each vector and the center of its cluster:

$$J = \sum_{i=1}^m \|x_i - \mu_{C(i)}\|^2$$

- finitely many possible clusterings: at most k^m
- each time we reassign a vector to a cluster with a nearer centroid, J decreases
- each time we recompute the centroids of each cluster, J decreases (or stays the same) \Rightarrow algorithm must terminate!

Does k-means always find same answer?

- solution is *locally optimal*
- error function has many local minima
- depends on initial assignment of instances to clusters



$$J = 0.2287$$



$$J = 0.3088$$

Finding good initial configurations

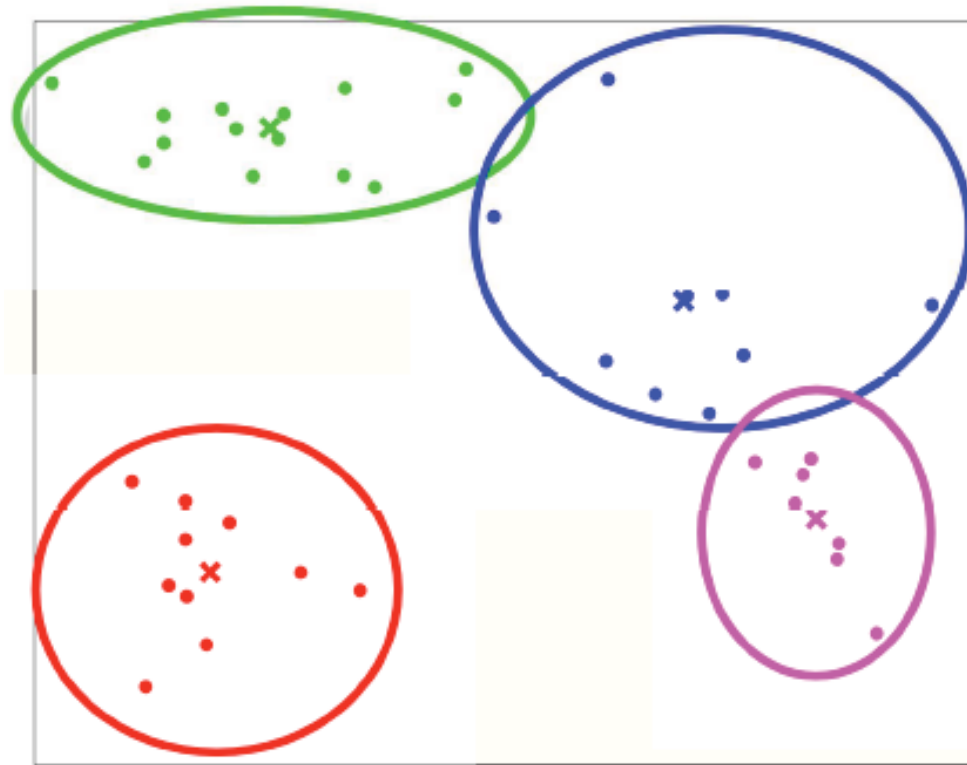
- place first center on a randomly chosen data point
- place second center on a data point far away as possible from first
- place the i^{th} center as far away from the closest of center 1 through $i-1$

Choosing number of clusters

- delete clusters that cover too few points
- split clusters that cover too many points
- add extra clusters for “outliers”
- *minimum description length* principle:
minimize loss + complexity of the clustering
- perceptual or other application-specific criteria

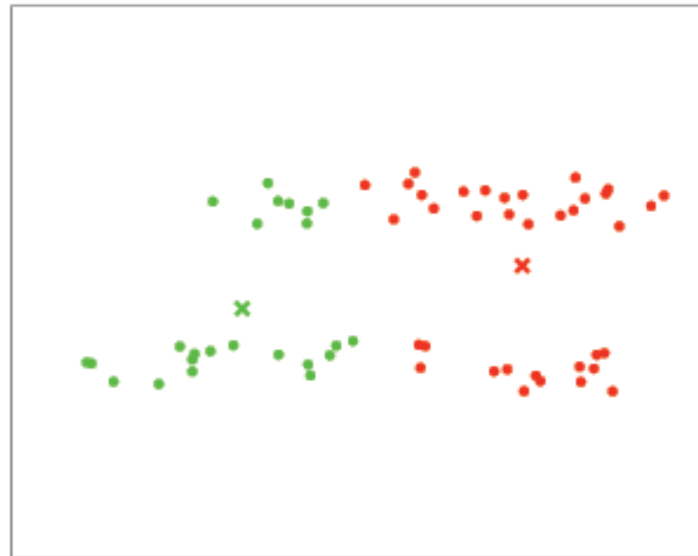
Why use sum of squared Euclidian distances?

- subjective: produces nice round clusters

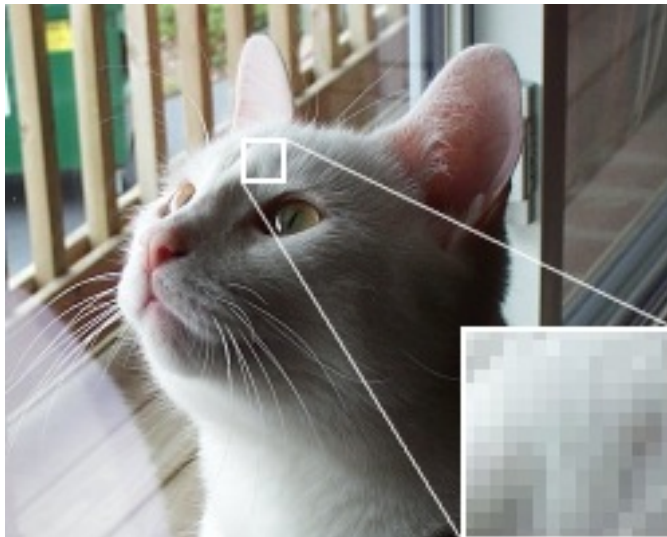


Why not use sum of squared Euclidian distances?

- produces nice round clusters
- differently scaled axes can dramatically affect results
- symbolic attributes may have to be treated differently



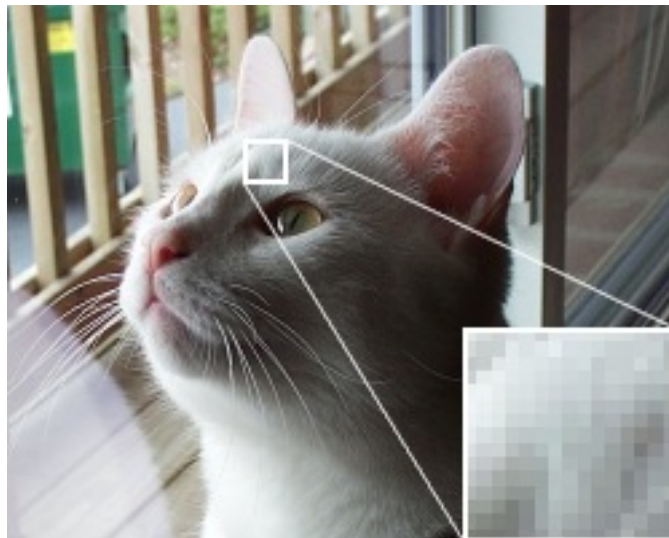
Example application: Color Quantization



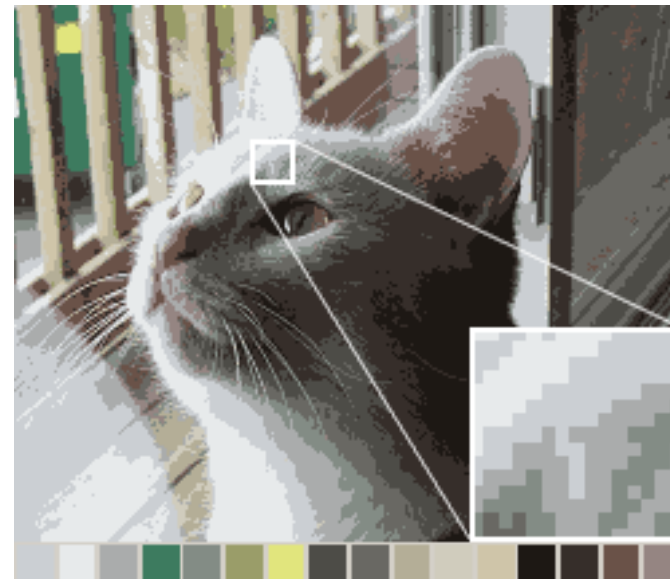
original: 24-bit RGB color

- want compressed version to look as similar as possible to original
- can transmit only compressed version plus color map
- want to minimize reconstruction error

Example application: Color Quantization



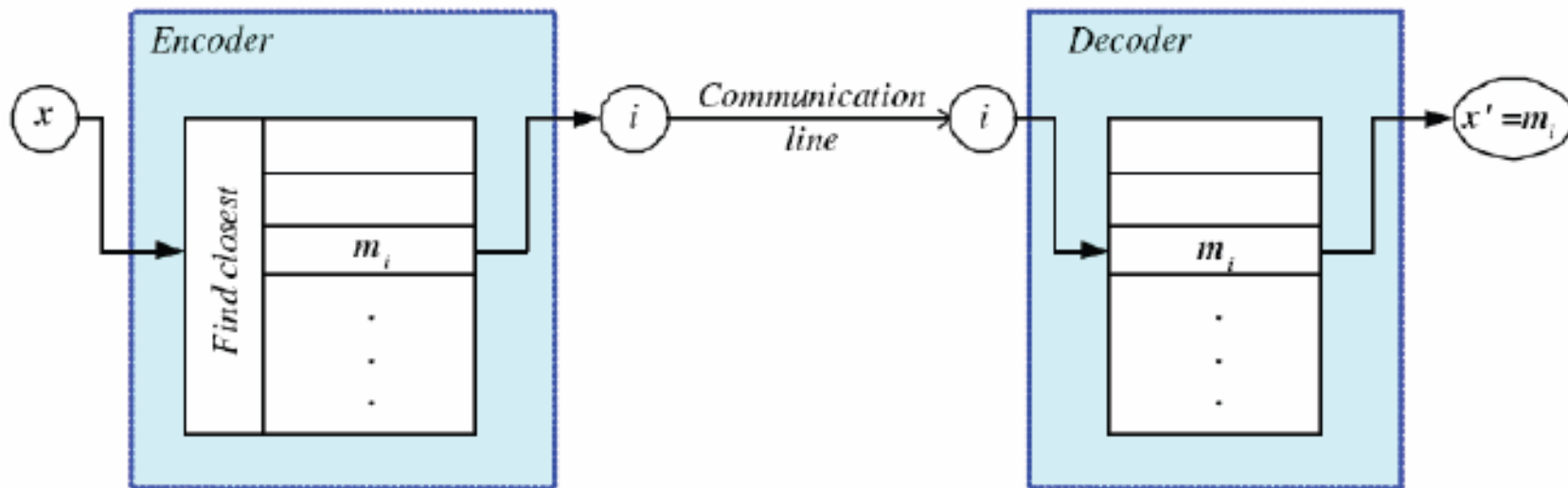
original: 24-bit RGB color



reduced to 4-bits (16 colors)

Example application: Speech Coder

- determine a compact (low bits/sec) representation of speech, possibly for transmission over communication line



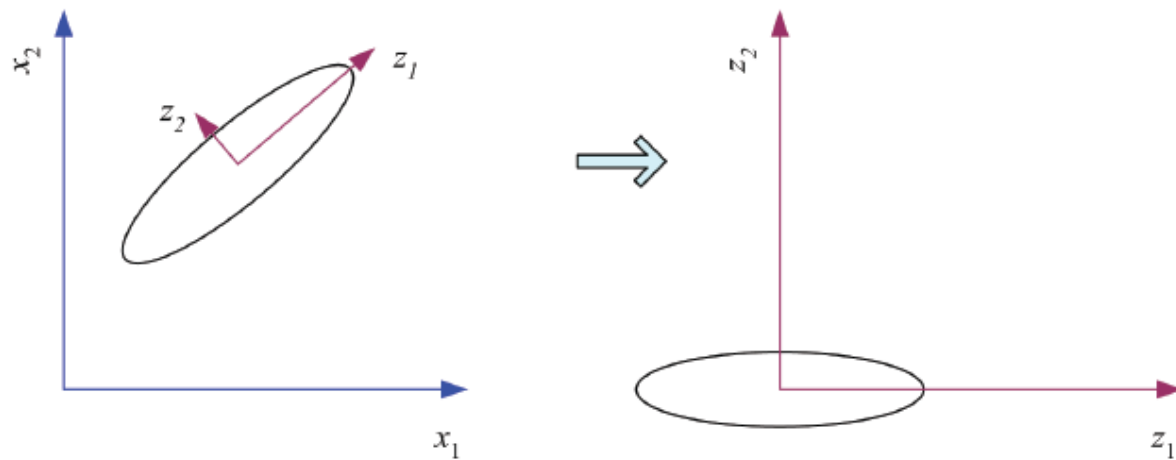
Principal Components Analysis

- If a multivariate dataset is visualised as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA supplies the user with a lower-dimensional picture, a "shadow" of this object when viewed from its (in some sense) most informative viewpoint.

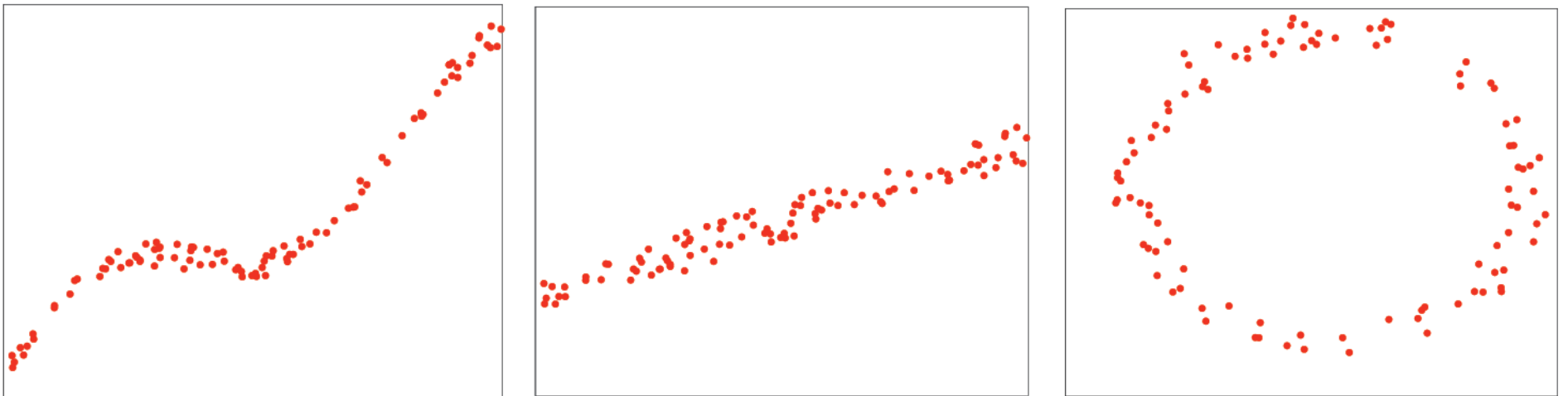
http://en.wikipedia.org/wiki/Principal_components_analysis

How PCA works

- PCA centers the data and rotates the axes
- this reduces dimensionality of data
- greatest variance of data represented by first component of transformed data



What is the true dimensionality?



$$\mathbf{w}_1 = \arg \max_{\|\mathbf{w}\|=1} \text{var}\{\mathbf{w}^T \mathbf{x}\} = \arg \max_{\|\mathbf{w}\|=1} E \left\{ (\mathbf{w}^T \mathbf{x})^2 \right\}$$

Determining the lower-dimensional subspaces

- find single axis v_1 that best models the data points d_i
- this gives a one-dimensional model for the data
- next, find another axis v_2 that best models the deviation of data points from the first axis
- repeat previous step as desired for v_j up to dimensionality of the data space

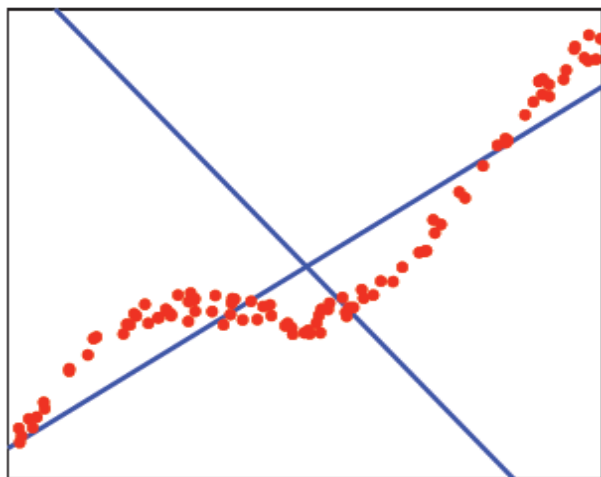
Finding the axes v : two approaches

1. minimize the square modeling error: $E_R = \sum \|d_i - v^T d_i v\|^2$
where $v^T d_i v$ represents projection of the data onto axis v

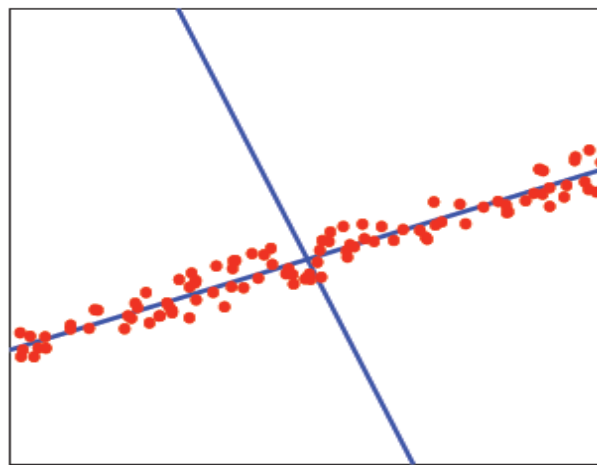
2. maximize the variance of the data after projection onto the axis v (assuming the projection has zero mean):

$$\sigma^2 = \frac{1}{N-1} \sum (d_i^T v)^2$$

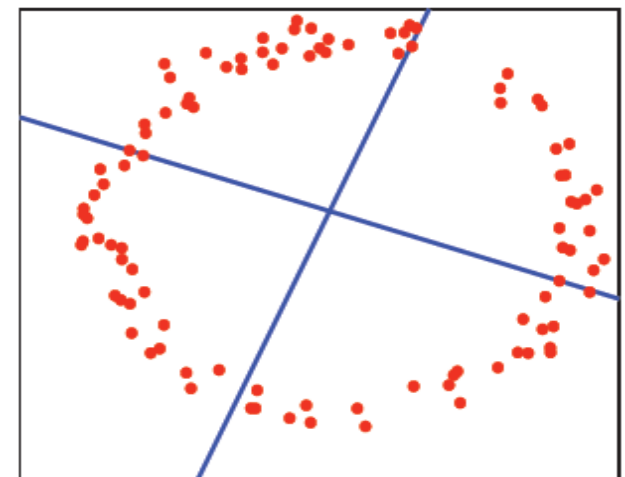
First two principal components



$\lambda_1 = 0.0938, \lambda_2 = 0.0007$



$\lambda_1 = 0.1260, \lambda_2 = 0.0054$



$\lambda_1 = 0.0884, \lambda_2 = 0.0725$

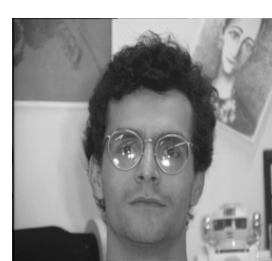
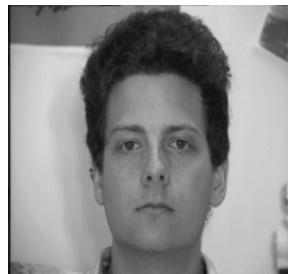
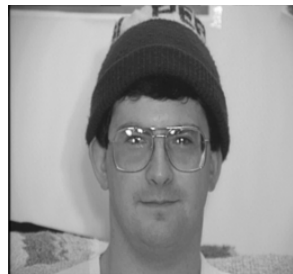
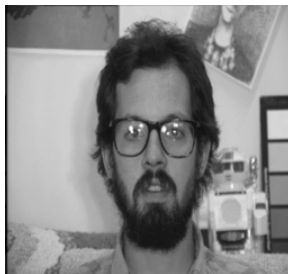
Face Recognition

- *Identify face from a database of known faces.*

?



From MIT database



Why is this hard?

- faces are similar, same set of features (e.g. eyes, nose, mouth) in same configuration
- have to find representation that captures enough features to identify face from database of faces.
- early attempts – model relations between face features
- ignores important information about texture and shape

Face Recognition using PCA

[Sirovich and Kirby, 1987, 1990]

- PCA provides a set of standard ingredients to model faces
- good results using simple recognition algorithms with eigenface decomposition, e.g. nearest neighbor classifier
- found that 50 Principal Components give an average modeling error of 3.68% over ten test images
- better results by performing PCA on local features of an image

Claimed advantages

- good for vague recognition that face is “known” or “unknown”
- capture global, non-subjective features of faces

Eigenfaces

[Turk & Pentland, 1991]

