Machine Learning

CS 165B

Prof. Matthew Turk

Wednesday, May 18, 2016

Today

- Features (Ch. 10)

Notes

Four Eyes Lab Open House

Friday, May 27, 5-8pm

3rd floor Elings Hall

Along with the Media Arts and Technology End of Year Show

Exhibition 5-9pm, throughout Elings Hall

http://show.mat.ucsb.edu

Summary: Distance methods and clustering

- Similarity is a function of distance
- Euclidian distance may not always be the right choice
- Nearest neighbor methods assign classes/clusters based on distances to points or exemplars, not based on computed boundaries
- For good clustering, we want high within-class (intra-class) similarity and low between-class (inter-class) similarity
- The scatter matrix is an important structure in clustering
- The K-means algorithm (and variations) is widely used

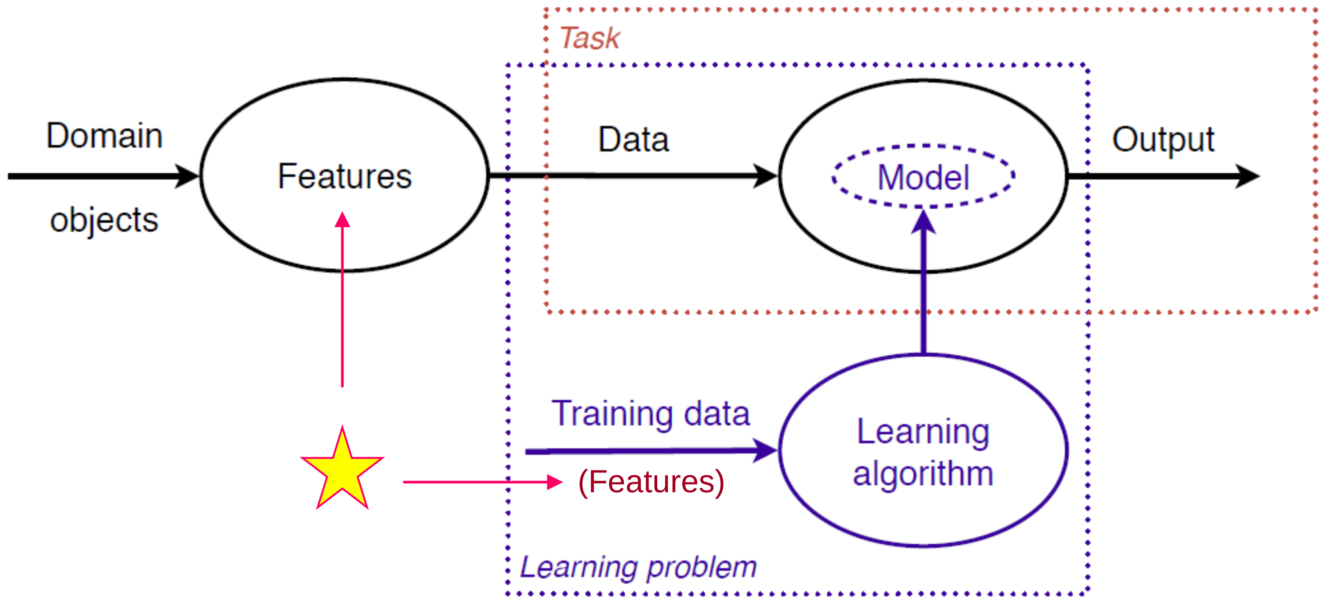
Features

Chapter 10 in the textbook

(mostly 10.3)

Schedule change: skipping Chapter 9!!

A machine learning system



A **task** requires an appropriate mapping – a **model** – from data described by **features** to outputs. Obtaining such a model from training data is what constitutes a **learning problem**.

Tasks are addressed by **models**.

Learning problems are solved by **learning algorithms** that produce **models**.

Features

- Let's think about features (aka attributes) a bit more than we have….
- What's a feature? A mapping from the instance space $\mathscr{X}$ to the feature domain $\mathscr{F}$

$$f_i : \mathscr{X} \rightarrow \mathscr{F}_i$$

- The model can be thought of as just a new feature – a particular combination of the input features, constructed to solve the task at hand
  - E.g., the $\boldsymbol{w}^T\boldsymbol{x}$ value in a linear classifier or SVM
- There are different categories of features and of permissible operations on features

Main feature types

- Quantitative features
  - Measured on a meaningful numeric scale
  - Domain is often real values
  - E.g.: weight, height, age, angle, match to template, …
- Ordinal features
  - The relevant information is the ordering, not the scale
  - Domain is an ordered set
  - E.g., rank, street addresses, preference, ratings, …
- Categorical features
  - No scale or order information
  - Domain is an unordered set
  - E.g., colors, names, parts of speech, binary attributes, …

Main feature types

Kinds of features, their properties, and allowable statistics:

Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	×	×	mode	n/a	n/a
Ordinal	√	×	median	quantiles	n/a
Quantitative	√	√	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Statistics – calculations on the features

Mode – the value that occurs most frequently
Median – the middle value in an ordered list
Mean (expected value) – the arithmetic average of the values

$\{ 0, 0, 0, 1, 2, 9, 1000 \} \Rightarrow$ Mode = 0
Median = 1
Mean = 144.57

Main feature types

Kinds of features, their properties, and allowable statistics:

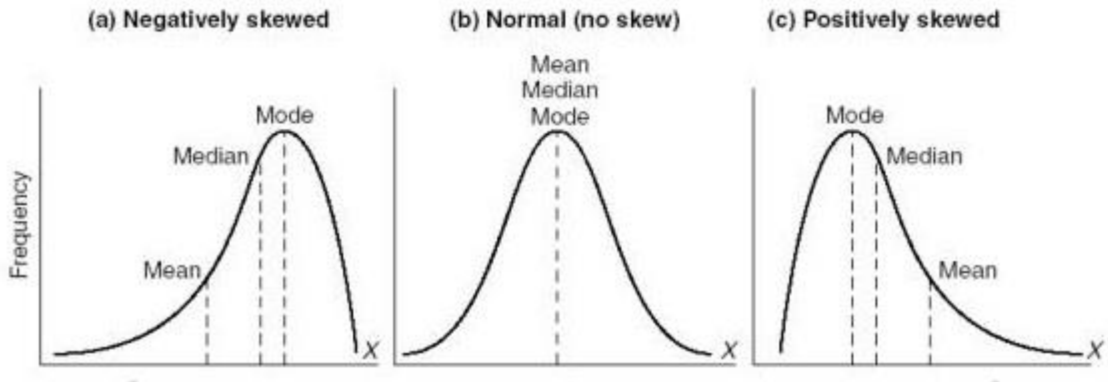
Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	$\times$	$\times$	mode	n/a	n/a
Ordinal	$\surd$	$\times$	median	quantiles	n/a
Quantitative	$\surd$	$\surd$	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Statistics – calculations on the features

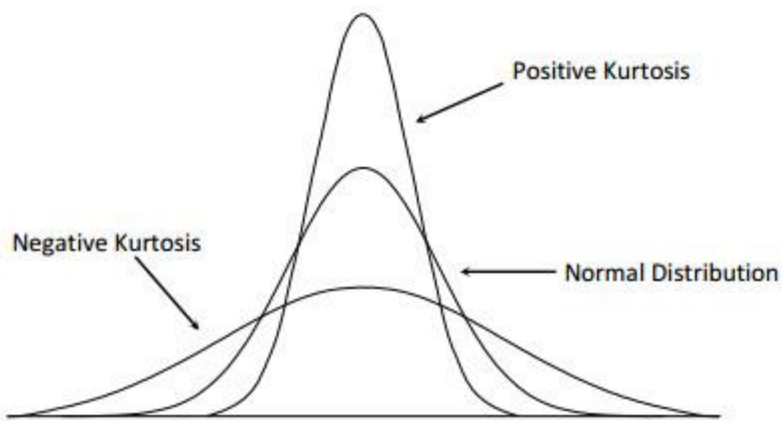
Range = (max. value – min. value)
Standard deviation (σ) = Square root of the variance (σ^2)
Skewness ∝ 3rd moment (lack of symmetry: right or left skewed)
Kurtosis ∝ 4th moment (peakedness relative to a normal distribution)

Skewness and kurtosis

Skewness:



Kurtosis:



Main feature types

Kinds of features, their properties, and allowable statistics:

Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	×	×	mode	n/a	n/a
Ordinal	√	×	median	quantiles	n/a
Quantitative	√	√	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Statistics – calculations on the features

Percentiles: p^{th} percentile = the value such that *p* percent of the instances fall below it

Deciles – multiples of 10 percentiles (10^{th}, 20^{th}, etc.)

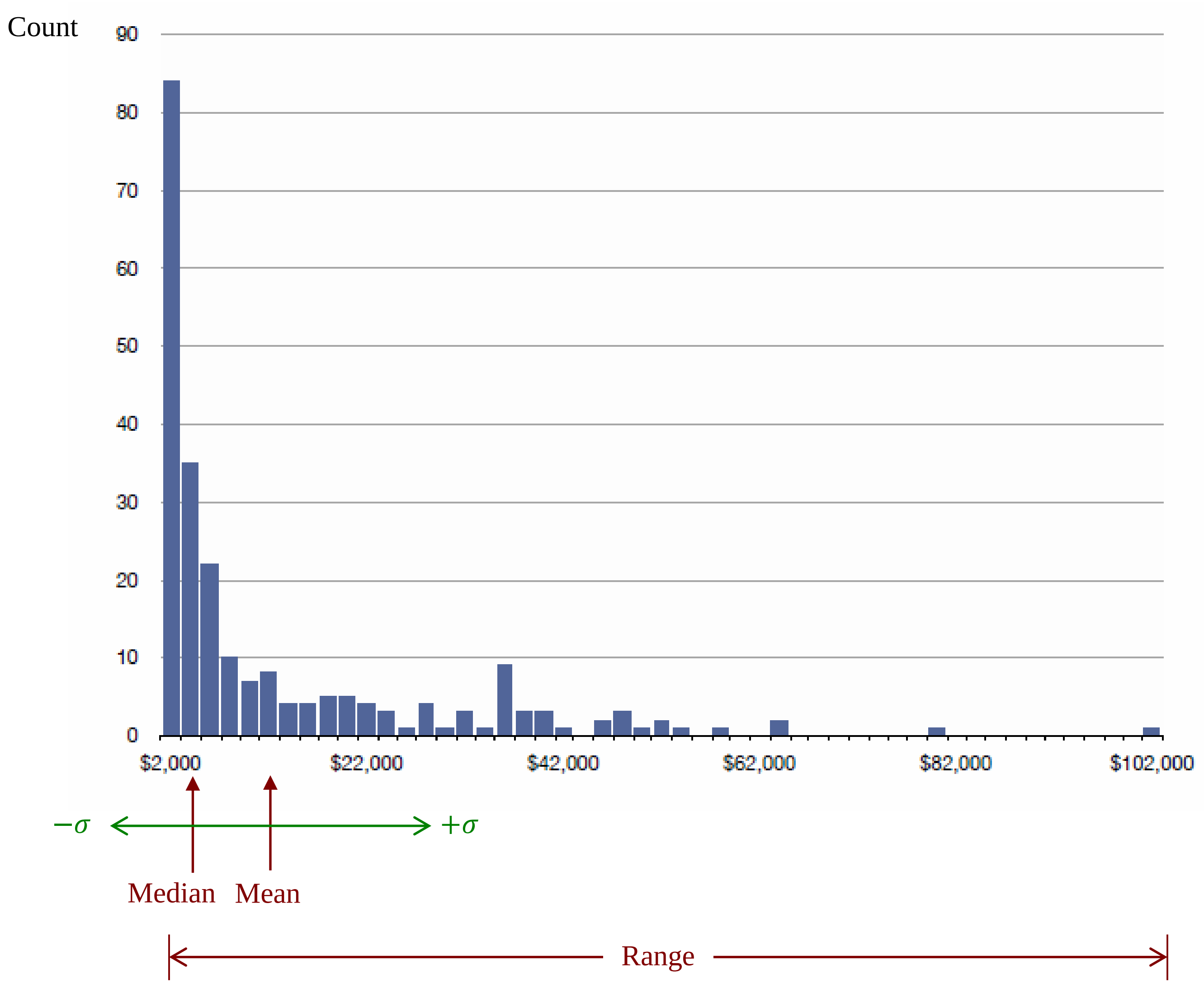
Quartiles – multiples of 25 percentiles (25^{th}, 50^{th}, etc.)

Interquartile range – the difference between the first and third quartiles (75^{th} and 25^{th} percentiles)

Histogram

Shows data as a frequency plot, with data collected in *bins*

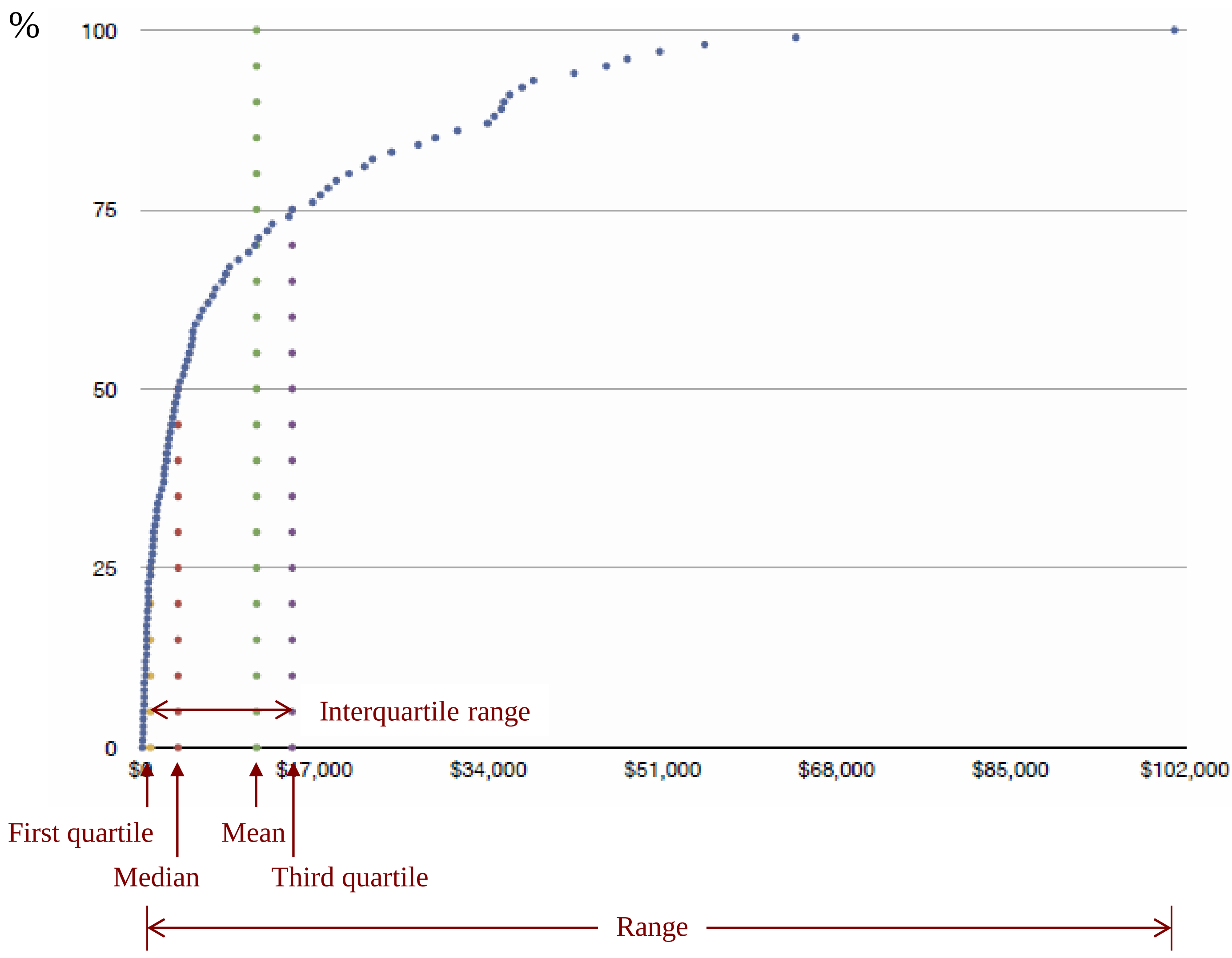
A histogram of GDP per capita for 231 countries (bin size = $2000):



Percentile plot

Shows cumulative fraction of data – what % fall below a given value

A percentile plot of GDP per capita for 231 countries:

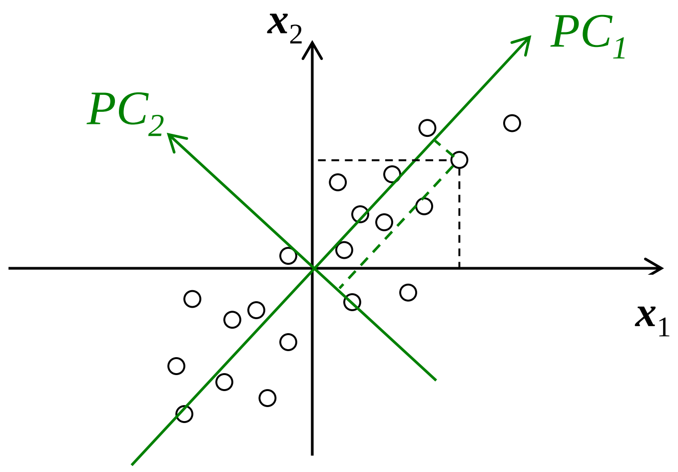


Feature construction and selection

- New features can be constructed from the given feature set
  - E.g., kernel methods do this, sometimes for great benefit
- Features may be combined in various ways, e.g.:
  - N-grams – groups of N consecutive elements
    - E.g., trigrams: “my first teacher,” “my first job,” “my first the”
  - Cartesian products of categorical features
    - E.g., Shapes × Colors = { blue circle, red square, green line, … }
  - ***w*** vector for a perceptron (combining training data point features)
  - Kernel function κ (e.g., 2D to 3D transformation)
- Typical approach: Generate new features, then select a suitable subset prior to learning
  - How to determine/evaluate suitability? How many to keep?
    - Many different approaches to this problem….

PCA and eigendecomposition

- One of the most widely used feature construction/selection techniques is Principal Component Analysis (PCA)
  - PCA constructs new features that are linear combinations of given features
- Computed eigenvectors and eigenvalues hold useful information
- Often used for dimensionality reduction, finding the intrinsic linear structure in the data



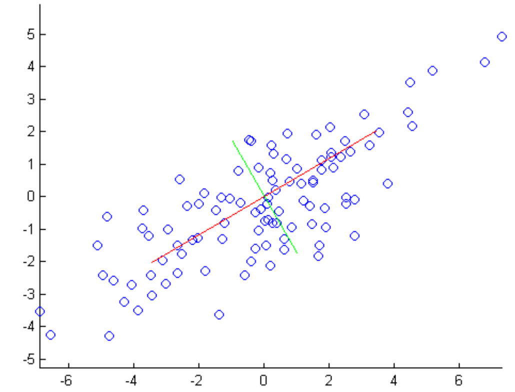
Given features 1 and 2 (x_1, x_2)
Computed features 1 and 2 (green axes)

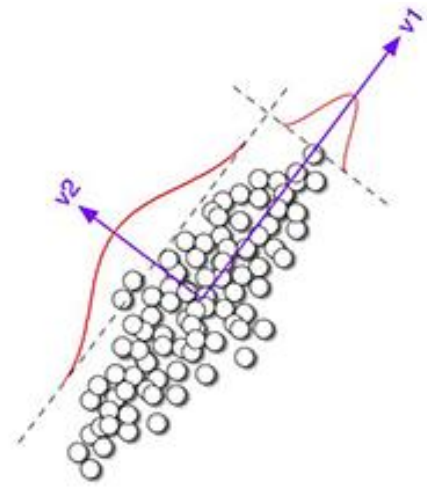
$$\mathrm{PC}_1 = \underset{\boldsymbol{y}}{\operatorname{argmax}} (y^T X)(X^T y)$$

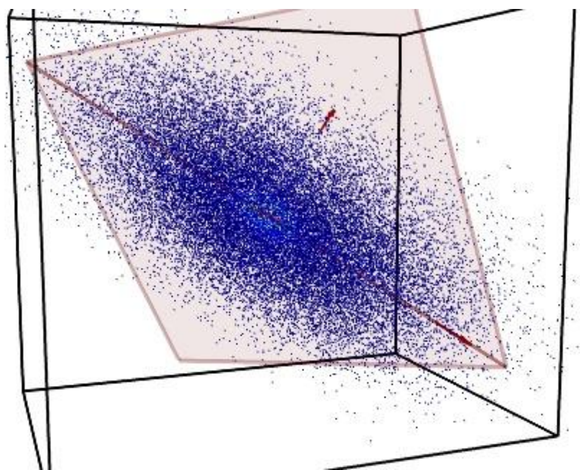
(maximize variance of points projected onto unit vector y)

PCA and eigendecomposition

- The first principal component is the direction of maximum (1D) variance in the data
- The second principal component is the direction, perpendicular to the 1st PC, of maximum variance in the data
  - Etc. for additional PCs
- For N-dimensional data, there are N principal components
  - But perhaps only *k* of them are useful (k < N) – dimensionality reduction!
- PCA typically assumes zero-mean data
  - First subtract the mean from each data point; centroid is thus (0, 0)





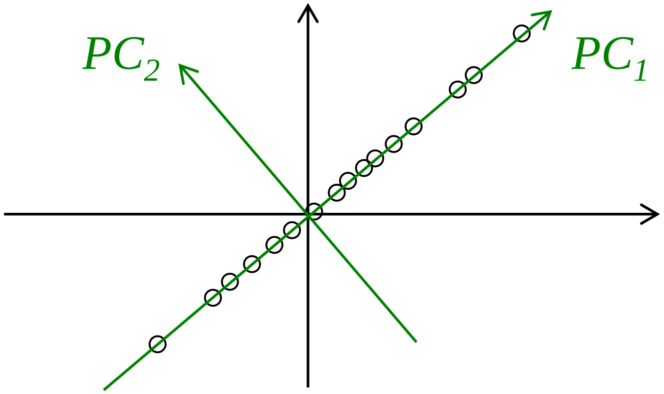


PCA and eigendecomposition

- For n points of dimension d, let $\mathrm{X} = [x_1\ x_2\ \ldots\ x_n]$ $(d \times n)$
- The eigenvectors of $\mathrm{S} = \mathrm{X_z X_z}^{\mathrm{T}}$ $(d \times d)$ are the principal components of the data, ordered by decreasing eigenvalues

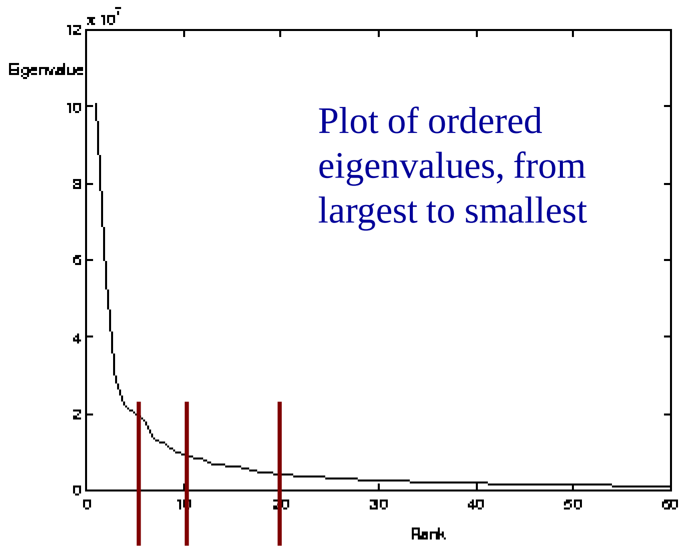
  $$Su_i = \lambda_i u_i \rightarrow SU = U\Lambda$$

  where U is the matrix of eigenvectors (columns of U) and Λ is a diagonal matrix of eigenvalues $\mathrm{diag}(\lambda_i, \lambda_2, \ldots, \lambda_d)$
- The eigenvector u_i associated with the largest eigenvalue λ_i is the first principal component
- If rank(S) < d, then some eigenvalues will be zero



PCA for dimensionality reduction

- So the eigenvalues can give clues to the inherent dimensionality of the data, or at least provide a way to more efficiently approximate high-dimensional with lower-dimensional feature vectors
- For example:



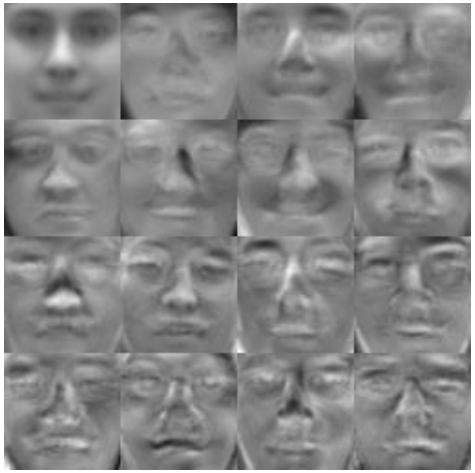
60-dimensional data (60 eigenvectors and eigenvalues)

Many of the eigenvalues are small, meaning that their associated eigenvectors don’t contribute much to the representation of the data

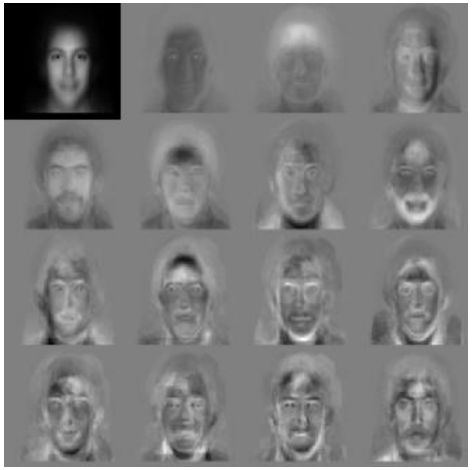
We can choose a cutoff – say, only use the first 20 eigenvectors (or 10, or 5)

Face recognition via "Eigenfaces"

- A well-known technique for face recognition based on computing eigenvectors of a training set of face images, i.e., Eigenfaces



Eigenfaces 1

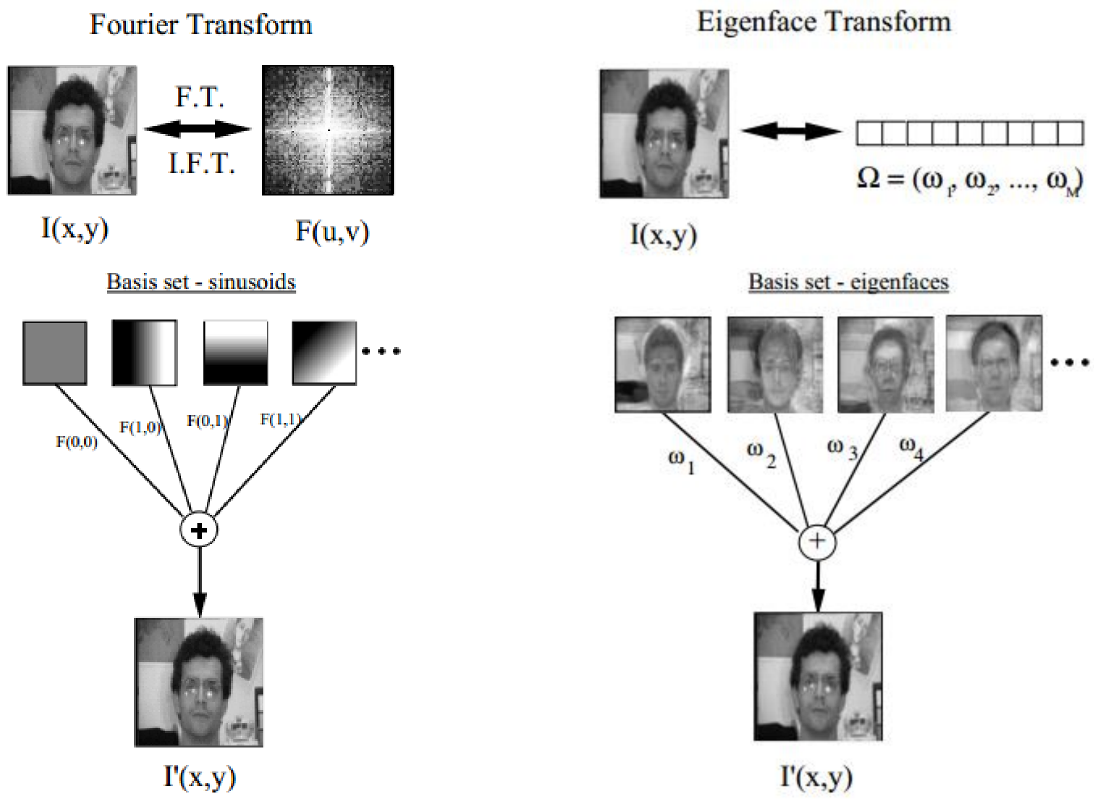


Eigenfaces 2

Keep in mind: an image is just an N-dimensional point or vector (where rows × cols = N)

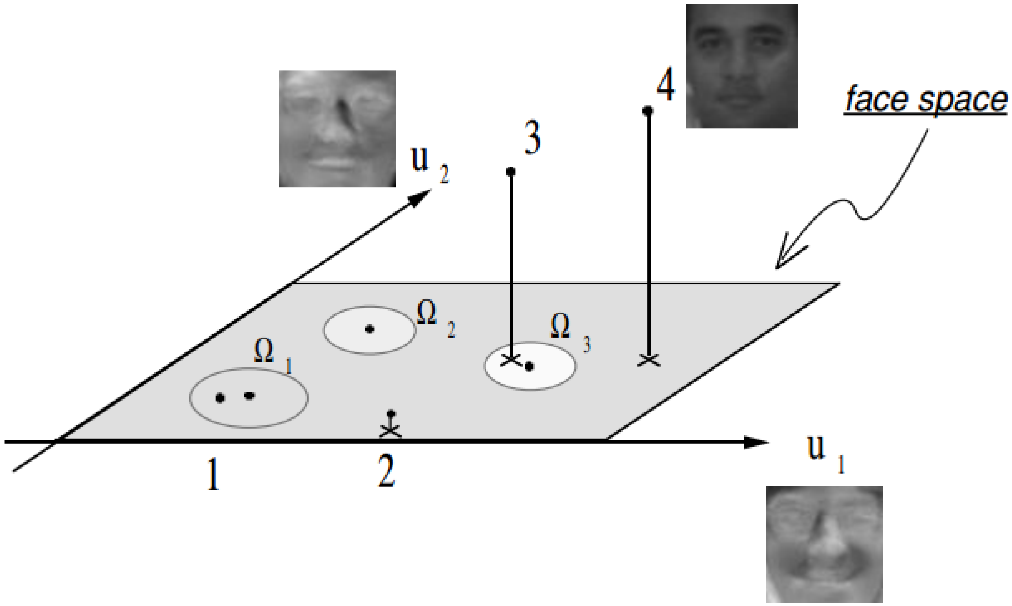
Face recognition via "Eigenfaces"

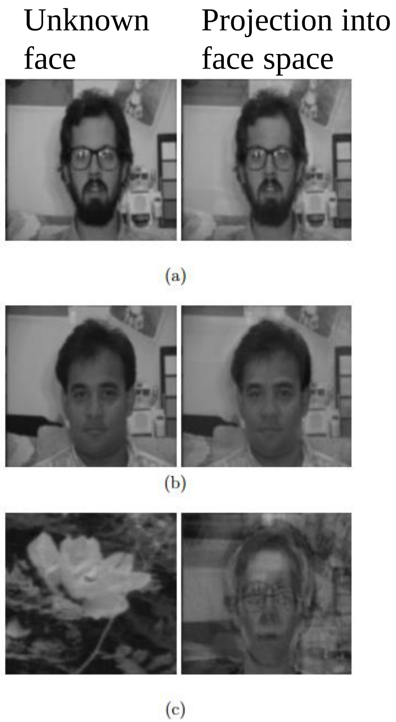
- Eigenvectors (eigenfaces) can be thought of as *basis vectors* for reconstructing data (face images)



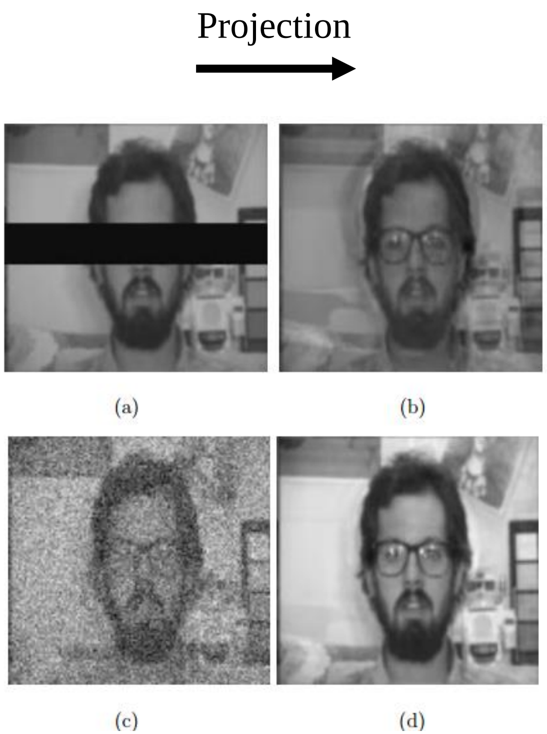
Face recognition via “Eigenfaces”

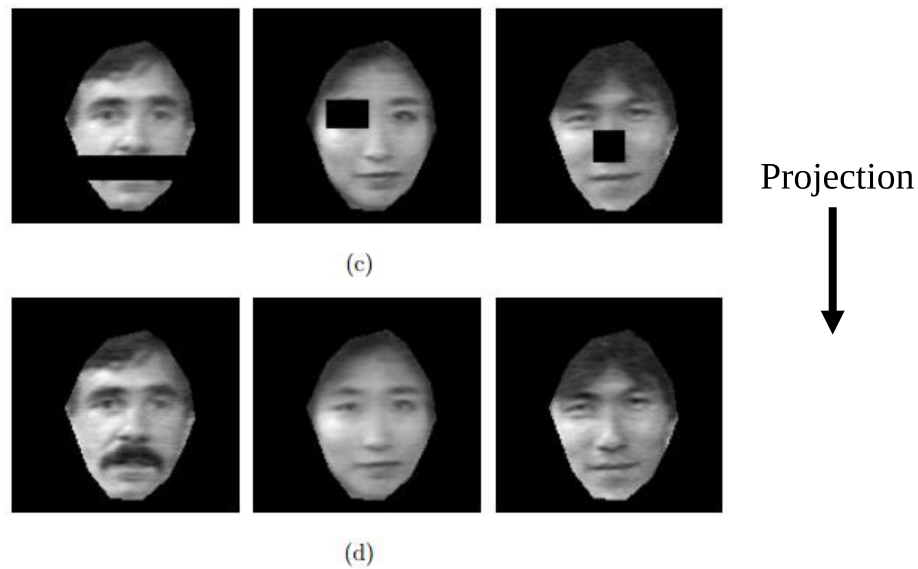
- The Eigenfaces span a (relatively) low-dimensional *face space*, representing all possible face images
- A new (unknown) face image is projected into the face space (reconstructed by the Eigenfaces)
  - The distance between the face image and its reconstruction is the *distance from face space*





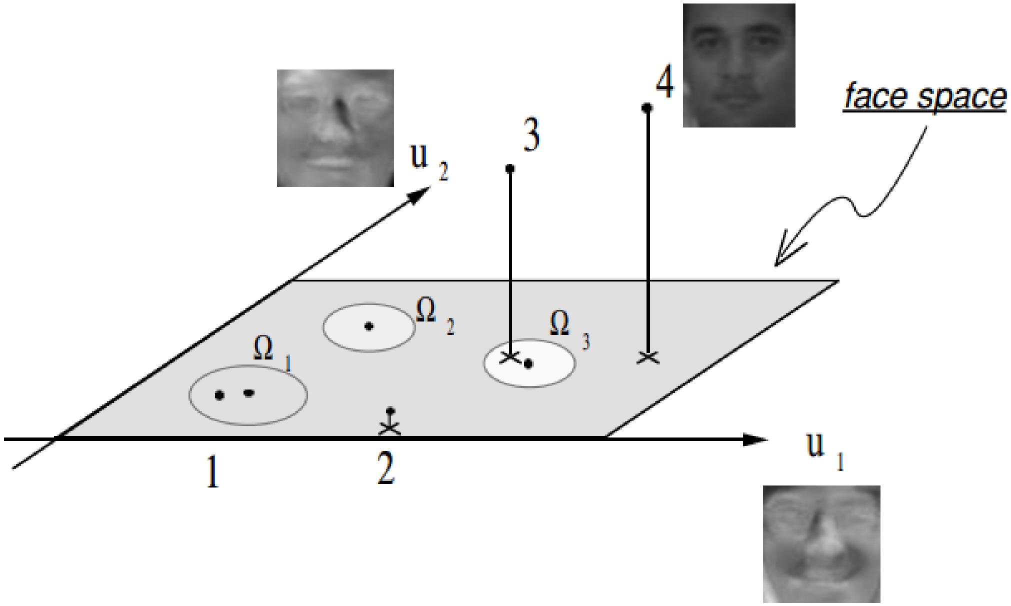
Face recognition via “Eigenfaces”

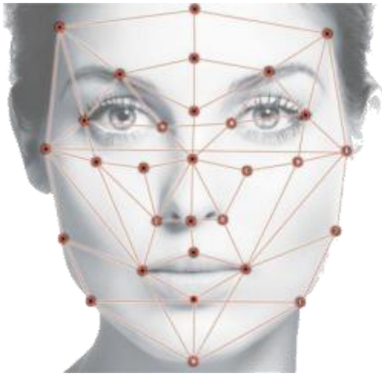


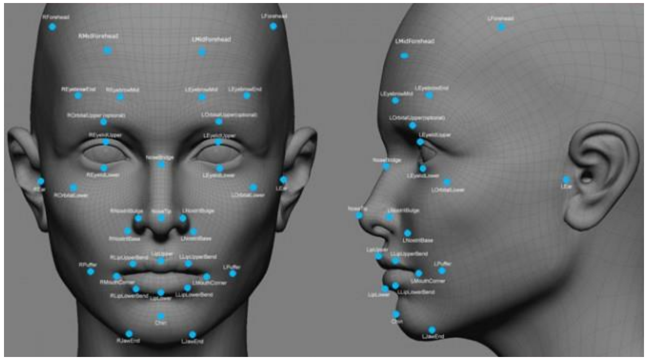


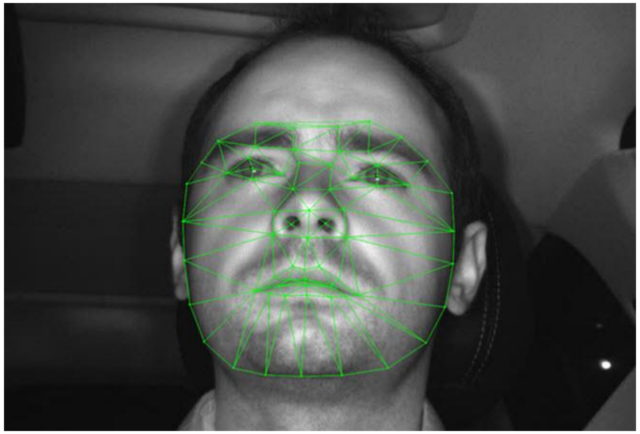
Face recognition via "Eigenfaces"

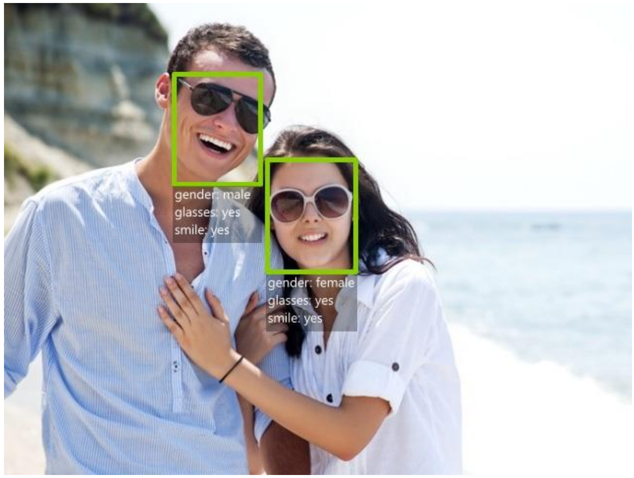
- The *distance from face space* measure could be used for face detection: Does this image (or part of an image) look like a face?
- If yes, then use the Eigenface weights (projections) as features for classification (classes Ω_1, Ω_2, etc.)
  - E.g., using k-NN

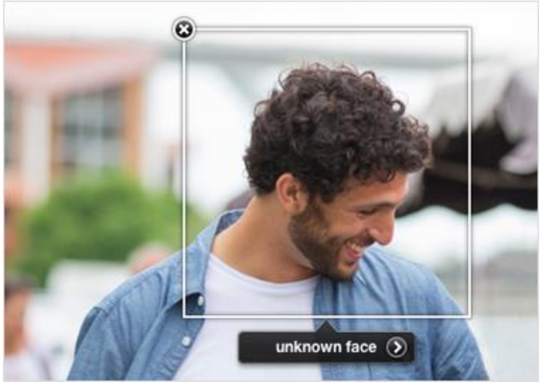









gender: male
glasses: yes
smile: yes
gender: female
glasses: yes
smile: yes


unknown face