Face recognition

http://www.cs.cmu.edu/~16385/

16-385 Computer Vision
Spring 2018, Lecture 21

Course announcements

- Homework 6 has been posted and is due on April 27th.
    - Any questions about the homework?
    - How many of you have looked at/started/finished homework 6?

- VASC Seminar today: Burak Uzkent, “Object Detection and Tracking on Low Resolution Aerial Images”.
    - Monday 3-4pm, NSH 3305.

Overview of today's lecture

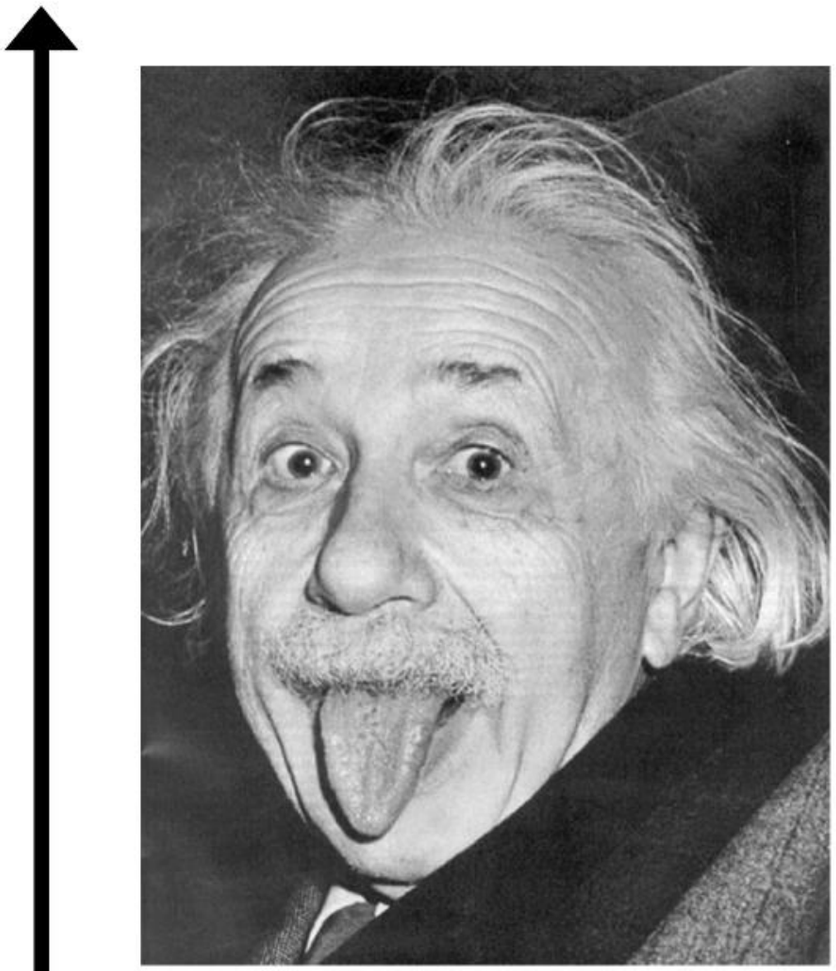
- Face recognition.
- Eigenfaces.
- Fisherfaces.
- Deep faces.
- Face detection.
- Boosting.
- Viola-Jones algorithm.

Slide credits

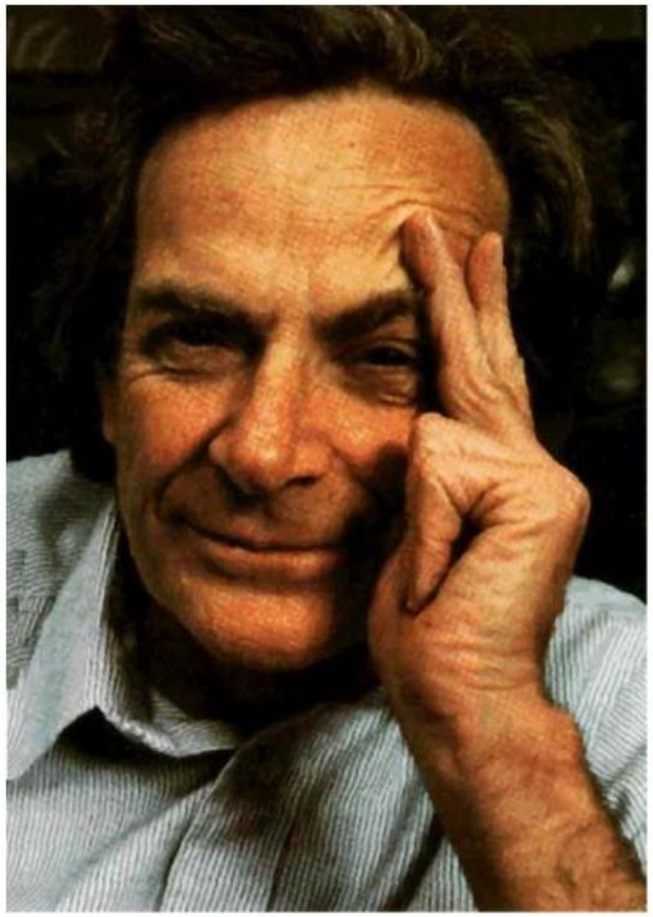
Most of these slides were adapted from:

- Kris Kitani (16-385, Spring 2017).
- Fei-Fei Li (Stanford University).
- Jia-Bin Huang (Virginia Tech).

What's 'recognition'?



vs.



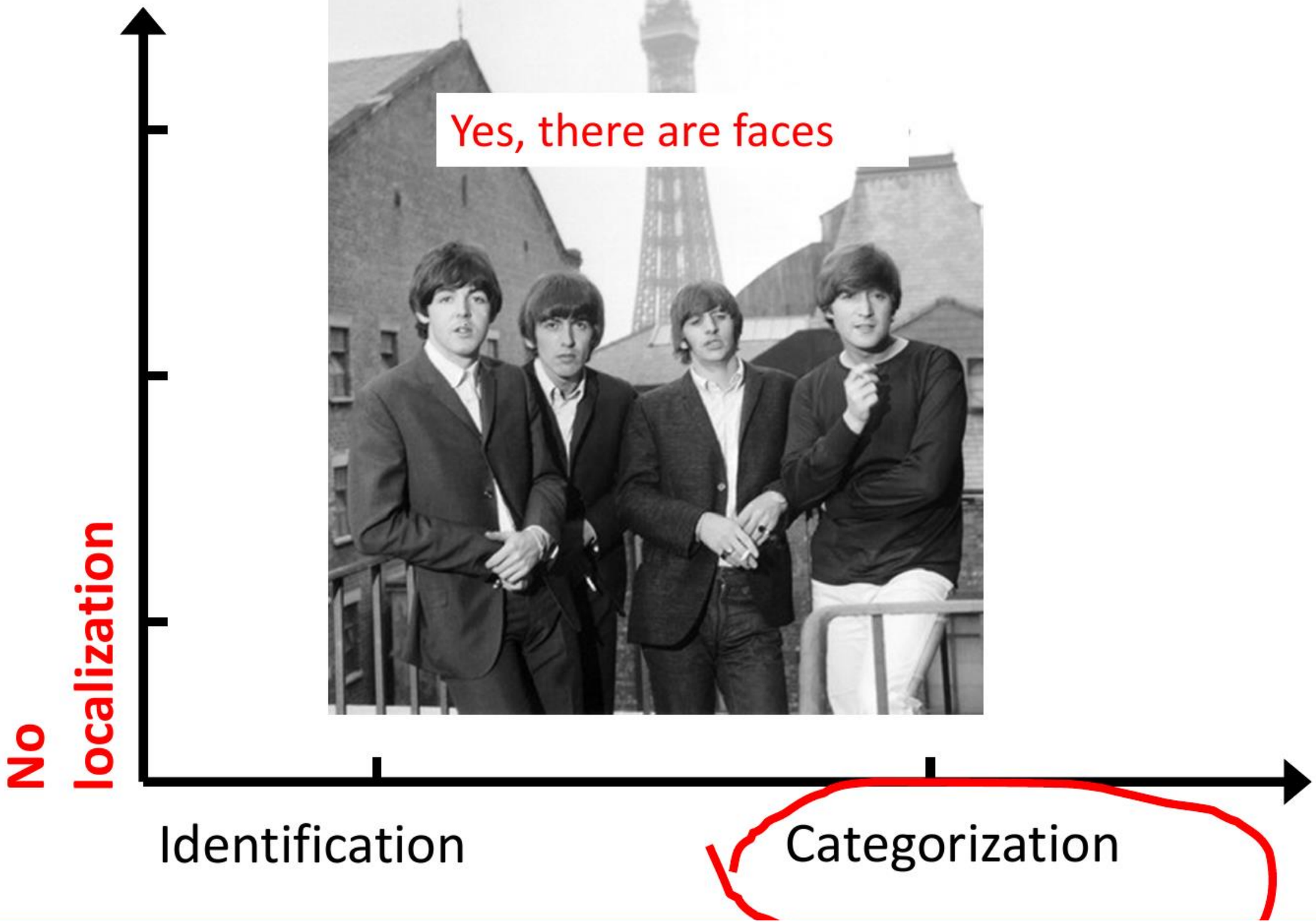
Identification

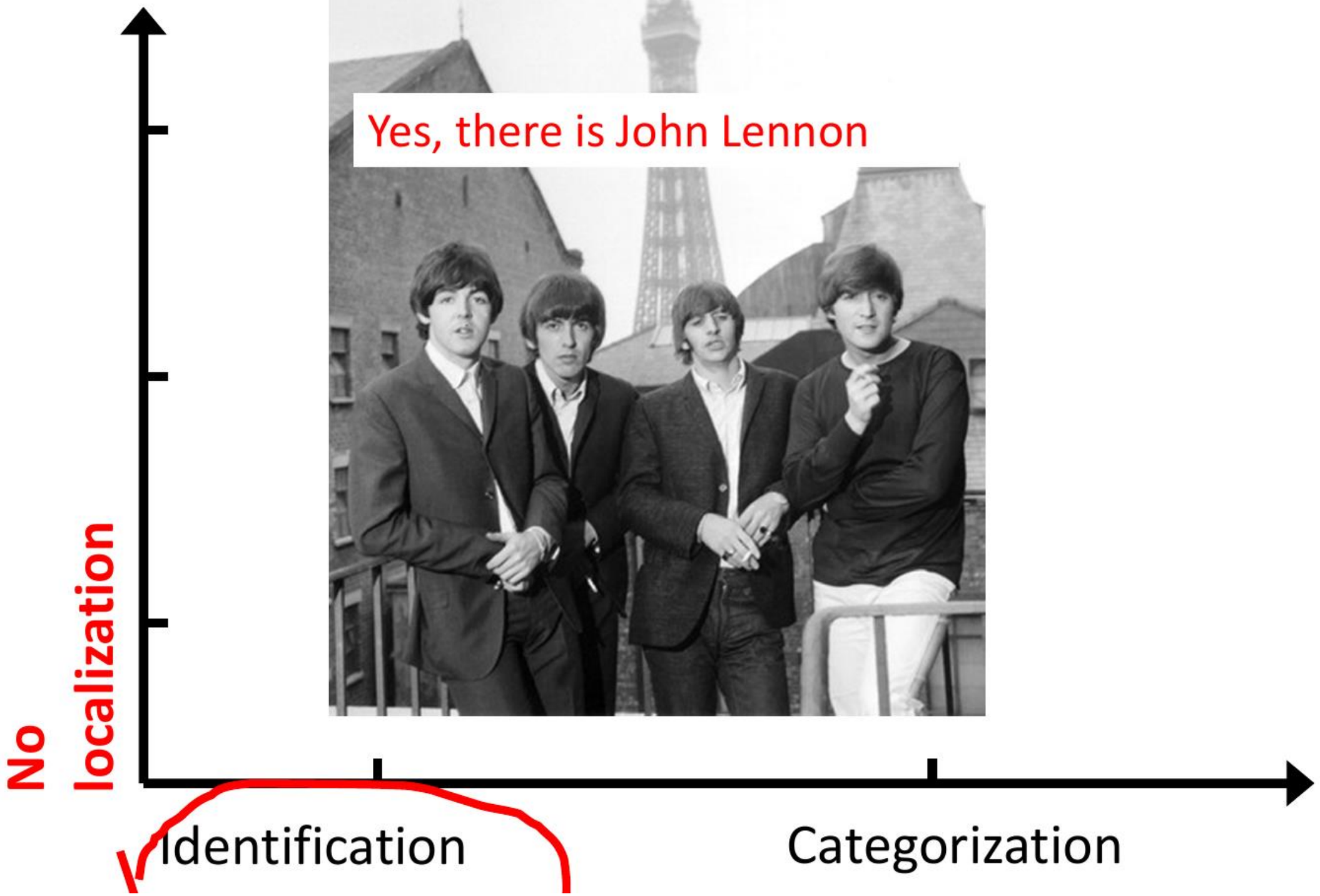
What's 'recognition'?

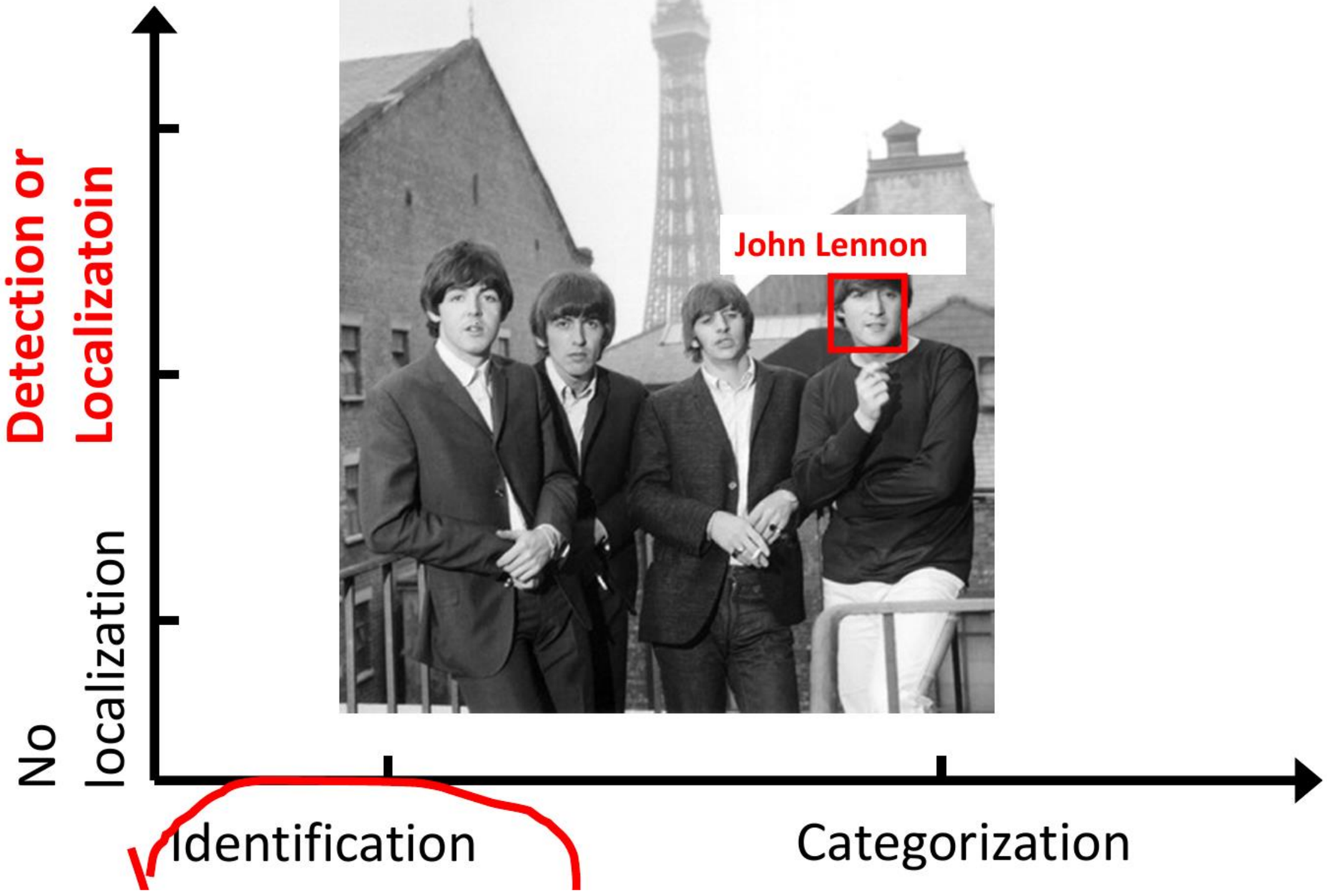
vs.

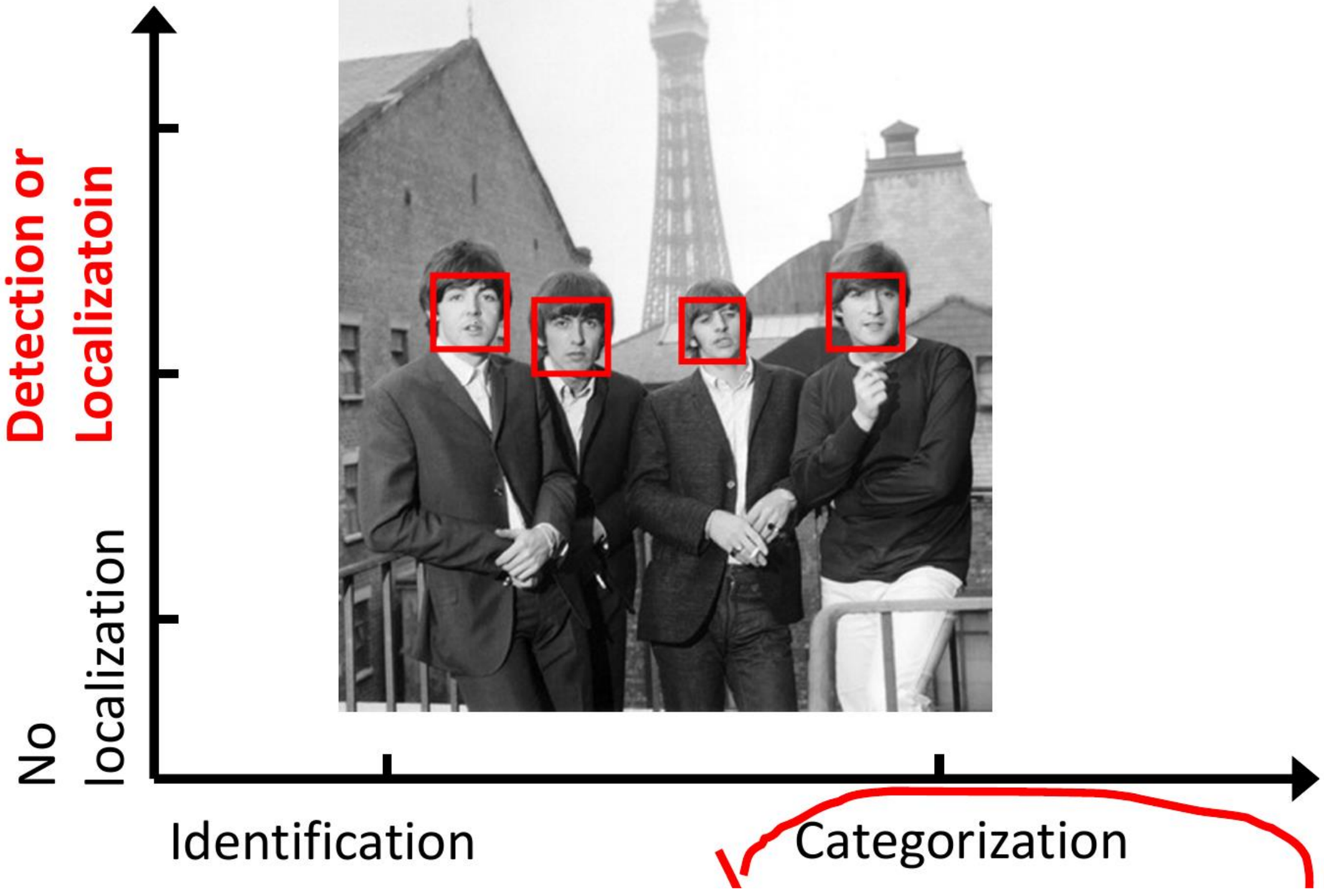
Identification

Categorization

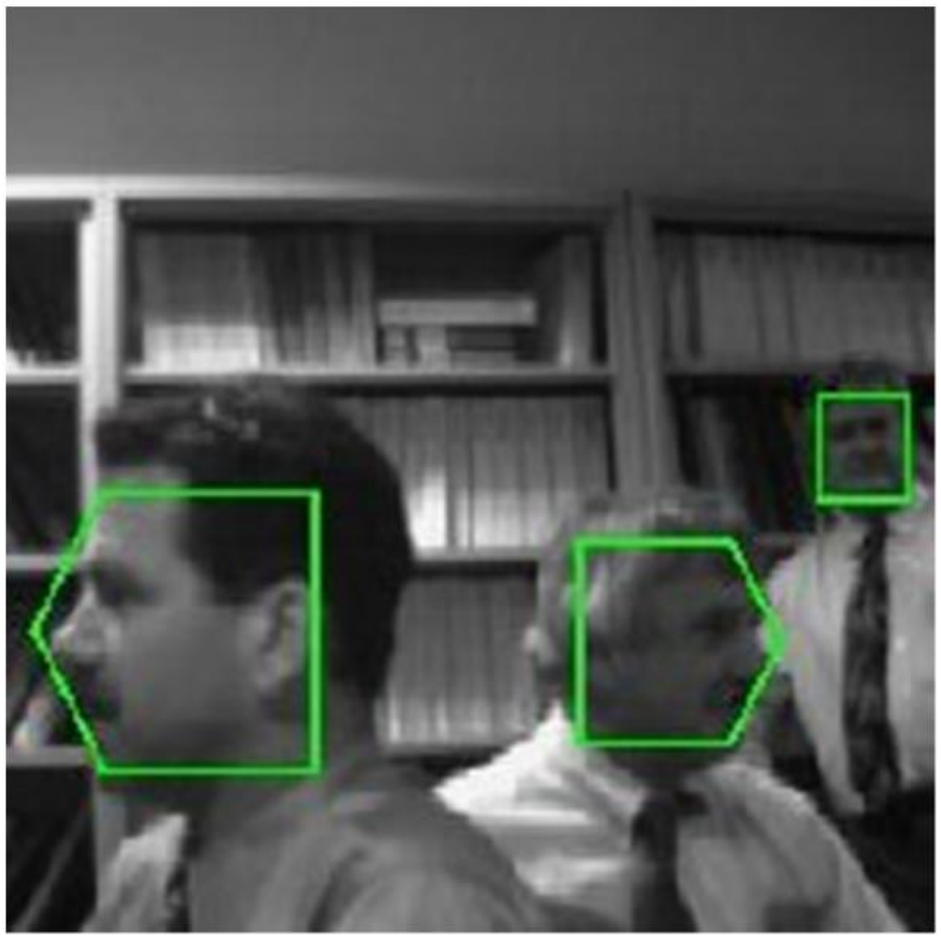

Yes, there are faces
No localization
Identification
Categorization


Yes, there is John Lennon
No localization
Identification
Categorization


John Lennon
Detection or
Localizatoin
No
localization
Identification
Categorization


Detection or
Localizatoin
No
localization
Identification
Categorization

Pose recognition



Detection or Localizatoin

No localization

Identification

Categorization

Face recognition: overview

- Typical scenario: few examples per face, identify or verify test example
- Why it's hard?
  - changes in expression, lighting, age, occlusion, **viewpoint**
- Basic approaches (all nearest neighbor)
  1. Project into a new subspace (or kernel space) (e.g., "Eigenfaces"=PCA)
  2. Measure face features
  3. Make 3d face model, compare shape+appearance, e.g., Active Appearance Model (AAM)

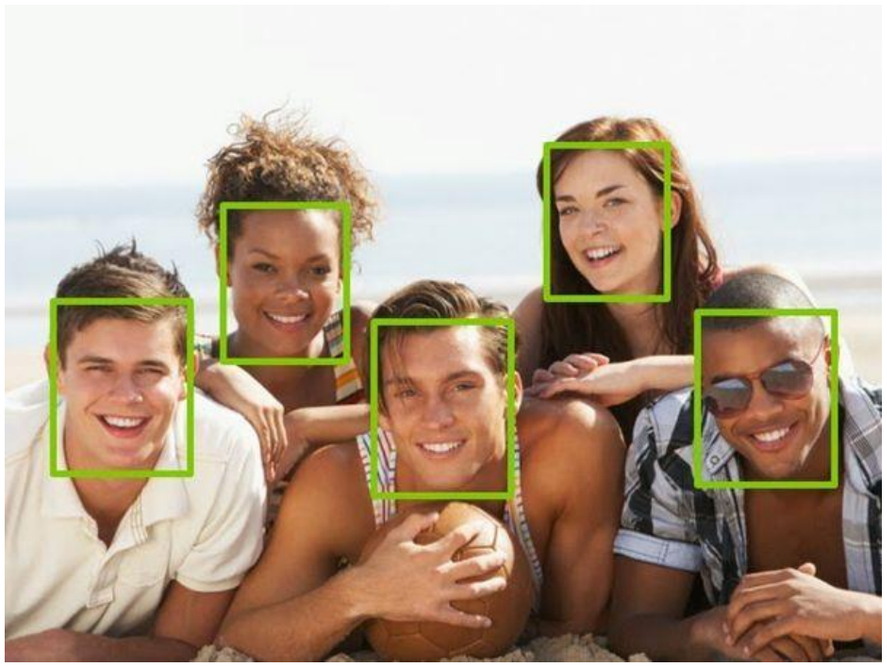
Typical face recognition scenarios

- **Verification**: a person is claiming a particular identity; verify whether that is true
  - E.g., security

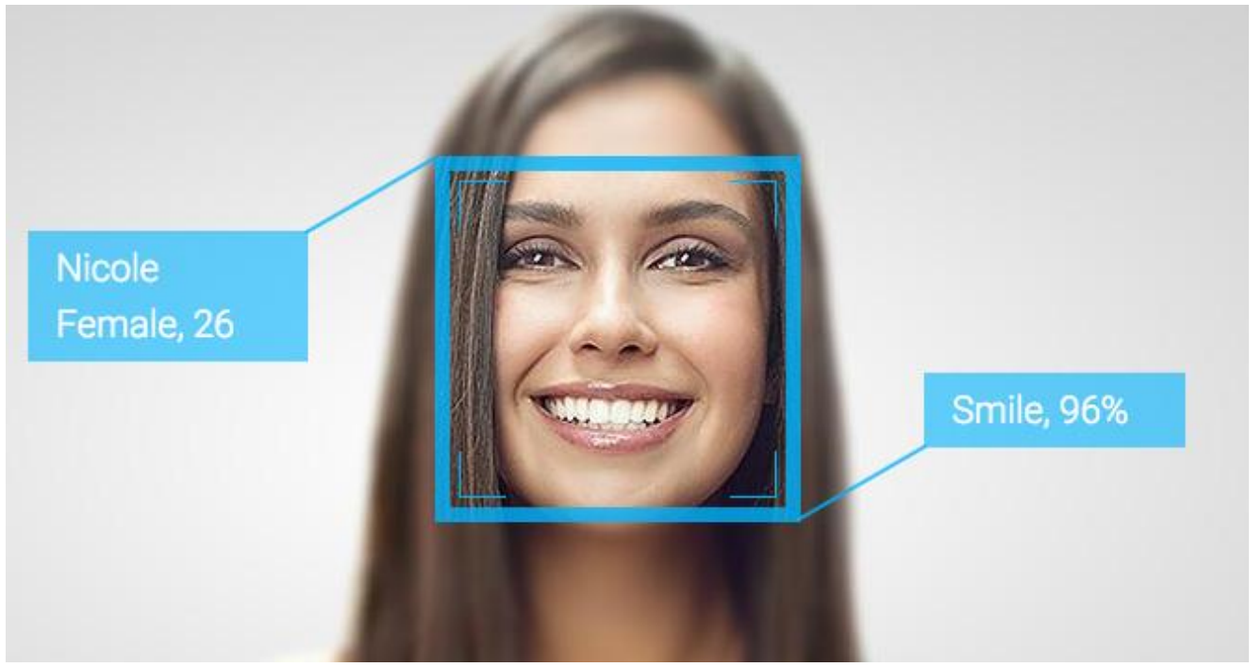
- **Closed-world identification**: assign a face to one person from among a known set

- **General identification**: assign a face to a known person or to “unknown”

Detection versus Recognition



Detection finds the faces in images



Recognition recognizes WHO the person is

Face Recognition

- 

Face Recognition

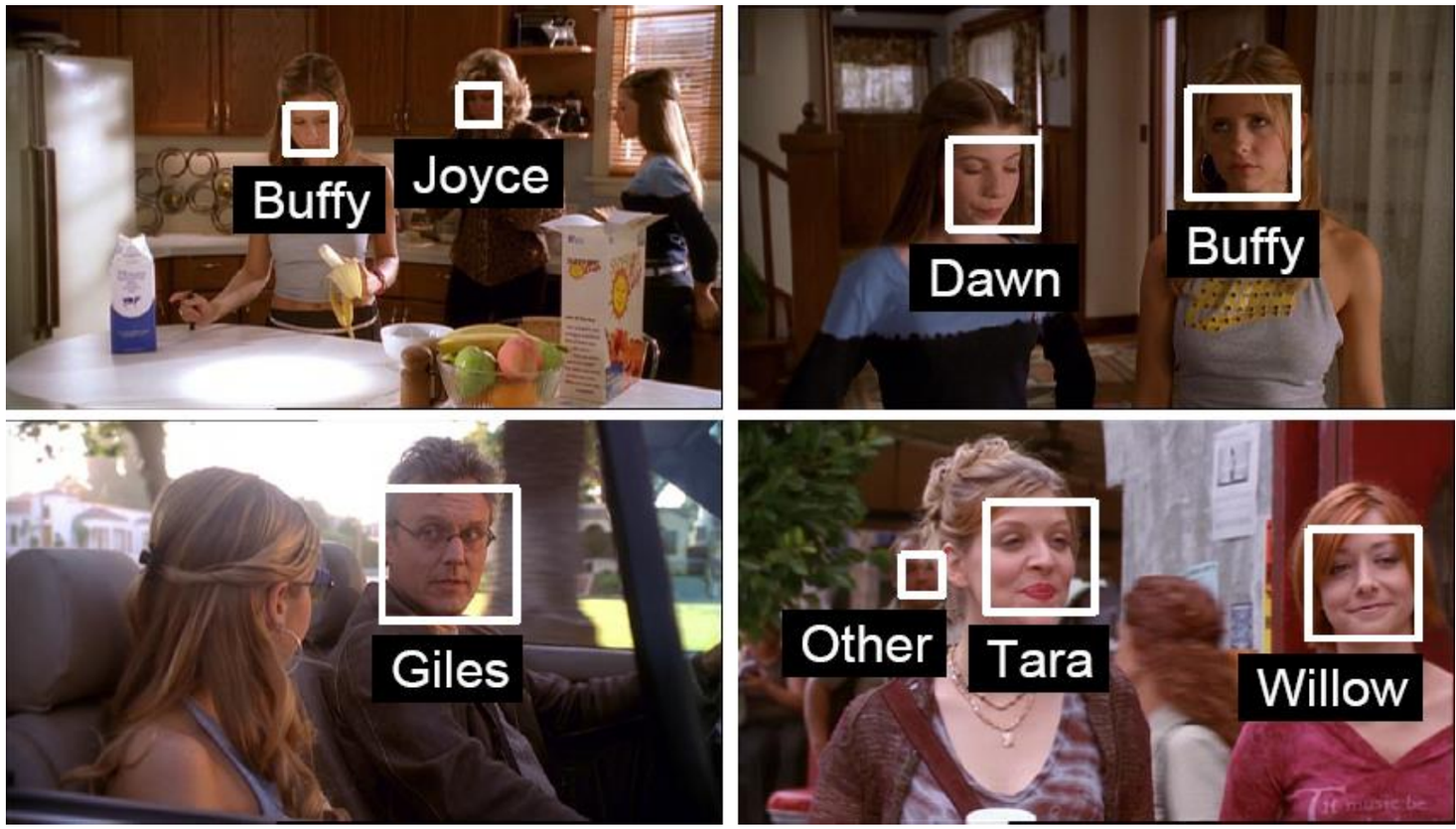
- Digital photogra
- Surveillance

Face Recognition

- Digital photography
- Surveillance
- Album organization

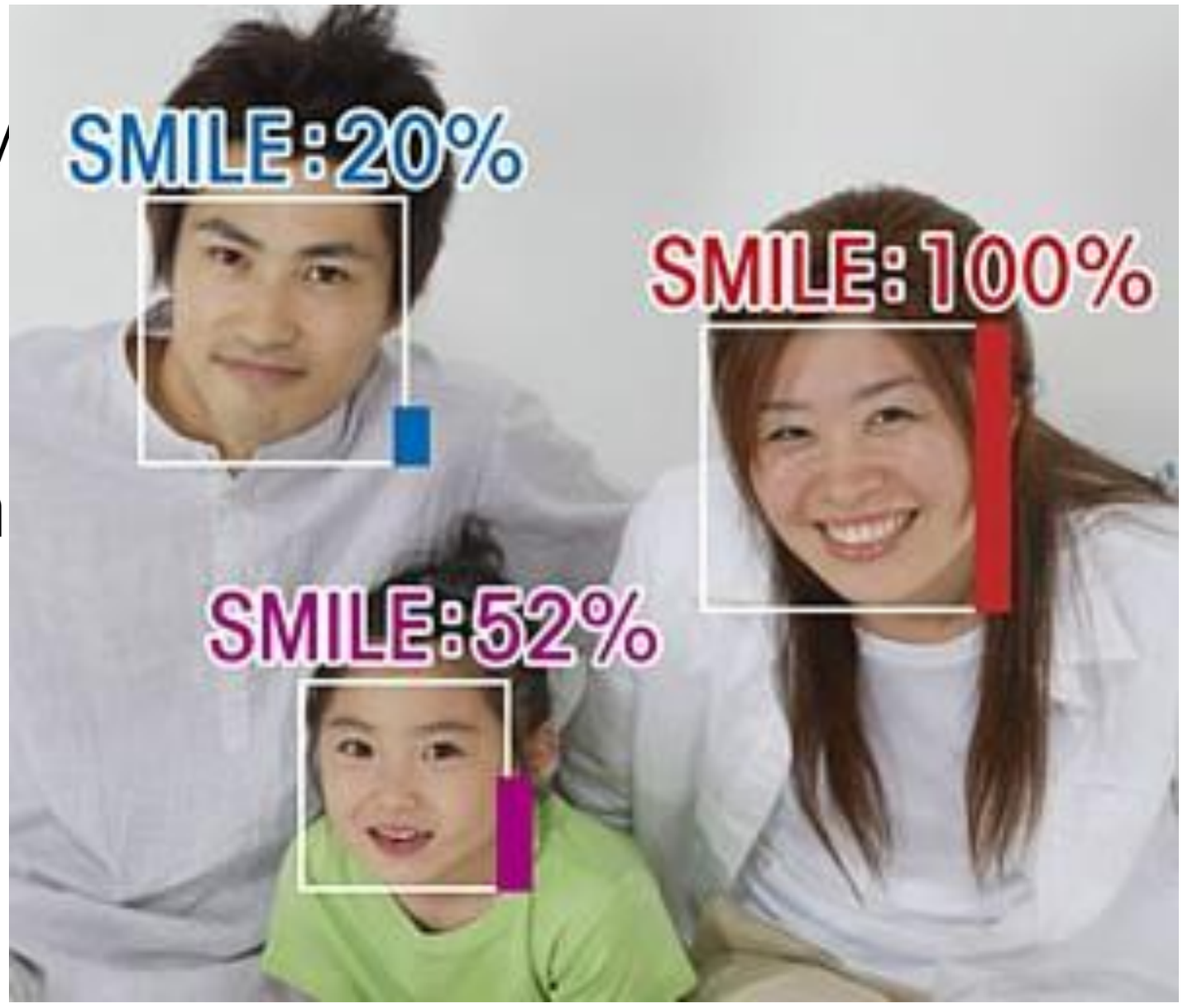
Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.



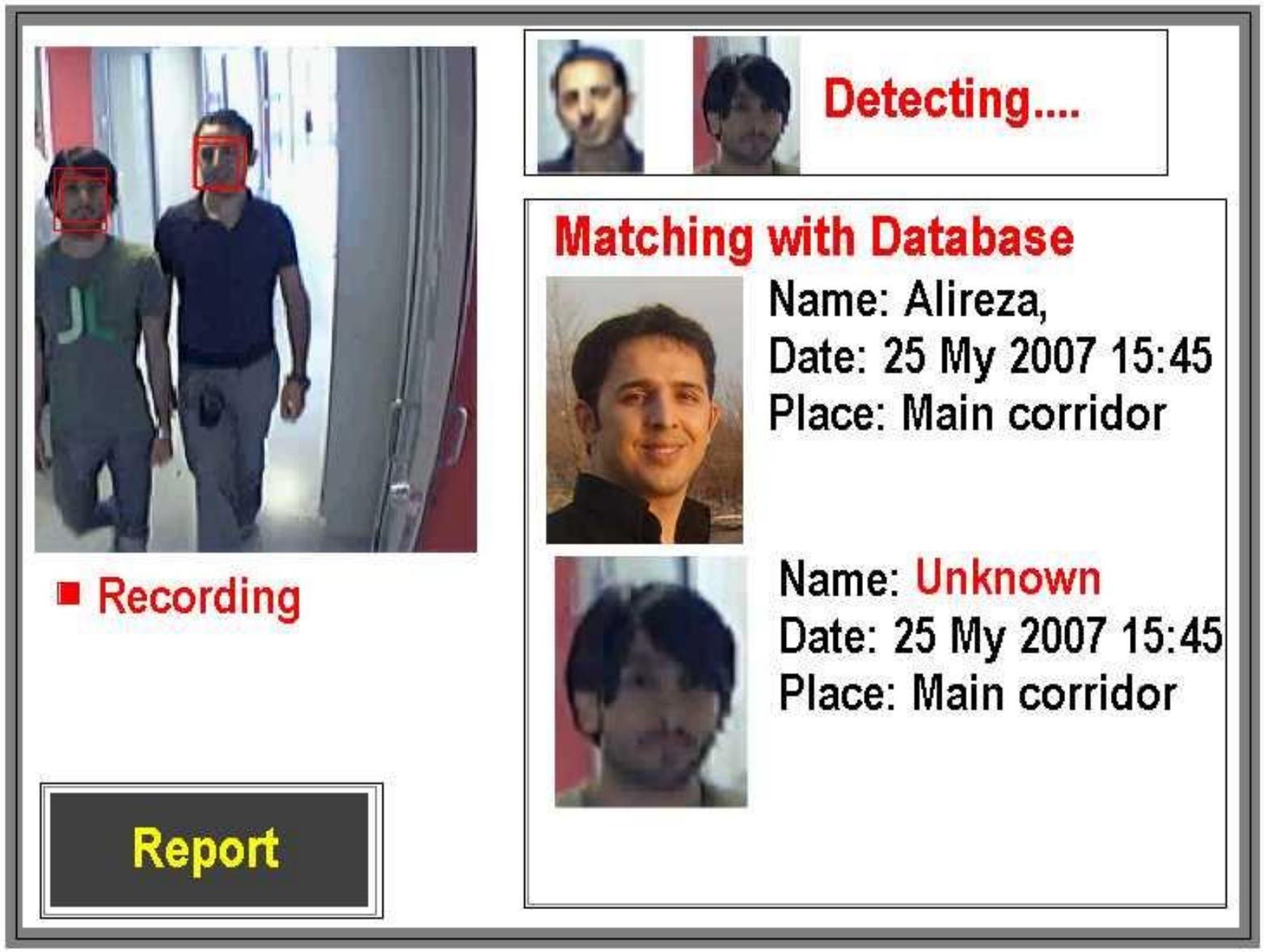
Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.
- Emotions and expressions

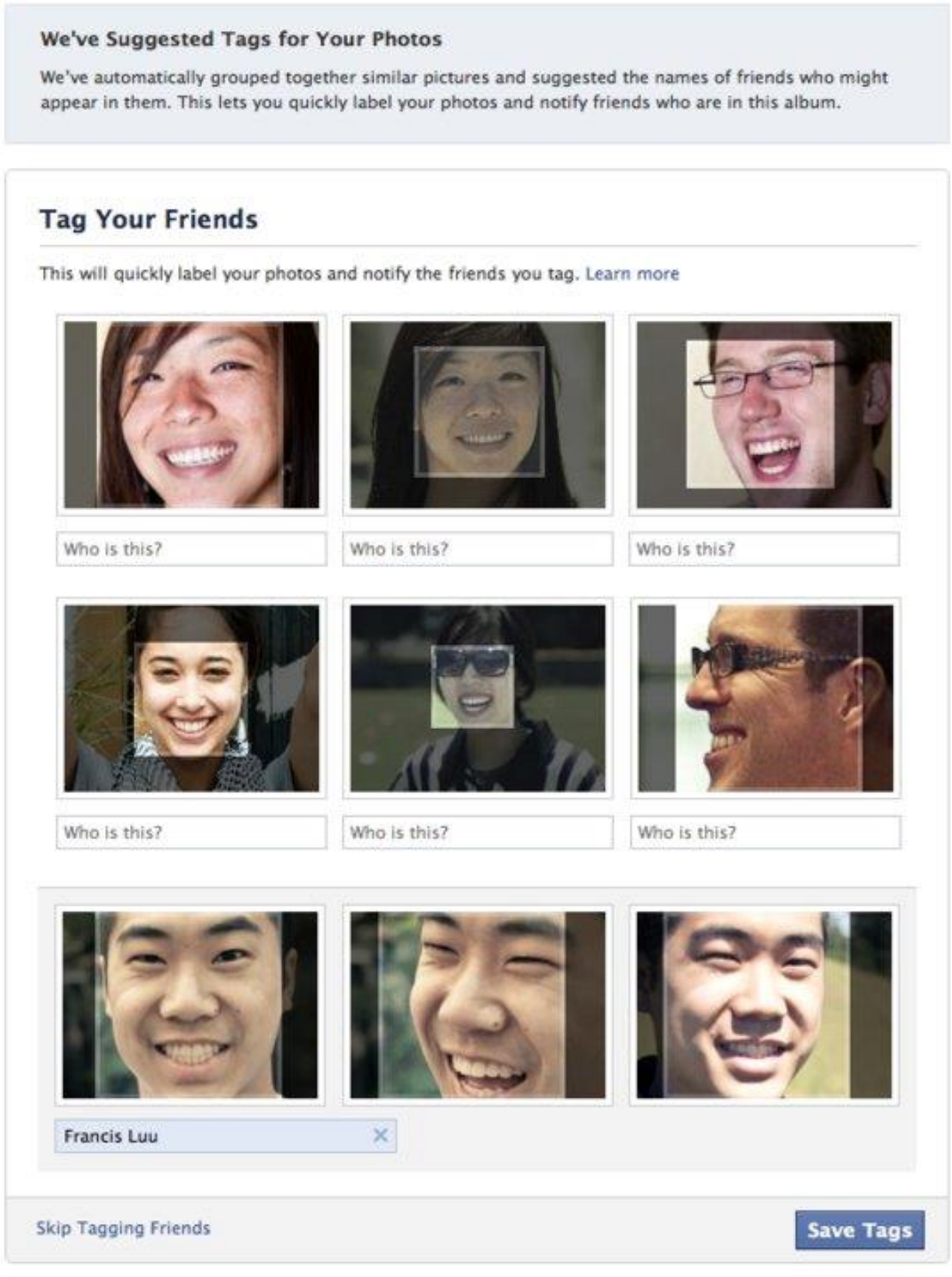


Applications of Face Recognition

- Surveillance



Facebook friend-tagging with auto-suggest



Face recognition: once you've detected and cropped a face, try to recognize it

Detection

Recognition

"Sally"

What makes face recognition hard?

Expression

What makes face recognition hard?

Lighting

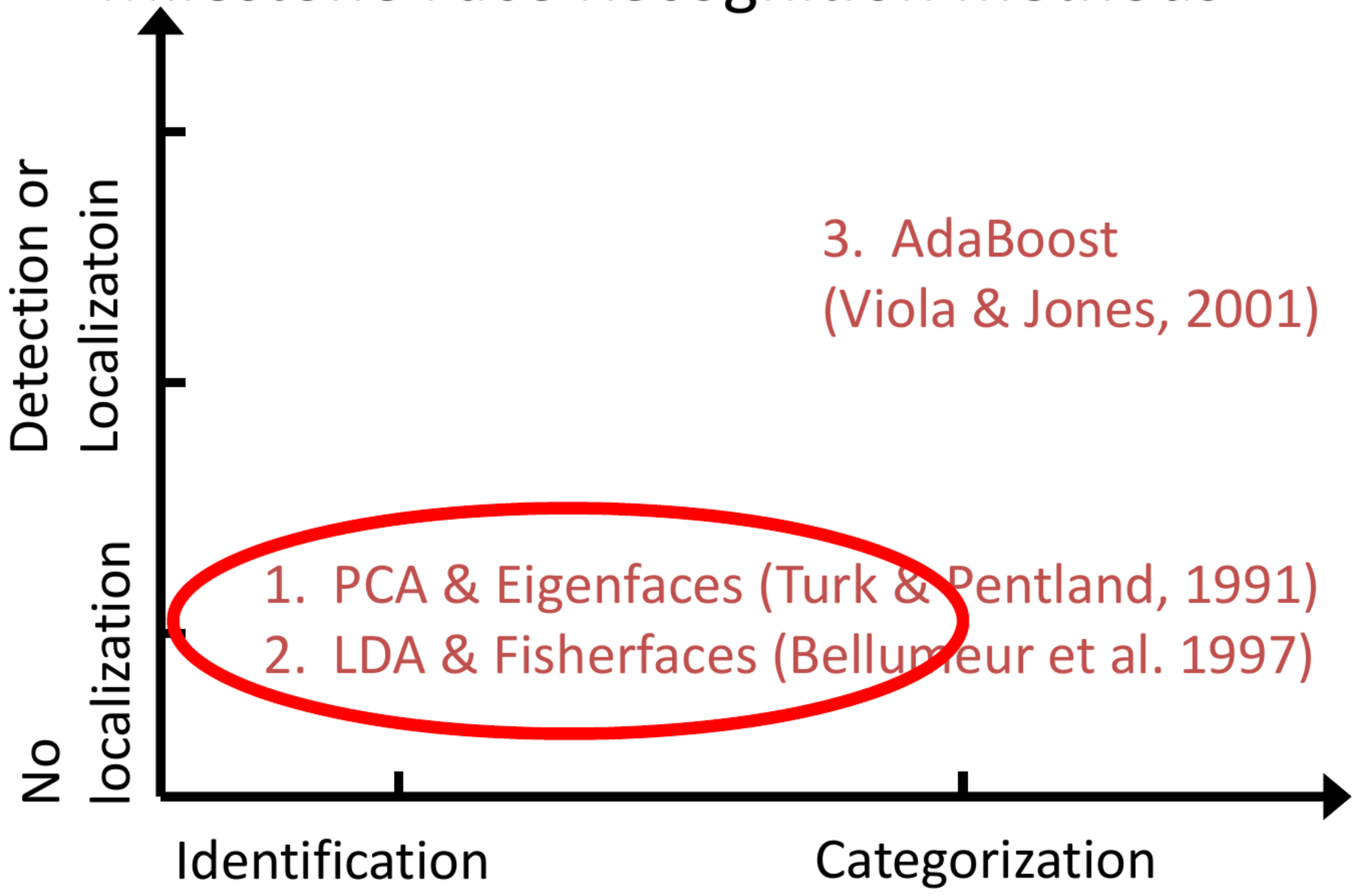
What makes face recognition hard?

Occlusion

What makes face recognition hard?

Viewpoint

Milestone Face Recognition methods



Simple idea for face recognition

1. Treat face image as a vector of intensities

$$\mathbf{x}$$

2. Recognize face by nearest neighbor in database

$$\mathbf{y}_1 \cdots \mathbf{y}_n$$

$$k = \operatorname*{argmin}_k \left\| \mathbf{y}_k - \mathbf{x} \right\|$$

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

100x100 images can contain many things other





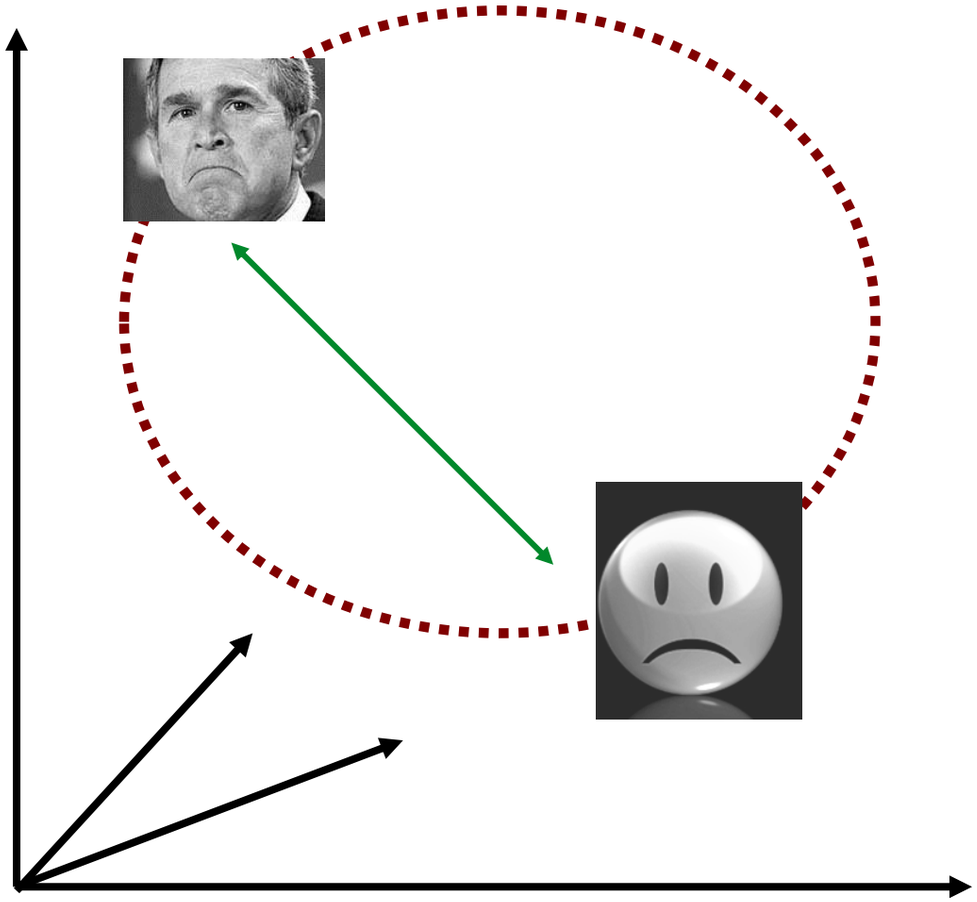








The Space of Faces



- An image is a point in a high dimensional space

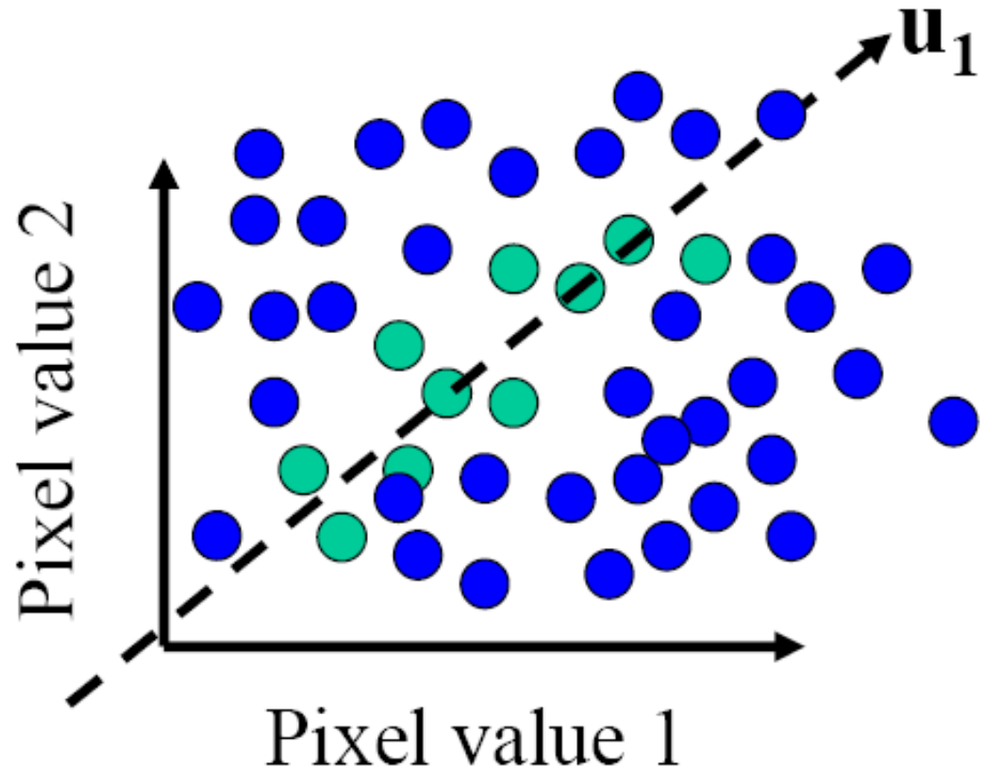
- If represented in grayscale intensity, an N x M image is a point in R^{NM}

- E.g. 100x100 image = 10,000 dim

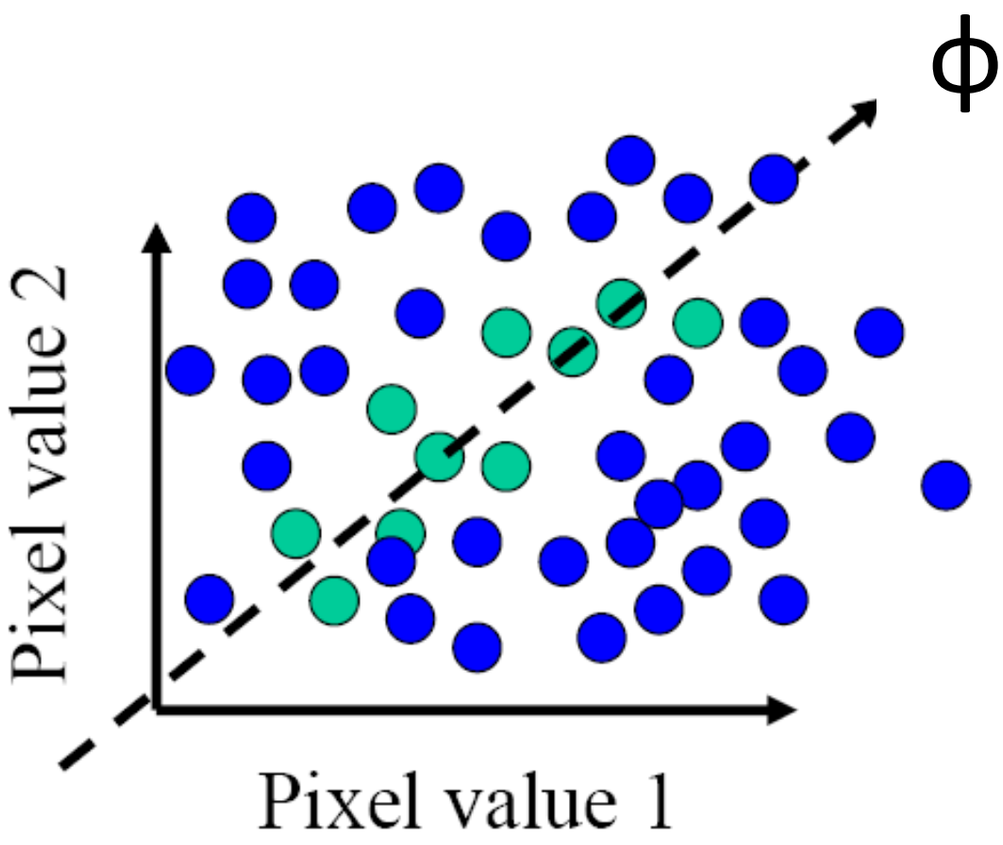
Slide credit: Chuck Dyer, Steve Seitz, Nishino

The space of all face images

- Eigenface idea: construct a low-dimensional linear subspace that best explains the variation in the set of face images



The Space of Faces



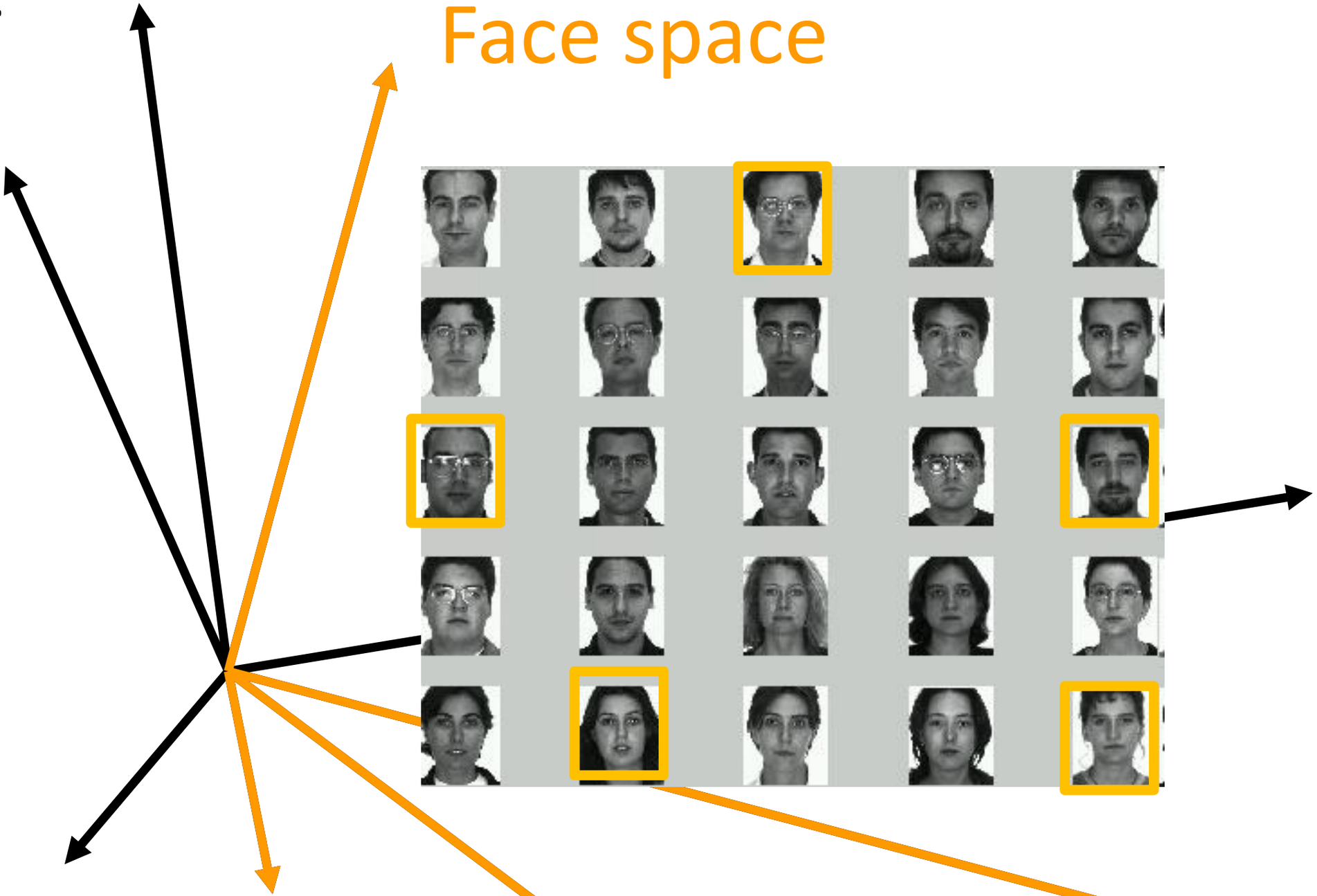
- A face image
- A (non-face) image

- An image is a point in a high dimensional space
  - If represented in grayscale intensity, an N x M image is a point in R^{NM}
  - E.g. 100x100 image = 10,000 dim
- However, relatively few high dimensional vectors correspond to valid face images
- We want to effectively model the subspace of face images

Slide credit: Chuck Dyer, Steve Seitz, Nishino

Image space

Face space



- Compute n-dim subspace such that the projection of the data points onto the subspace has the largest variance among all n-dim subspaces.
- Maximize the scatter of the training images in face space

Eigenfaces: key idea

- Assume that most face images lie on a low-dimensional subspace determined by the first k (k<<d) directions of maximum variance

- Use PCA to determine the vectors or “eigenfaces” that span that subspace

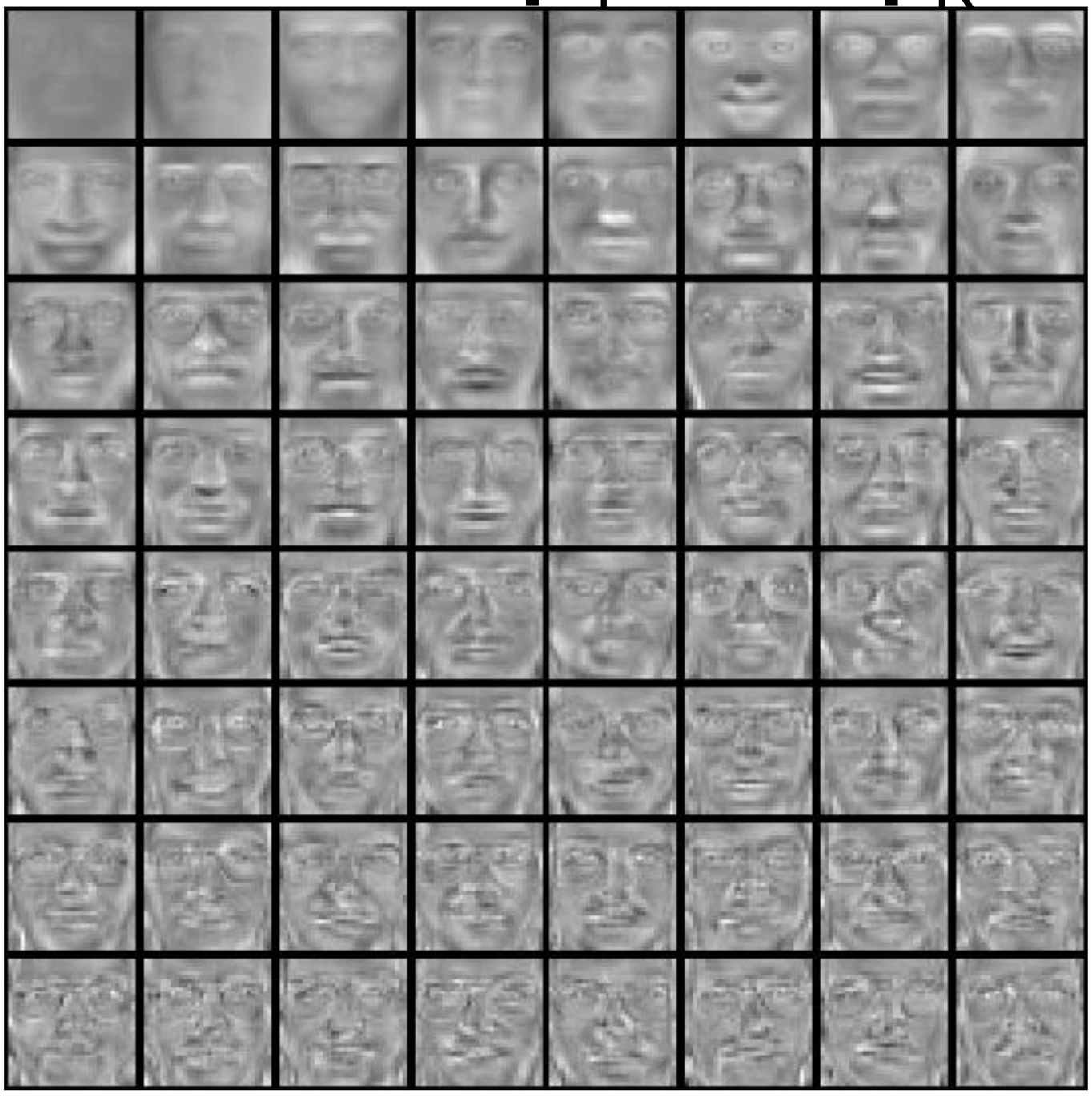
- Represent all face images in the dataset as linear combinations of eigenfaces

M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991

Training images: $\mathbf{x}_1, \ldots, \mathbf{x}_N$

Top eigenvectors: $\phi_1,\ldots,\phi_k$

Mean: μ



Visualization of eigenfaces

Principal component (eigenvector) ϕ_k

$\mu + 3\sigma_k\phi_k$

$\mu - 3\sigma_k\phi_k$

Eigenface algorithm

- Training

1. Align training images $x_1, x_2, \ldots, x_N$

Note that each image is formulated into a long vector!

1. Compute average face $m = \frac{1}{N}\sum_{i} x_i$

2. Compute the difference image (the centered data matrix)

$$X_c = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix} - \begin{bmatrix} | & & | \\ \mu & \cdots & \mu \\ | & & | \end{bmatrix}$$

$$= X - \mu 1^T = X - \frac{1}{n} X 1 1^T = X\left(I - \frac{1}{n} 1 1^T\right)$$

Eigenface algorithm

4. Compute the covariance matrix

$$\Sigma = \frac{1}{n}\begin{bmatrix} | & & | \\ x_1^c & \dots & x_n^c \\ | & & | \end{bmatrix}\begin{bmatrix} - & x_1^c & - \\ & \vdots & \\ - & x_n^c & - \end{bmatrix} = \frac{1}{n} X_c X_c^T$$

4. Compute the eigenvectors of the covariance matrix Σ

5. Compute each training image x_i ‘s projections as

$$x_i \to \left(x_i^c \cdot f_1, x_i^c \cdot f_2, \dots, x_i^c \cdot f_K\right) \equiv \left(a_1, a_2, \dots, a_K\right)$$

$$x_i \approx m + a_1 f_1 + a_2 f_2 + \dots + a_K f_K$$

6. Visualize the estimated training face x_i

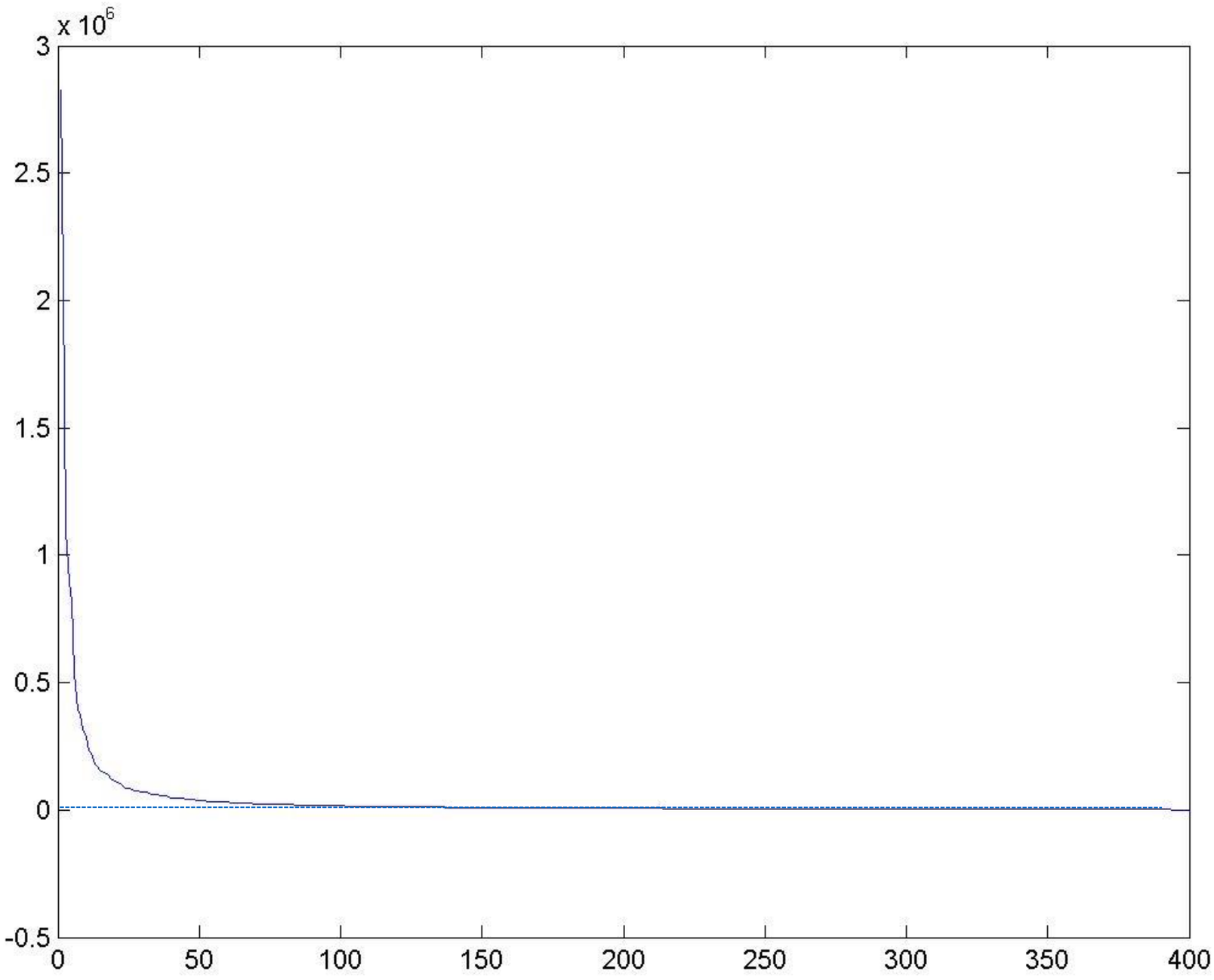
Eigenface algorithm

training face

$$x_i \to \left(x_i^c \cdot f_1, x_i^c \cdot f_2, \ldots, x_i^c \cdot f_K \right) \equiv \left(a_1, a_2, \ldots, a_K \right)$$

$$x_i \approx m + a_1 f_1 + a_2 f_2 + \ldots + a_K f_K$$

Eigenvalues (variance along eigenvectors)



Reconstruction and Errors

K = 4

K = 200

K = 400

- Only selecting the top K eigenfaces → reduces the dimensionality.
- Fewer eigenfaces result in more information loss, and hence less discrimination between faces.

Eigenface algorithm

- Testing

1. Take query image t

2. Project into eigenface space and compute projection

$$t \rightarrow \left((t - m) \cdot f_1, (t - m) \cdot f_2, \ldots, (t - m) \cdot f_K\right) \equiv \left(w_1, w_2, \ldots, w_K\right)$$

3. Compare projection w with all N training projections

   - Simple comparison metric: Euclidean

   - Simple decision: K-Nearest Neighbor

   (note: this “K” refers to the k-NN algorithm, different from the previous K’s referring to the # of principal components)

Shortcomings

- Requires carefully controlled data:
  - All faces centered in frame
  - Same size
  - Some sensitivity to angle
- Alternative:
  - “Learn” one set of PCA vectors for each angle
  - Use the one with lowest error
- Method is completely knowledge free
  - (sometimes this is good!)
  - Doesn’t know that faces are wrapped around 3D objects (heads)
  - Makes no effort to preserve class distinctions

Summary for Eigenface

Pros

- Non-iterative, globally optimal solution

Limitations

- PCA projection is **optimal for reconstruction** from a low dimensional basis, but **may NOT be optimal for discrimination…**

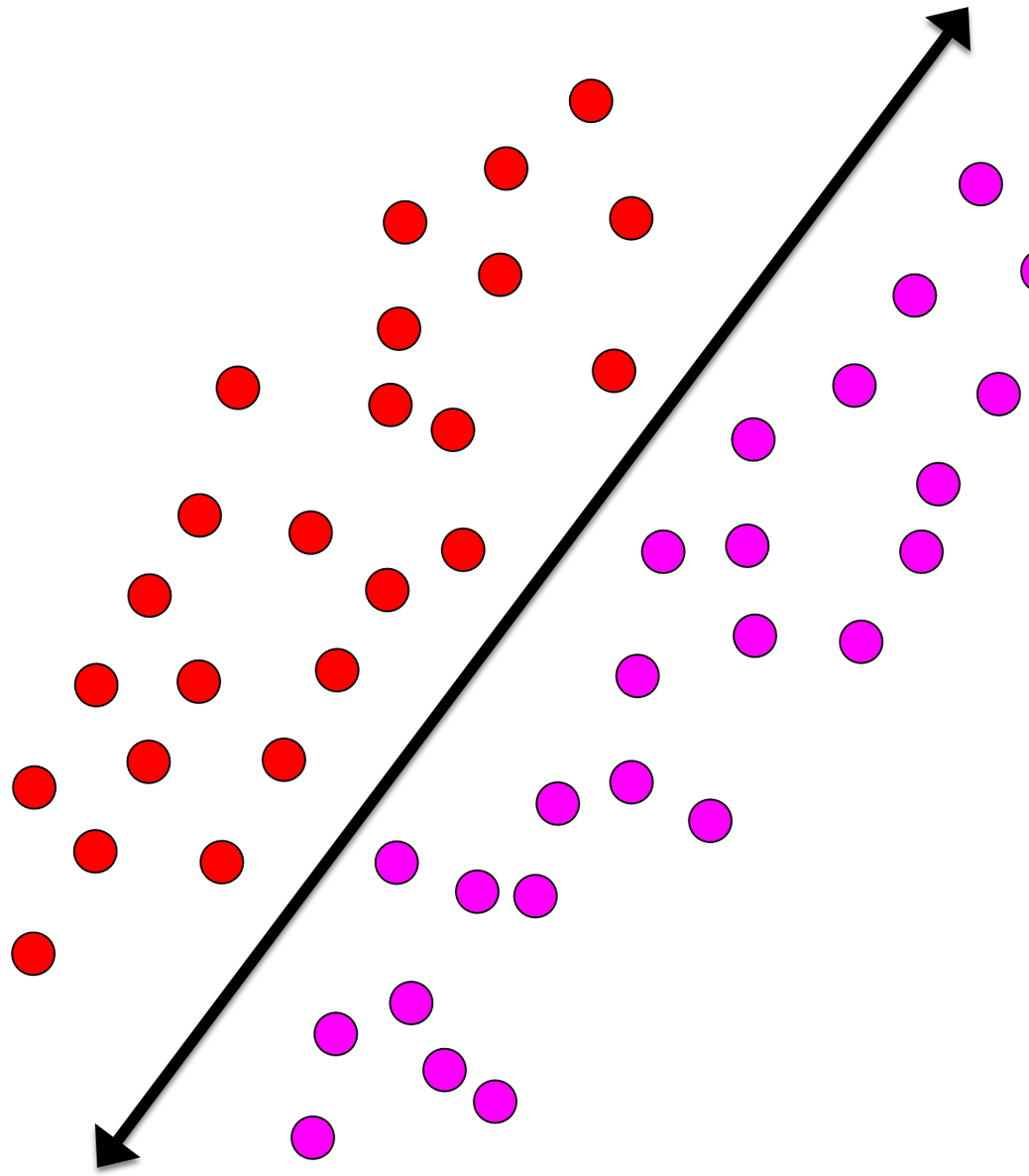
Besides face recognitions, we can also do

Facial expression recognition

Happiness subspace

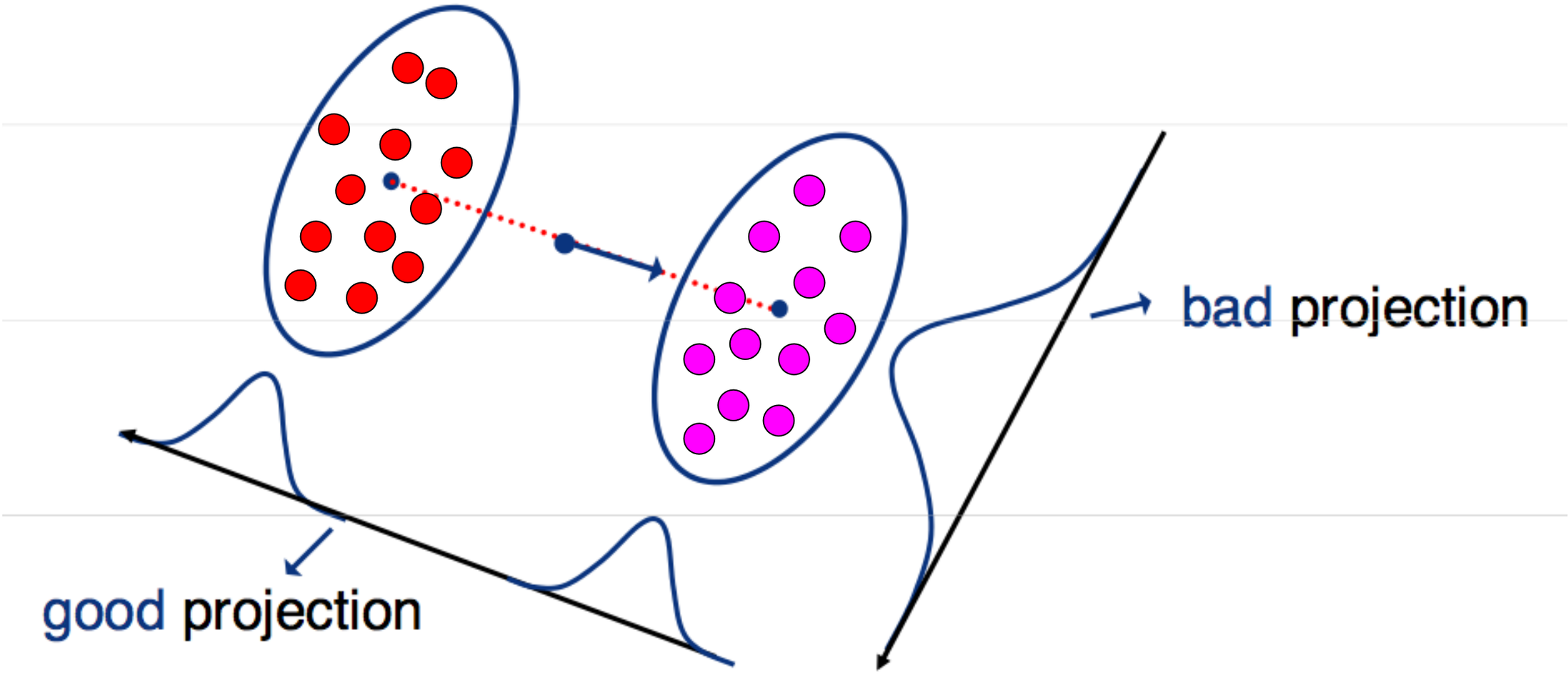
Disgust subspace

Which direction will is the first principle component?



Fischer's Linear Discriminant Analysis

- Goal: find the best separation between two classes



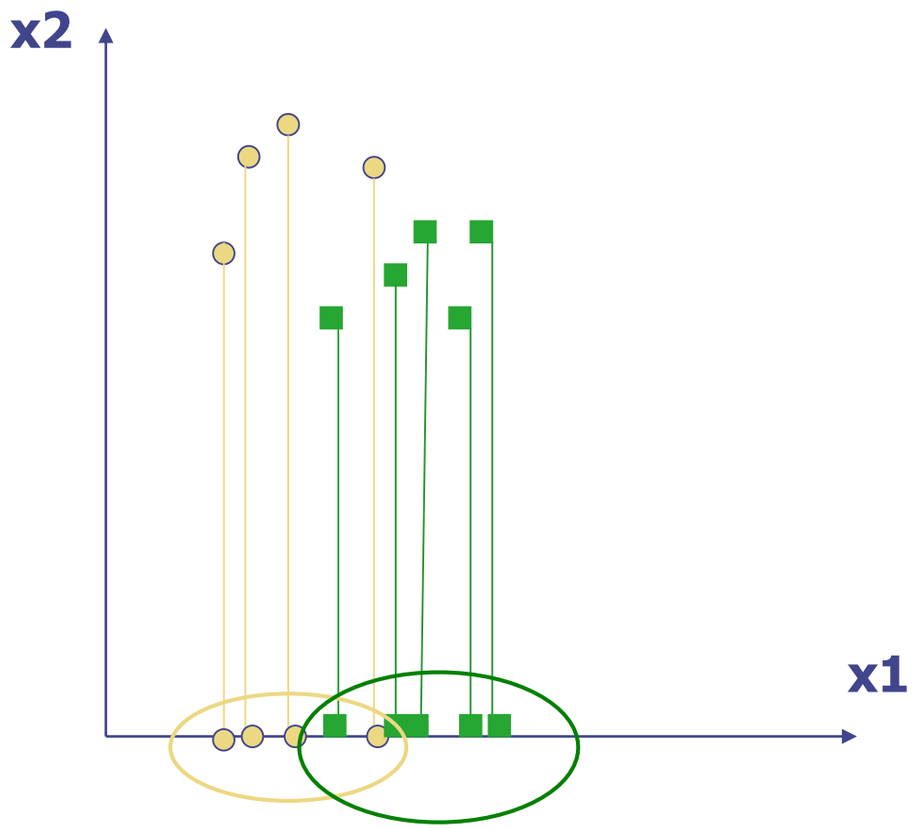
Difference between PCA and LDA

- PCA preserves maximum variance

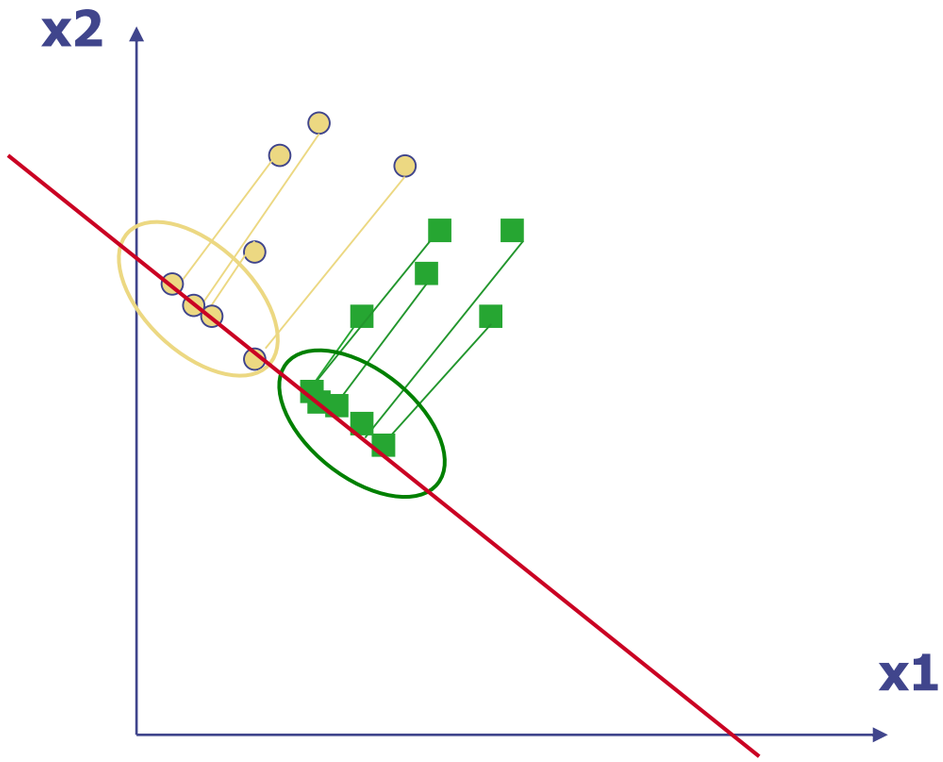
- LDA preserves discrimination
  - Find projection that maximizes scatter between classes and minimizes scatter within classes

Illustration of the Projection

- Using two classes as example:

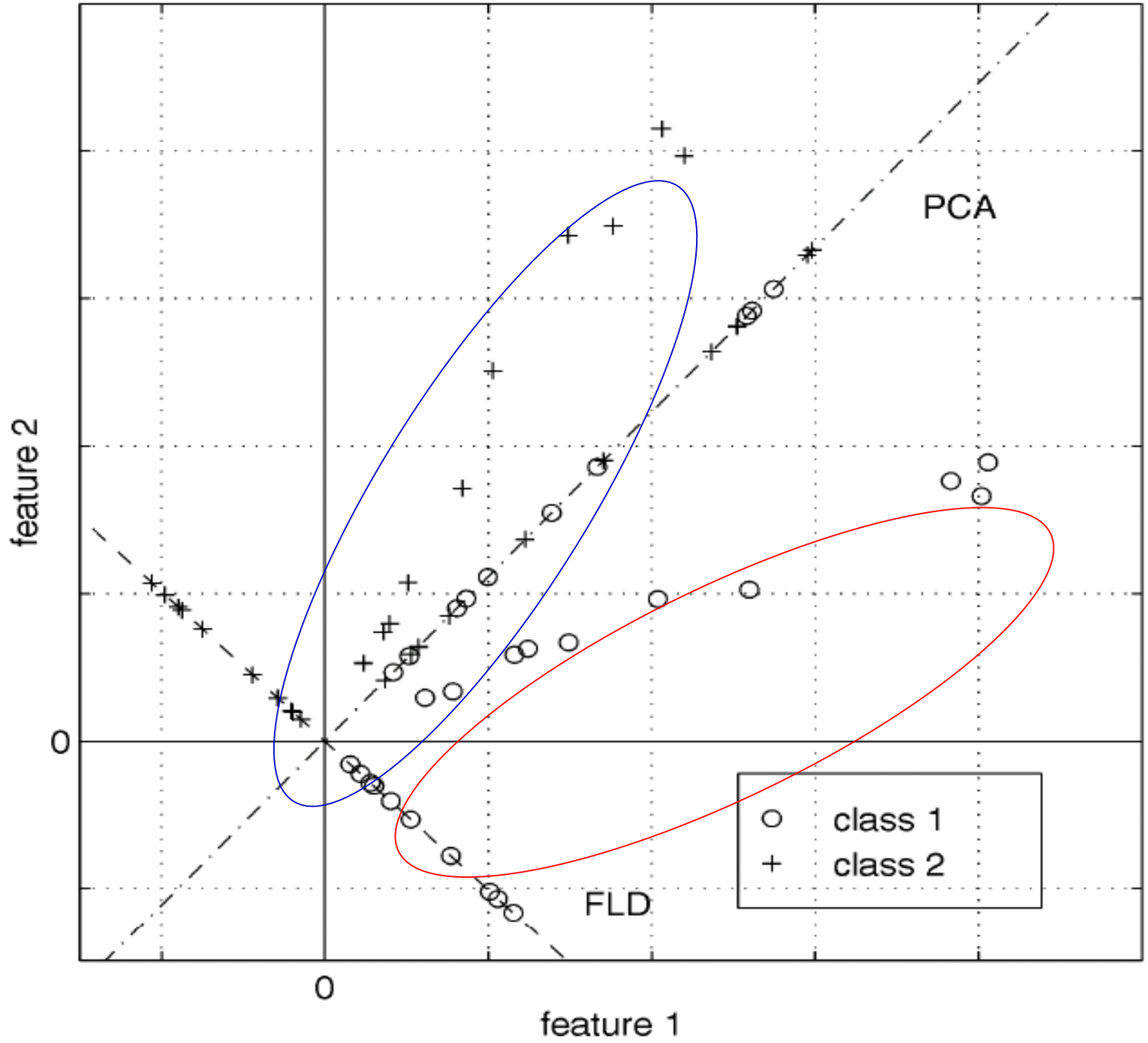


Poor Projection



Good

Basic intuition: PCA vs. LDA



LDA with 2 variables

- We want to learn a projection W such that the projection converts all the points from x to a new space (For this example, assume m == 1):

$$z = w^T x \qquad z \in \mathbf{R}^m \quad x \in \mathbf{R}^n$$

- Let the **per class** means be:

$$E_{X|Y}[X \mid Y = i] = \mu_i$$

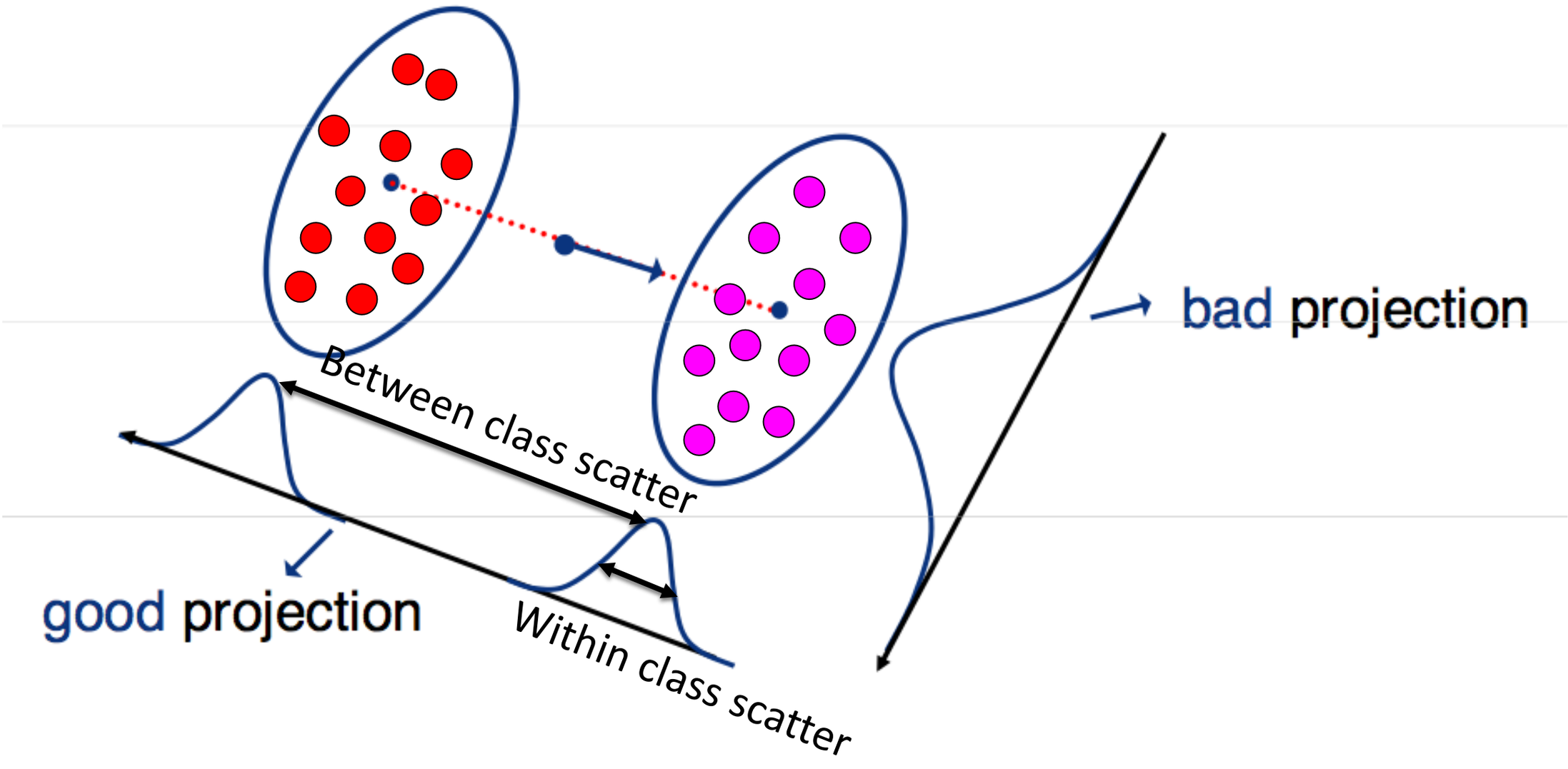
- And the **per class** covariance matrices be:

$$E_{X|Y}[(X - \mu_i)(X - \mu_i)^T \mid Y = i] = \Sigma_i$$

- We want a projection that maximizes:

$$J(w) = \max \frac{between\ \ class\ \ scatter}{within\ \ class\ \ scatter}$$

Fischer's Linear Discriminant Analysis



LDA with 2 variables

The following objective function:

$$J(w) = \max \frac{between\ \ class\ \ scatter}{within\ \ class\ \ scatter}$$

Can be written as

$$J(w) = \frac{\left(E_{Z|Y}\left[Z \mid Y=1\right] - E_{Z|Y}\left[Z \mid Y=0\right]\right)^2}{\mathrm{var}\left[Z \mid Y=1\right] + \mathrm{var}\left[Z \mid Y=0\right]}$$

LDA with 2 variables

- We can write the between class scatter as:

$$\left(E_{Z|Y}\left[Z \mid Y=1\right]-E_{Z|Y}\left[Z \mid Y=0\right]\right)^2=\left(w^T\left[\mu_1-\mu_0\right]\right)^2$$
$$=w^T\left[\mu_1-\mu_0\right]\left[\mu_1-\mu_0\right]^T w$$

- Also, the within class scatter becomes:

$$\operatorname{var}\left[Z \mid Y=i\right]=E_{Z|Y}\left\{\left(z-E_{Z|Y}\left[Z \mid Y=i\right]\right)^2 \mid Y=i\right\}$$
$$=E_{Z|Y}\left\{\left(w^T\left[x-\mu_i\right]\right)^2 \mid Y=i\right\}$$
$$=E_{Z|Y}\left\{w^T\left[x-\mu_i\right]\left[x-\mu_i\right]^T w \mid Y=i\right\}$$
$$=w^T\Sigma_i w$$

LDA with 2 variables

- We can plug in these scatter values to our objective function:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

between class scatter

$$S_B = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T$$

$$S_W = (\Sigma_1 + \Sigma_0)$$

within class scatter

- And our objective becomes:

$$J(w) = \frac{\left(E_{Z|Y}[Z \mid Y=1] - E_{Z|Y}[Z \mid Y=0]\right)^2}{\mathrm{var}[Z \mid Y=1] + \mathrm{var}[Z \mid Y=0]}$$

$$= \frac{w^T (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T (\Sigma_1 + \Sigma_0) w}$$

LDA with 2 variables

- The scatter variables

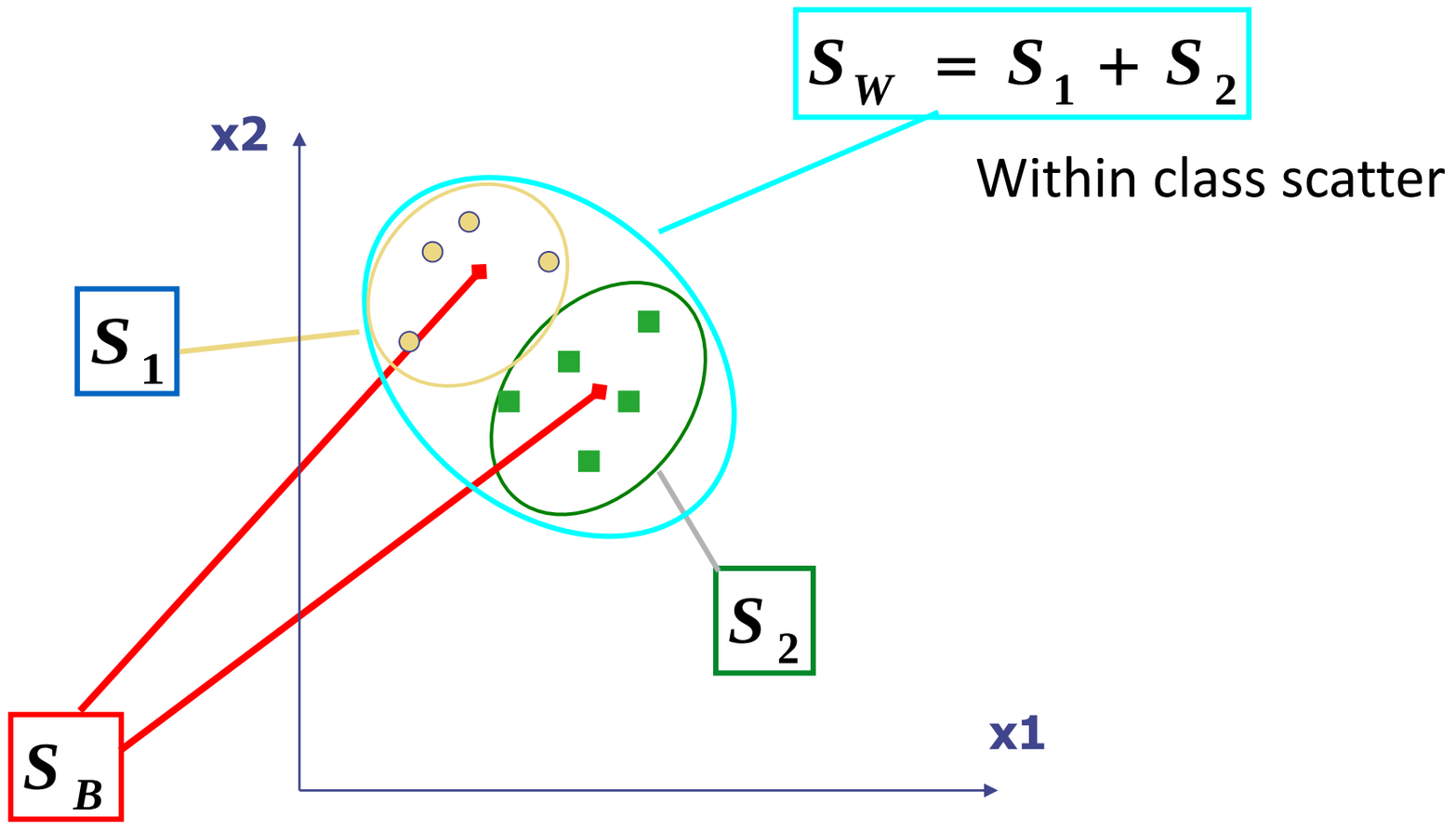
between class scatter

$$S_B = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T$$

$$S_W = (\Sigma_1 + \Sigma_0)$$

within class scatter

Visualization



Between class scatter

Linear Discriminant Analysis (LDA)

- Maximizing the ratio

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

- Is equivalent to maximizing the numerator while keeping the denominator constant, i.e.

$$\max_w w^T S_B w \quad \text{subject to} \quad w^T S_W w = K$$

- And can be accomplished using Lagrange multipliers, where we define the Lagrangian as

$$L = w^T S_B w - \lambda\left(w^T S_W w - K\right)$$

- And maximize with respect to both w and λ

Linear Discriminant Analysis (LDA)

- Setting the gradient of

$$L = w^T \left(S_B - \lambda S_W \right) w + \lambda K$$

With respect to w to zeros we get

$$\nabla_w L = 2\left(S_B - \lambda S_W \right) w = 0$$

$$\boxed{S_B w = \lambda S_W w}$$

- This is a generalized eigenvalue problem
- The solution is easy when $\boxed{S_w^{-1} = \left(\Sigma_1 + \Sigma_0 \right)^{-1}}$ exists

Linear Discriminant Analysis (LDA)

- In this case

$$S_W^{-1} S_B w = \lambda w$$

- And using the definition of S_B

$$S_W^{-1}(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w = \lambda w$$

- Noting that $(\mu_1-\mu_0)^T w=\alpha$ is a scalar this can be written as

$$S_W^{-1}(\mu_1 - \mu_0) = \frac{\lambda}{\alpha} w$$

- and since we don't care about the magnitude of w

$$w^* = S_W^{-1}(\mu_1 - \mu_0) = (\Sigma_1 + \Sigma_0)^{-1}(\mu_1 - \mu_0)$$

LDA with N variables and C classes

Variables

- N Sample images: $\{x_1, \cdots, x_N\}$
- C classes: $\{Y_1, Y_2, \ldots, Y_c\}$
- Average of each class: $\mu_i = \frac{1}{N_i} \sum_{x_k \in Y_i} x_k$
- Average of all data: $\mu = \frac{1}{N} \sum_{k=1}^{N} x_k$

Scatter Matrices

- Scatter of class i: $$S_i = \sum_{x_k \in Y_i} (x_k - \mu_i)(x_k - \mu_i)^T$$
- Within class scatter: $$S_W = \sum_{i=1}^{c} S_i$$
- Between class scatter: $$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

Mathematical Formulation

- Recall that we want to learn a projection W such that the projection converts all the points from x to a new space z:

$$z = w^T x \qquad z \in \mathbf{R}^m \quad x \in \mathbf{R}^n$$

- After projection:
  - Between class scatter $\tilde{S}_B = W^T S_B W$
  - Within class scatter $\tilde{S}_W = W^T S_W W$
- So, the objective becomes:

$$W_{opt} = \arg\max_{\mathbf{w}} \frac{\left|\tilde{S}_B\right|}{\left|\tilde{S}_W\right|} = \arg\max_{\mathbf{w}} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$$

Mathematical Formulation

$$W_{opt} = \arg \max_{W} \frac{\left| W^T S_B W \right|}{\left| W^T S_W W \right|}$$

- Solve generalized eigenvector problem:

$$S_B w_i = \lambda_i S_W w_i \qquad i = 1, \ldots, m$$

Mathematical Formulation

- Solution: Generalized Eigenvectors

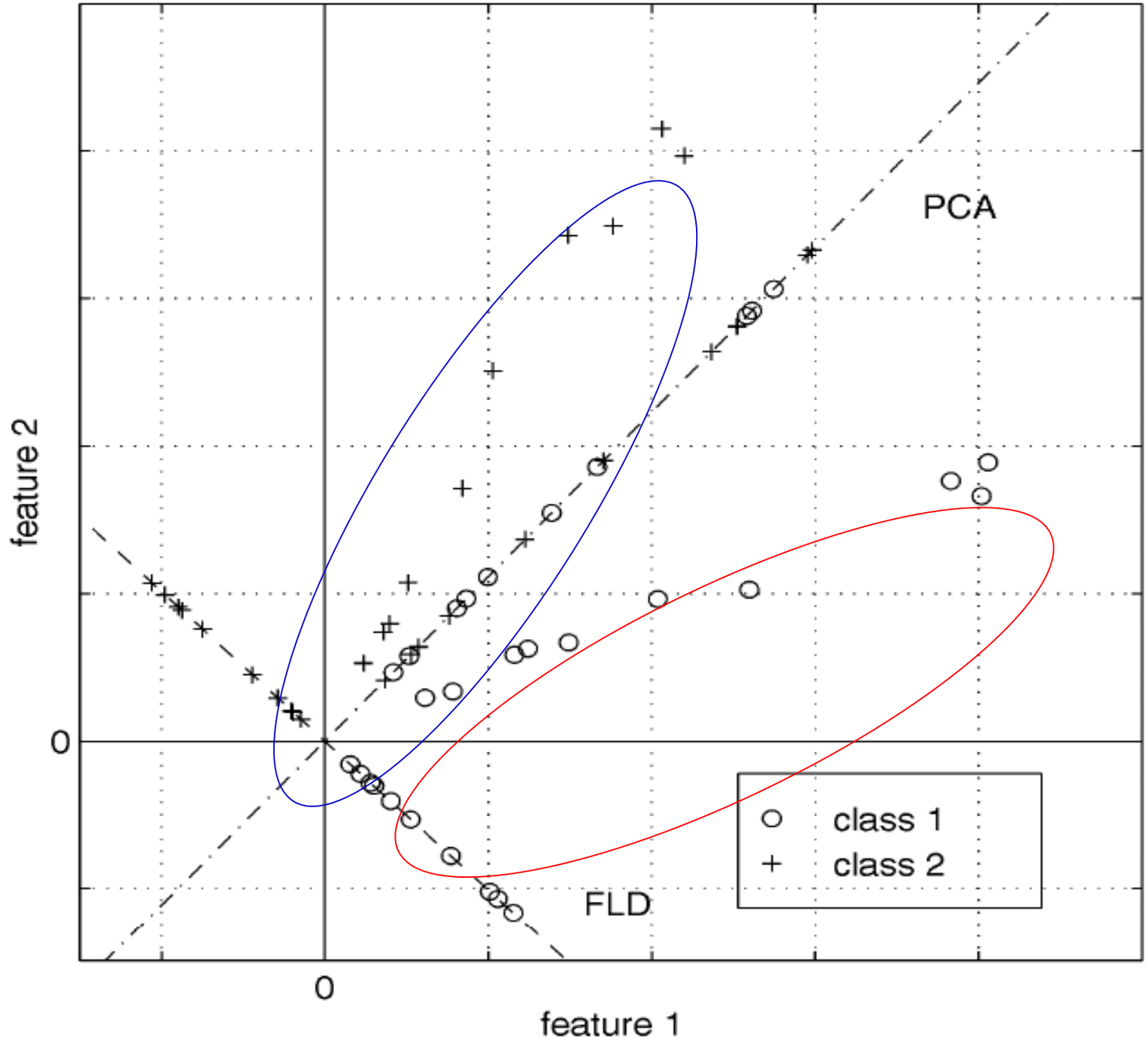
$$S_B w_i = \lambda_i S_W w_i \qquad i = 1, \ldots, m$$

- Rank of $\boldsymbol{W_{opt}}$ is limited
  - Rank(S_B) <= |C|-1
  - Rank(S_W) <= N-C

PCA vs. LDA

- Eigenfaces exploit the max scatter of the training images in face space
- Fisherfaces attempt to maximise the **between class scatter**, while minimising the **within class scatter**.

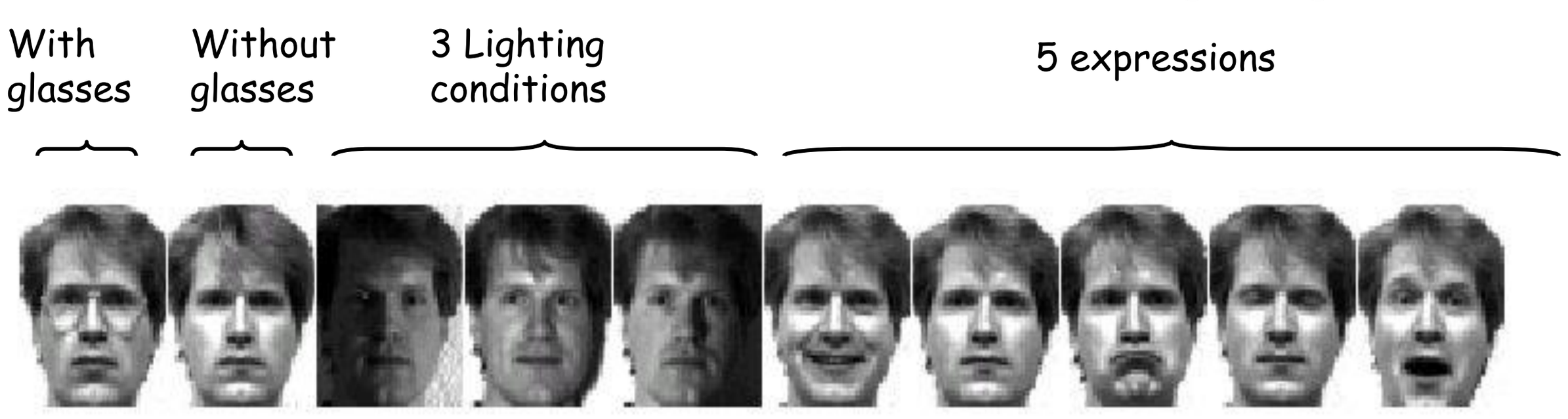
Basic intuition: PCA vs. LDA



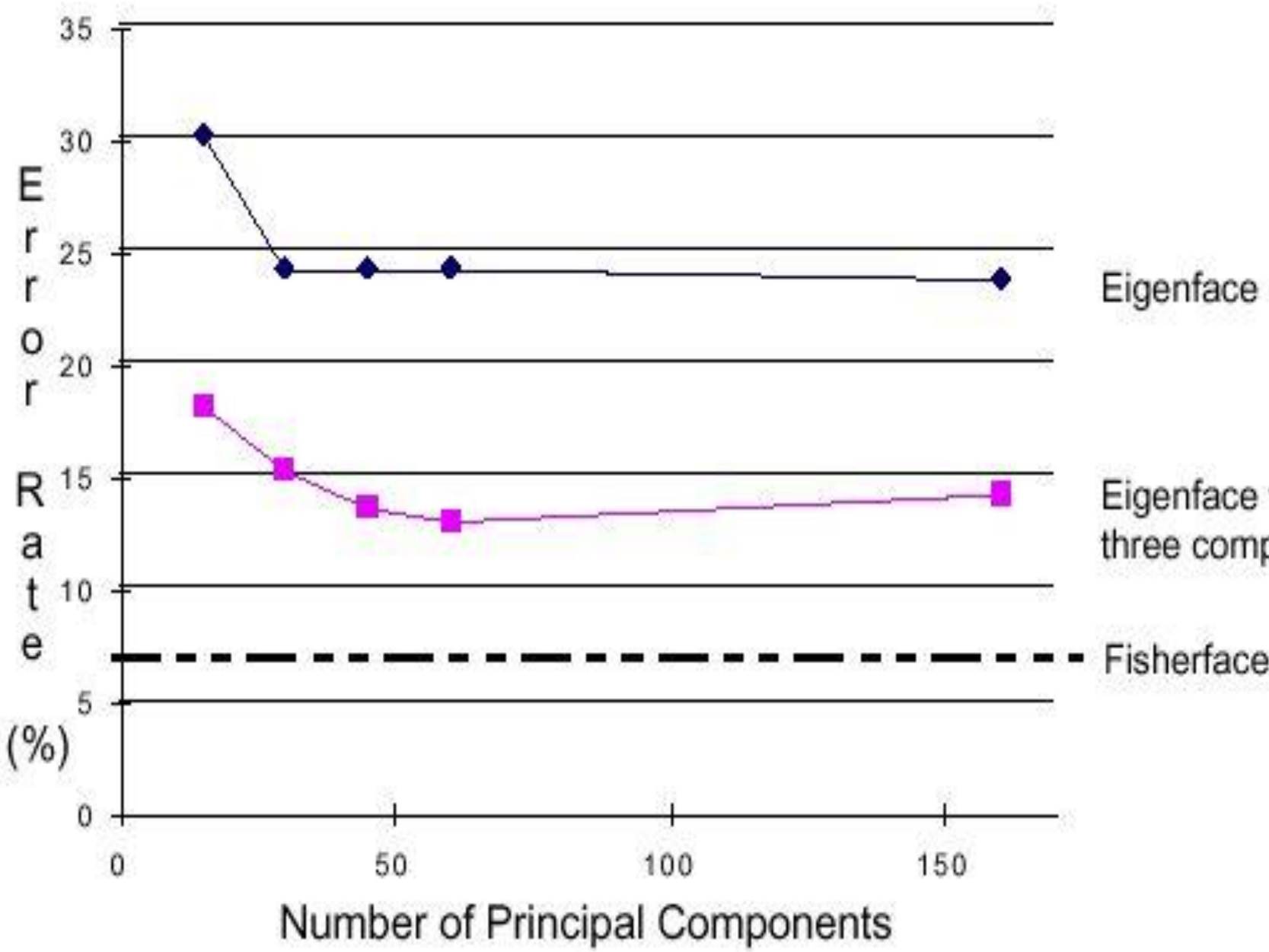
Results: Eigenface vs. Fisherface

- Input: 160 images of 16 people
- Train: 159 images
- Test: 1 image

- Variation in Facial Expression, Eyewear, and Lighting

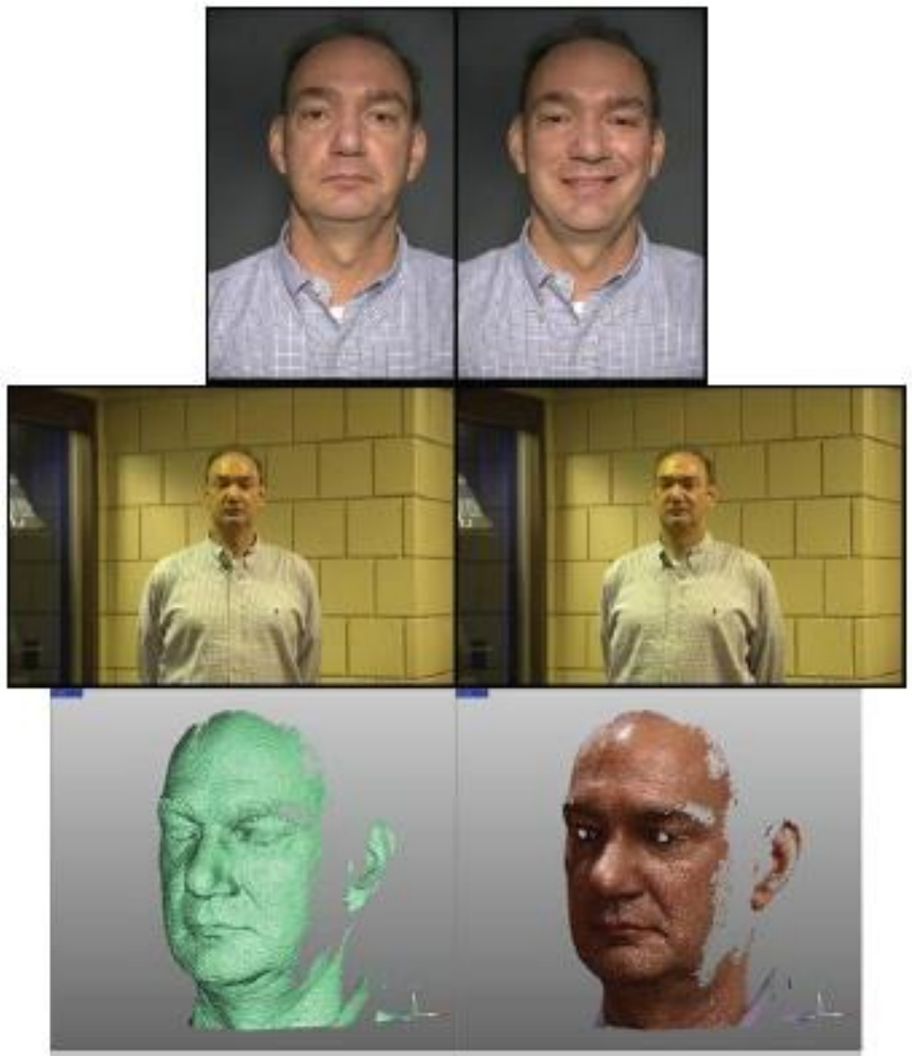


Eigenfaces vs. Fisherfaces

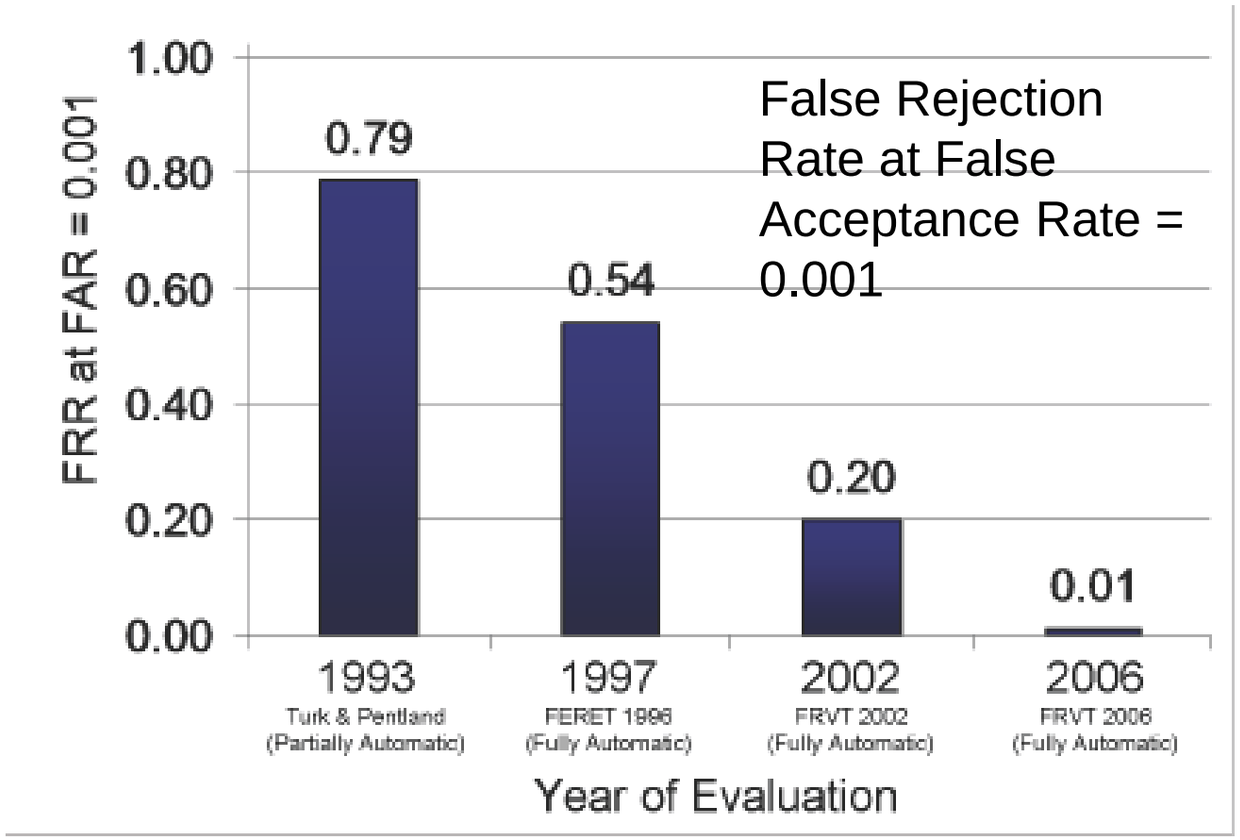


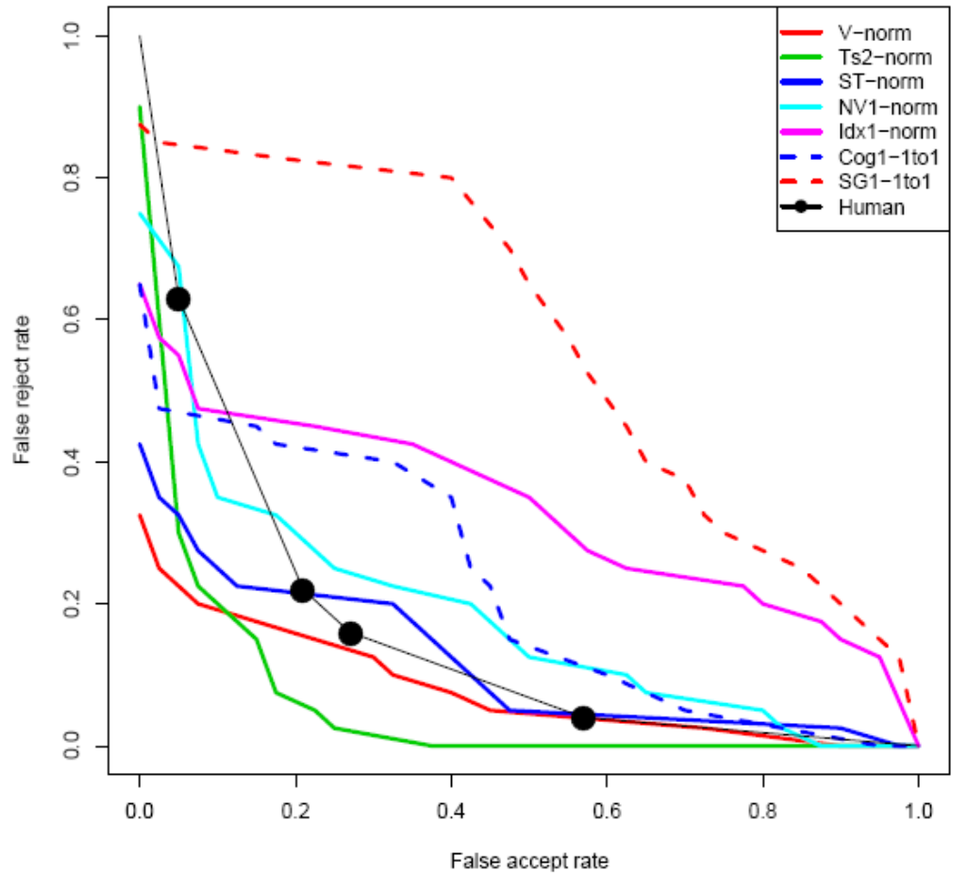
Large scale comparison of methods

- FRVT 2006 Report
- Not much (or any) information available about methods, but gives idea of what is doable



FVRT Challenge: interesting findings





- Left: Major progress since Eigenfaces
- Right: Computers outperformed humans in controlled settings (cropped frontal face, known lighting, aligned)
- Humans outperform greatly in less controlled settings (viewpoint variation, no crop, no alignment, change in age, etc.)

State-of-the-art Face Recognizers

- Most recent research focuses on “faces in the wild”, recognizing faces in normal photos
  - Classification: assign identity to face
  - Verification: say whether two people are the same

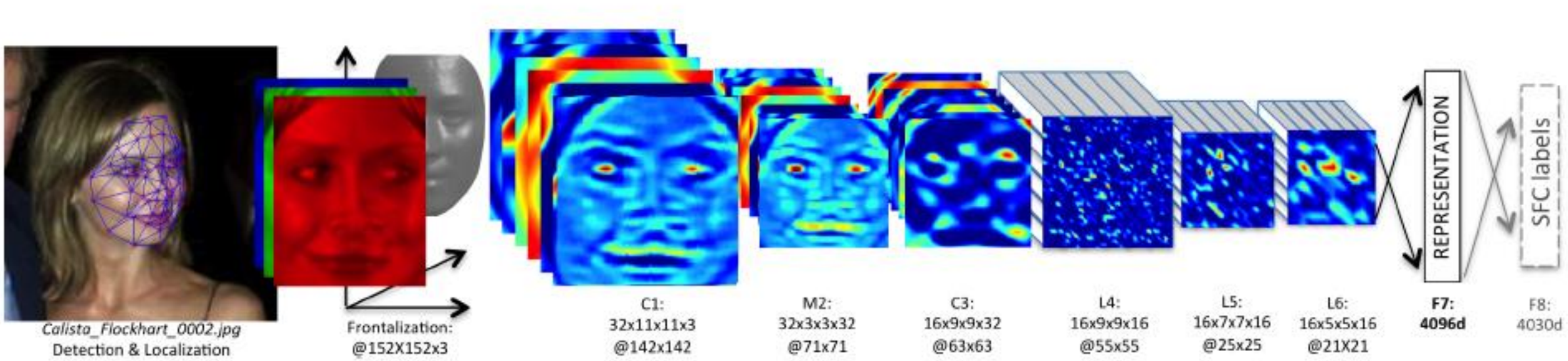
- Important steps
  1. Detect
  2. Align
  3. Represent
  4. Classify

DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Yaniv Taigman Ming Yang Marc'Aurelio Ranzato Lior Wolf

Facebook AI Research
Menlo Park, CA, USA
{yaniv, mingyang, ranzato}@fb.com

Tel Aviv University
Tel Aviv, Israel
wolf@cs.tau.ac.il



[DeepFace: Closing the Gap to Human-Level Performance in Face Verification]

Taigman, Yang, Ranzato, & Wolf (Facebook, Tel Aviv), CVPR 2014

Face Alignment

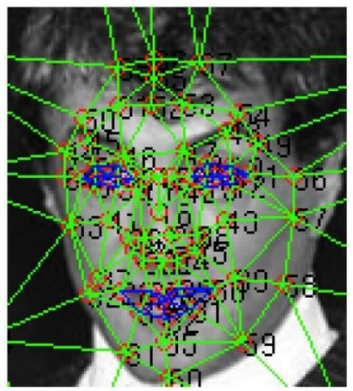
1. Detect a face and 6 fiducial markers using a support vector regressor (SVR)

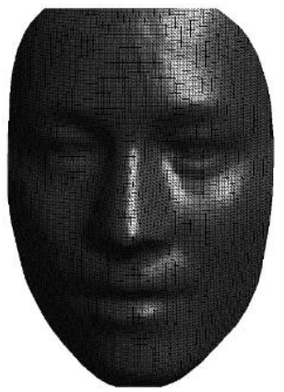
2. Iteratively scale, rotate, and translate image until it aligns with a target face

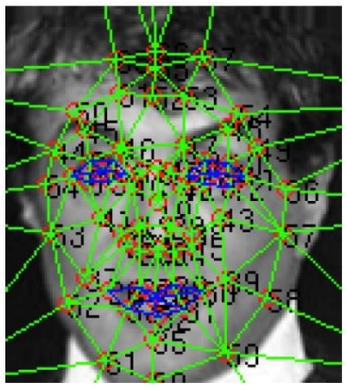
3. Localize 67 fiducial points in the 2D aligned crop

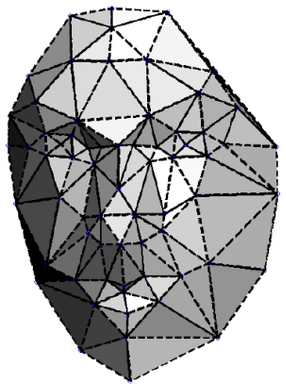
4. Create a generic 3D shape model by taking the average of 3D scans from the USF Human-ID database and manually annotate the 67 anchor points

5.Fit an affine 3D-to-2D camera and use it to direct the warping of the face

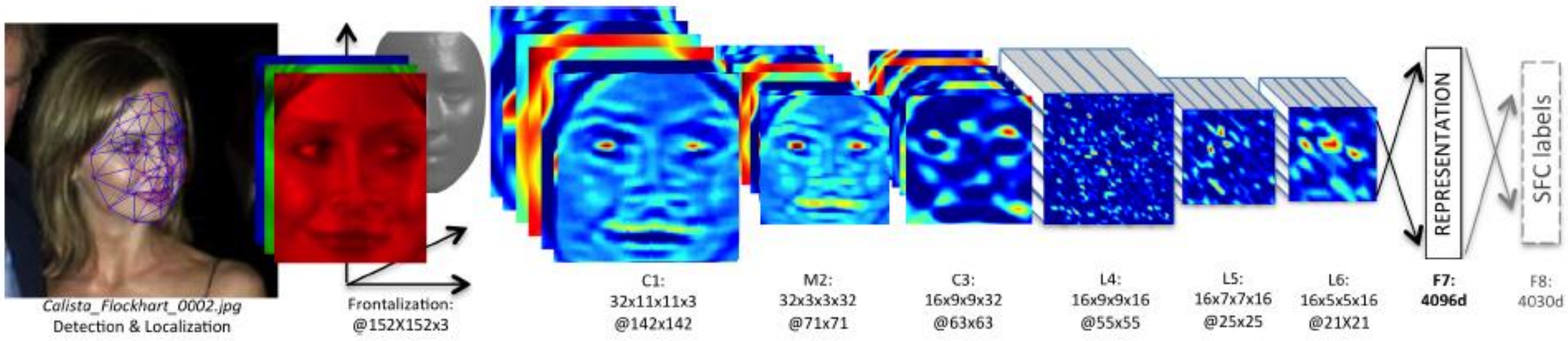








Train DNN classifier on aligned faces



Architecture (deep neural network classifier)

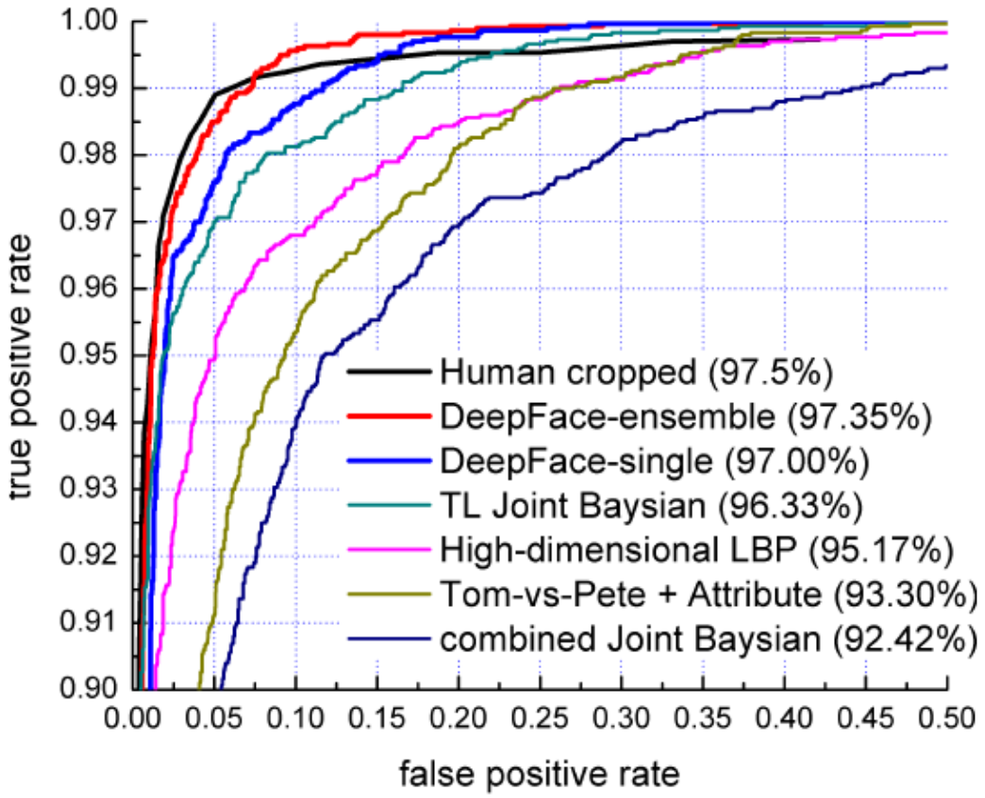
- Two convolutional layers (with one pooling layer)
- 3 locally connected and 2 fully connected layers
- > 120 million parameters

Train on dataset with 4400 individuals, ~1000 images each

- Train to identify face among set of possible people

Verification is done by comparing features at last layer for two faces

Results: Labeled Faces in the Wild Dataset



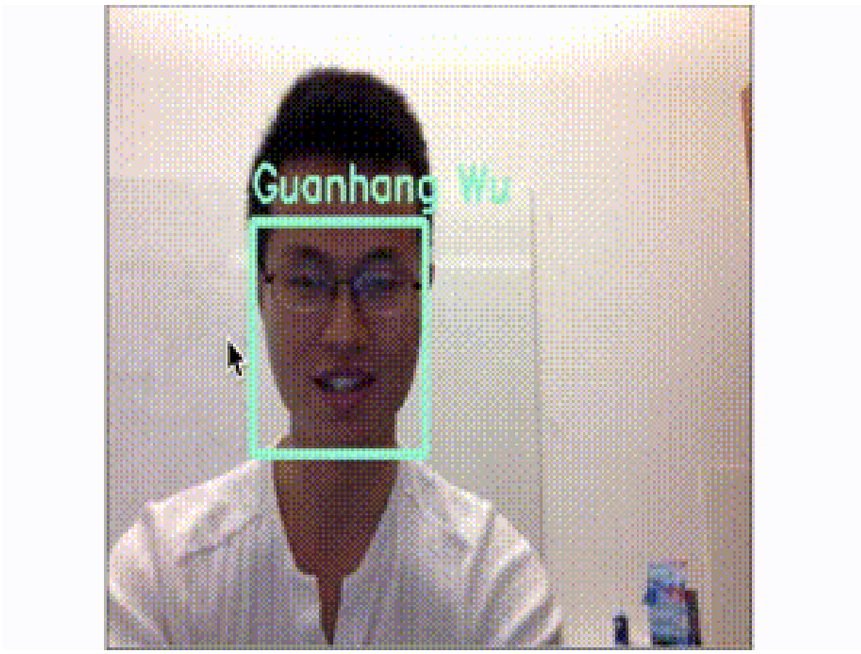
Method	Accuracy ± SE	Protocol
Joint Bayesian [6]	0.9242 ±0.0108	restricted
Tom-vs-Pete [4]	0.9330 ±0.0128	restricted
High-dim LBP [7]	0.9517 ±0.0113	restricted
TL Joint Bayesian [5]	0.9633 ±0.0108	restricted
DeepFace-single	**0.9592** ±0.0029	unsupervised
DeepFace-single	**0.9700** ±0.0028	restricted
DeepFace-ensemble	**0.9715** ±0.0027	restricted
DeepFace-ensemble	**0.9735** ±0.0025	unrestricted
Human, cropped	0.9753	

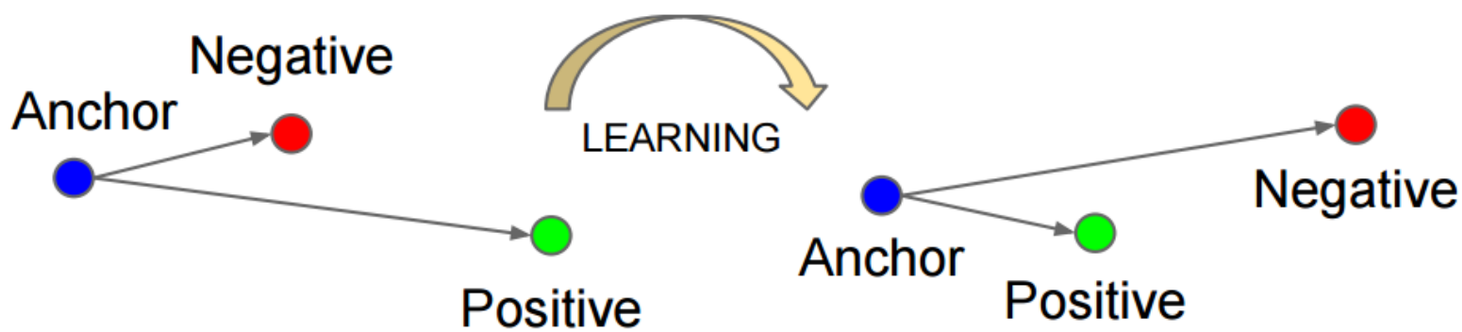
Performs similarly to humans!

(note: humans would do better with uncropped faces)

Experiments show that alignment is crucial (0.97 vs 0.88) and that deep features help (0.97 vs. 0.91)

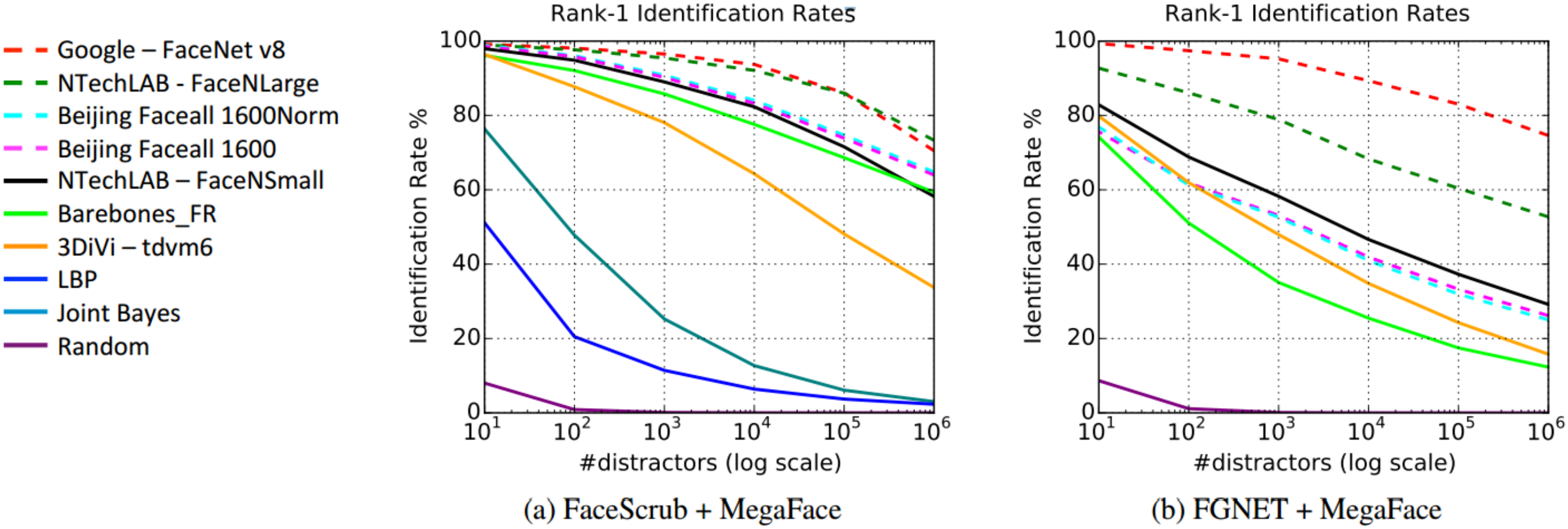
OpenFace (FaceNet)





FaceNet: A Unified Embedding for Face Recognition and Clustering, CVPR 2015 http://cmusatyalab.github.io/openface/

MegaFace Benchmark

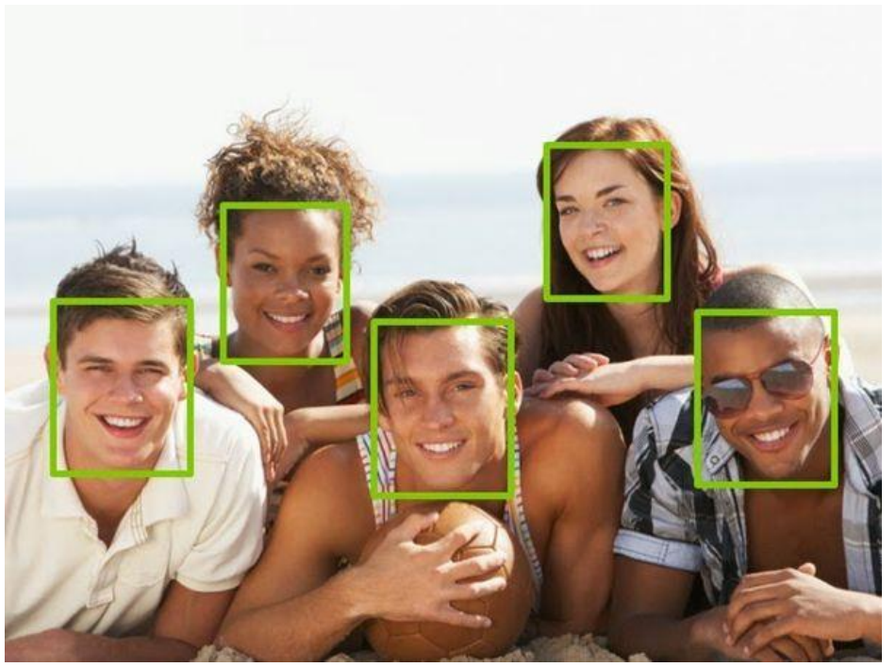


(a) FaceScrub + MegaFace

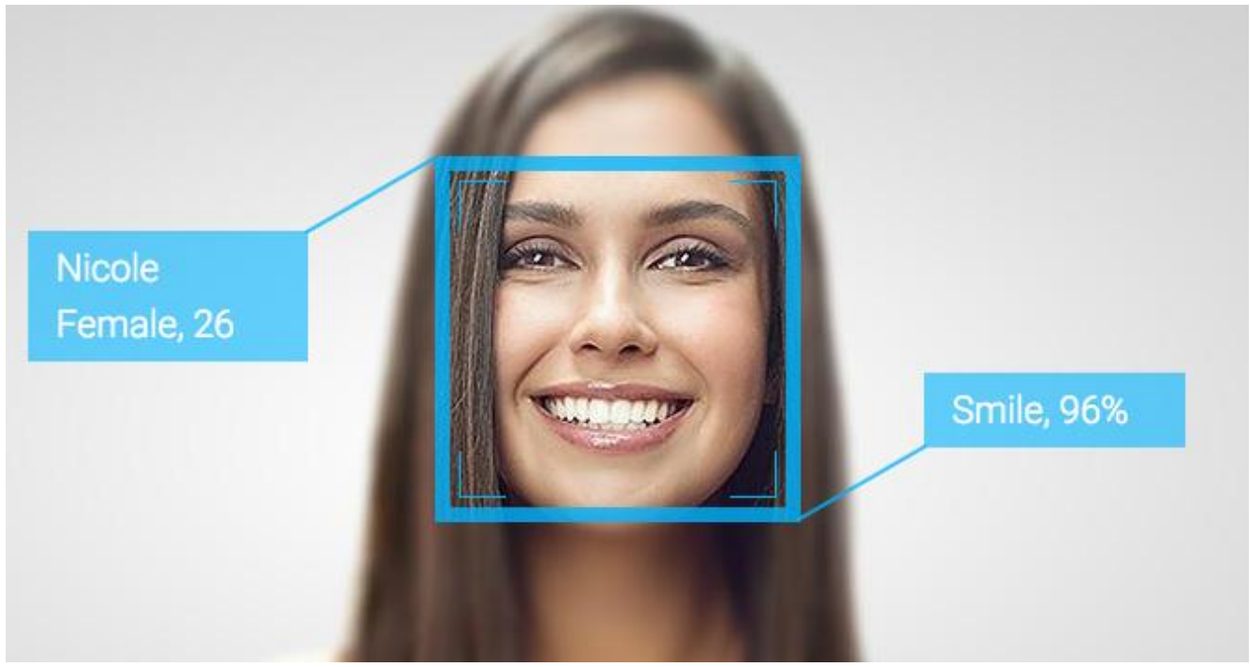
(b) FGNET + MegaFace

The MegaFace Benchmark: 1 Million Faces for Recognition at Scale, CVPR 2016

Detection versus Recognition

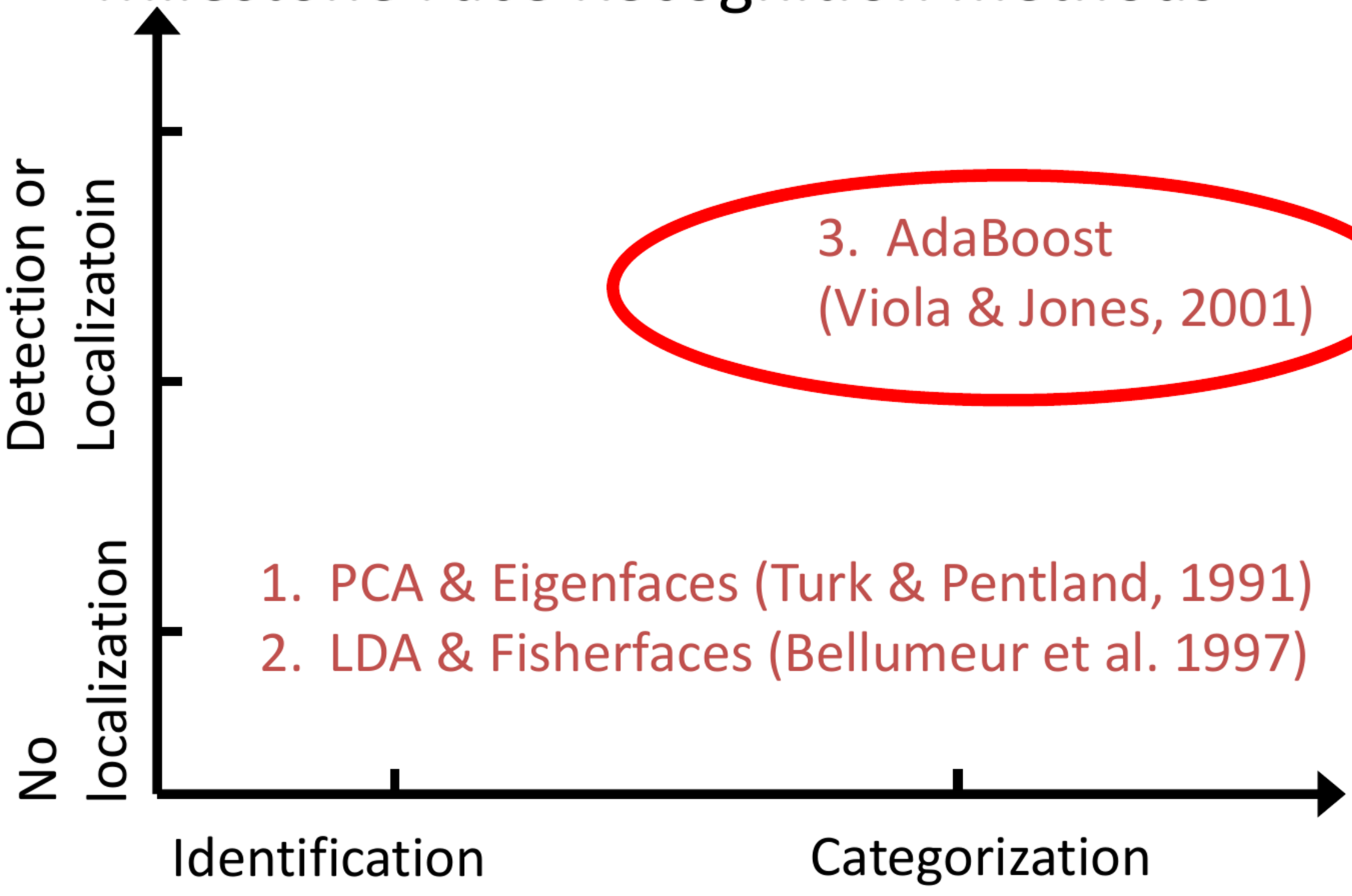


Detection finds the faces in images



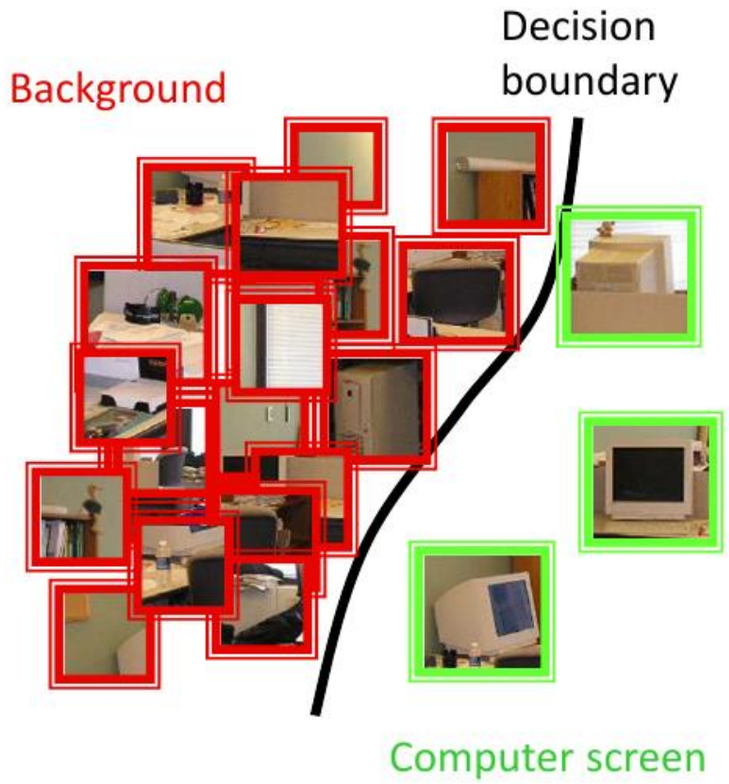
Recognition recognizes WHO the person is

Milestone Face Recognition methods



Detecting foreground objects: A binary classification formulation

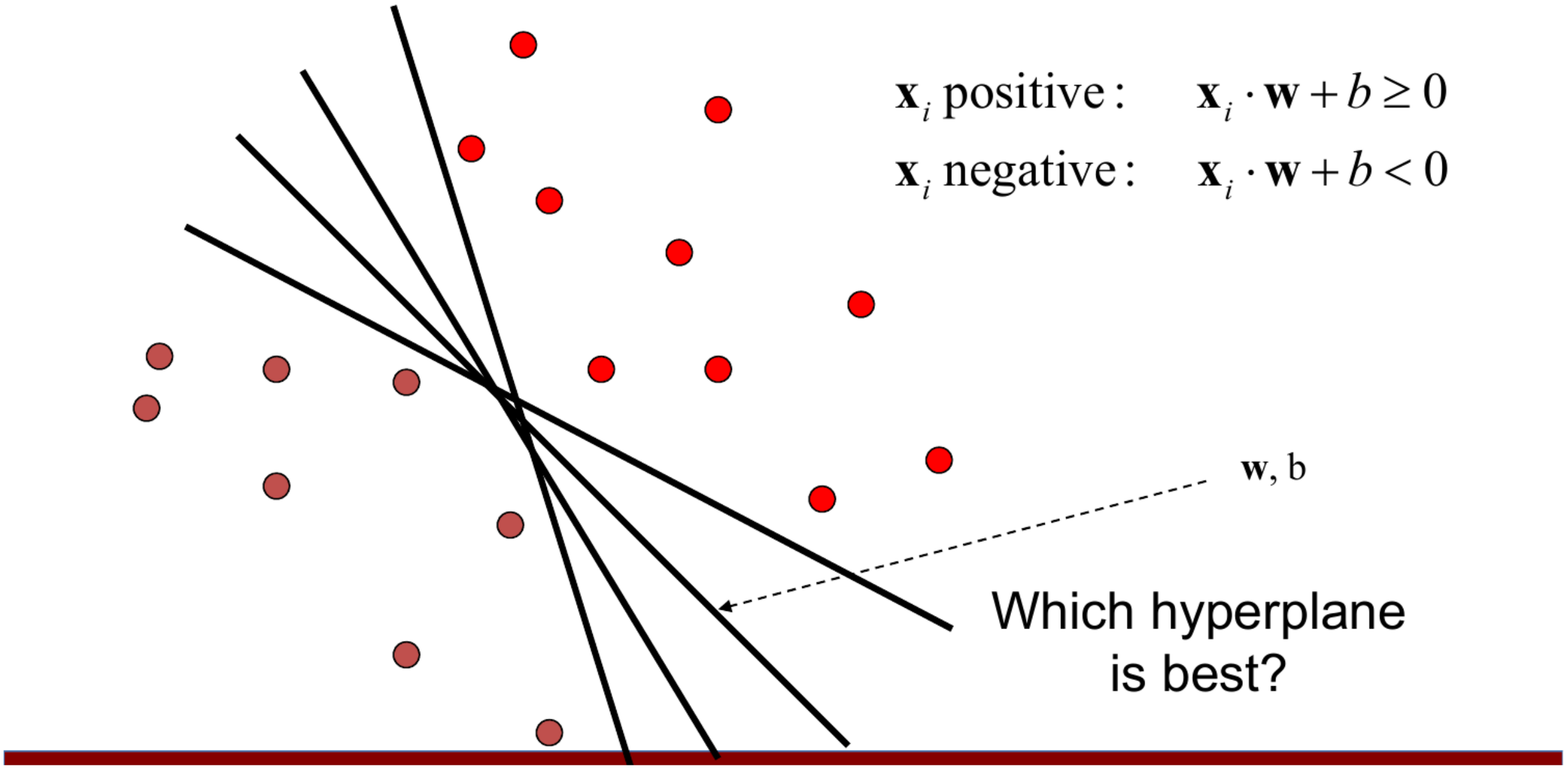
Features



In some feature space

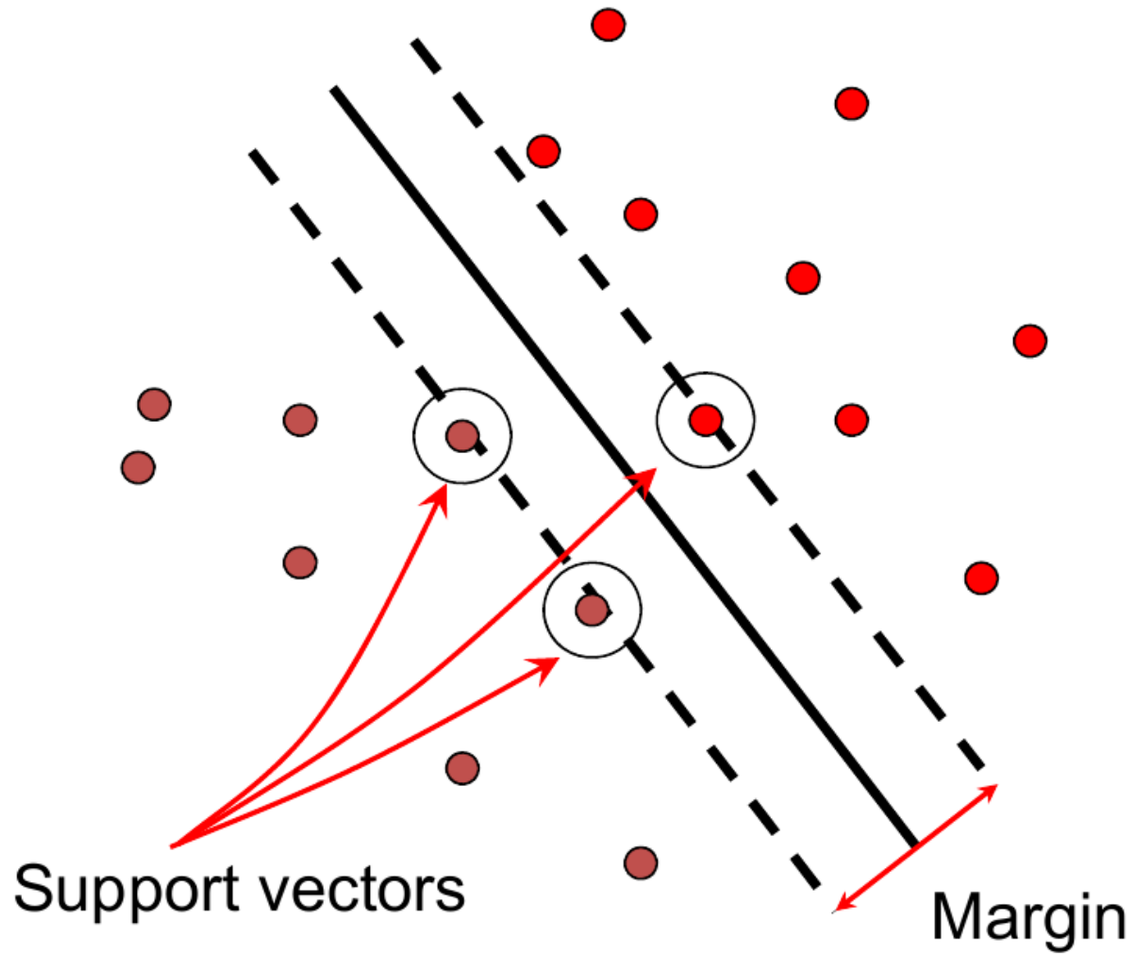
Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



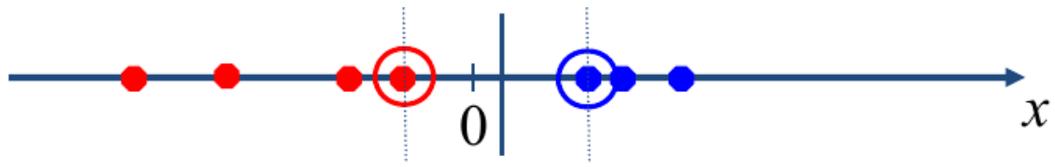
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



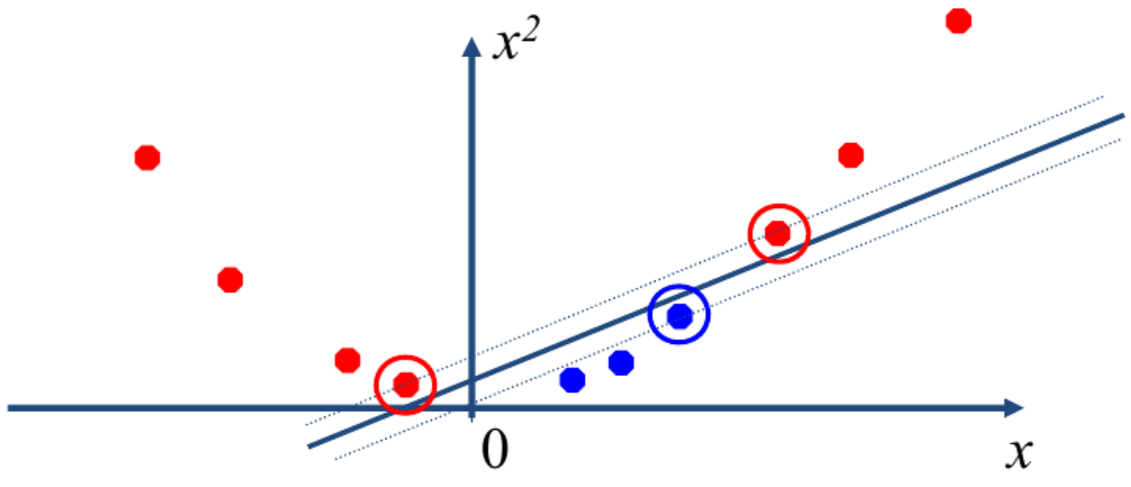
Nonlinear SVMs

- Datasets that are linearly separable work out great:



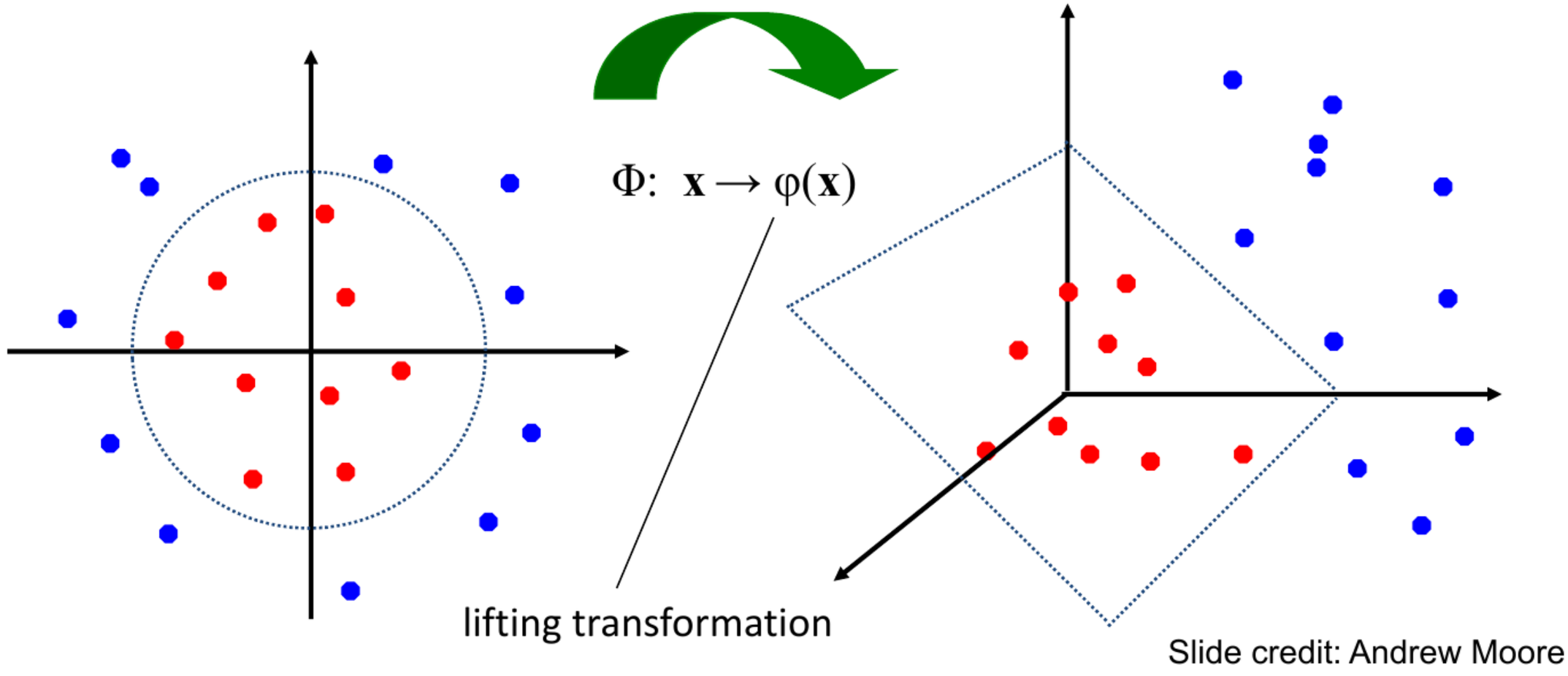
- But what if the dataset is just too hard?

- We can map it to a higher-dimensional space:



Nonlinear SVMs

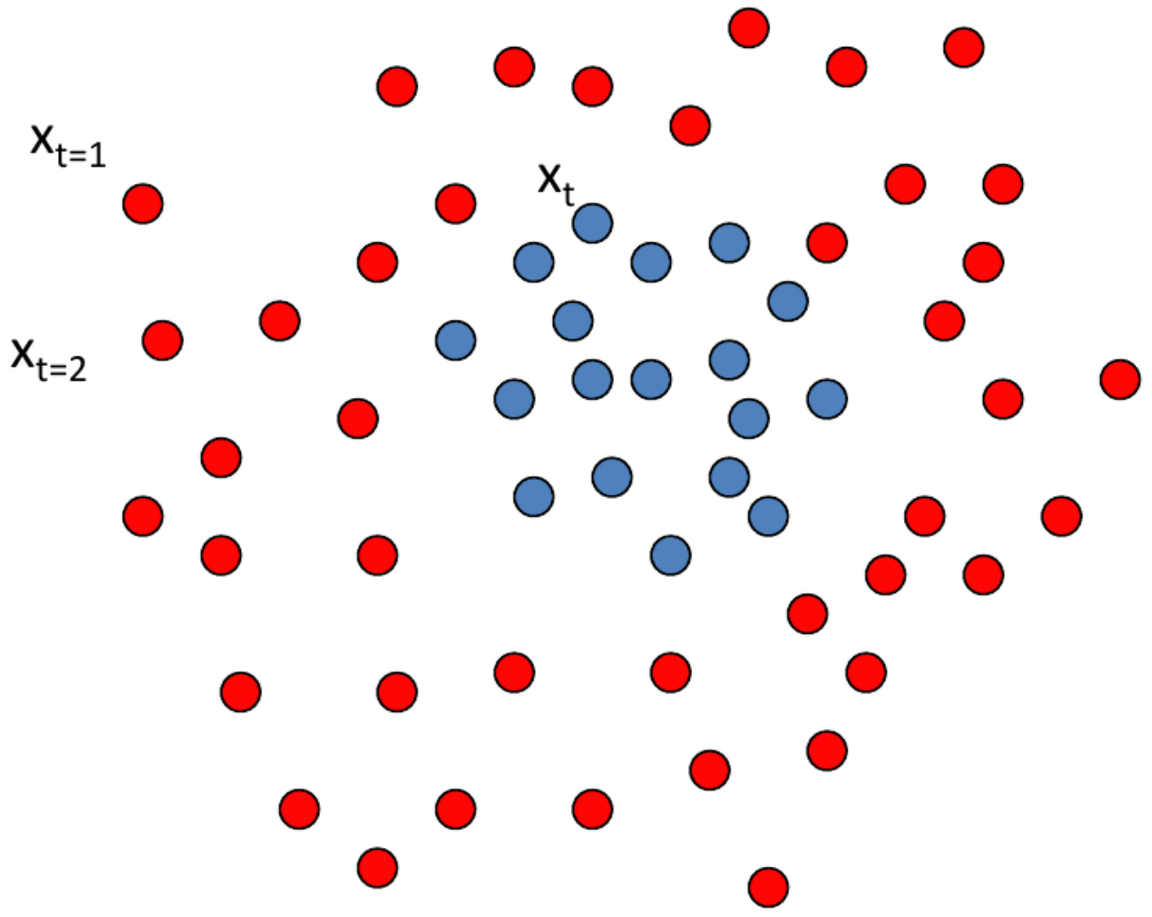
- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Boosting

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.



Each data point has a class label:

$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

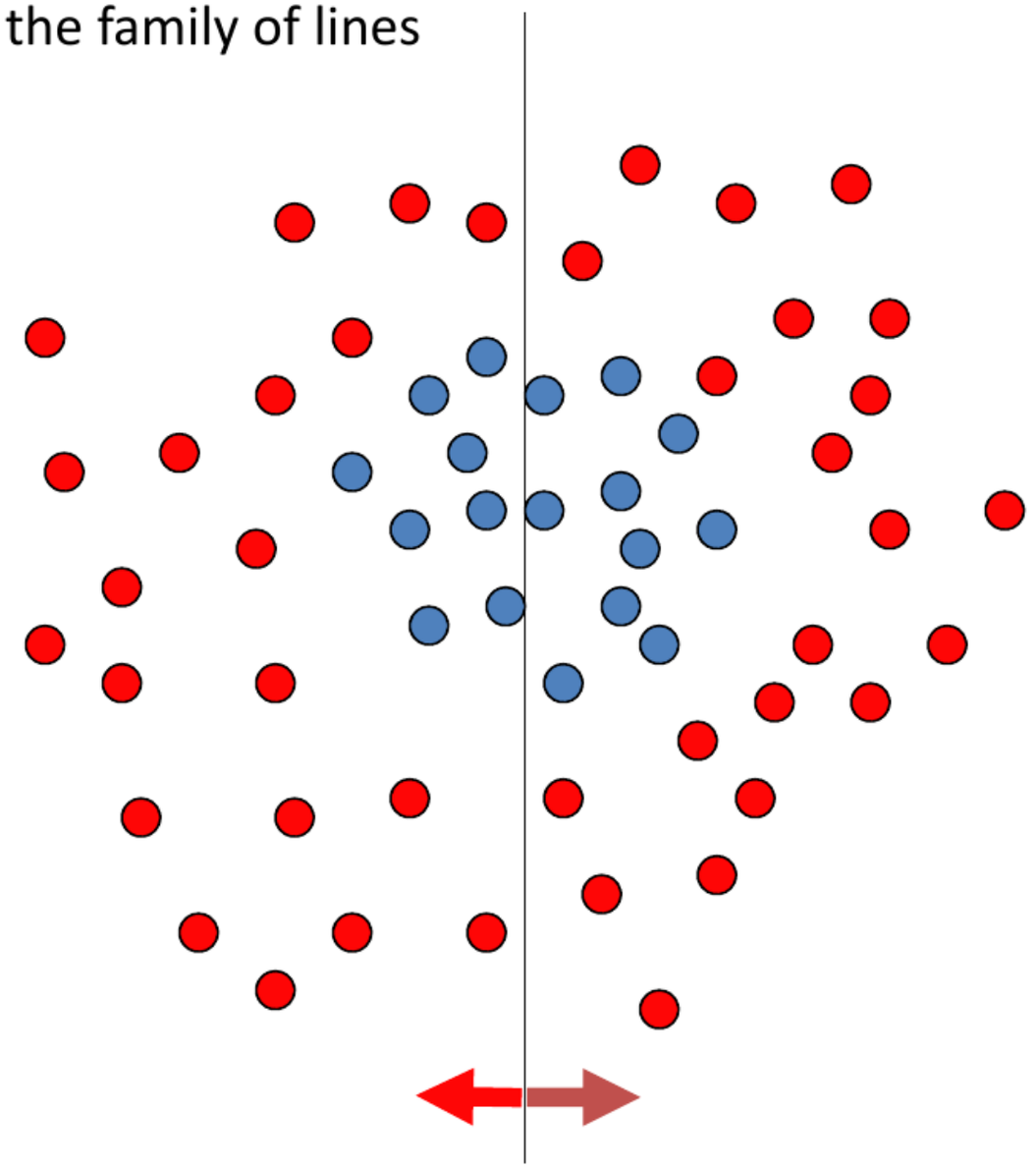
and a weight:

$w_t = 1$

- It is a sequential procedure:

Toy example

Weak learners from the family of lines



Each data point has a class label:

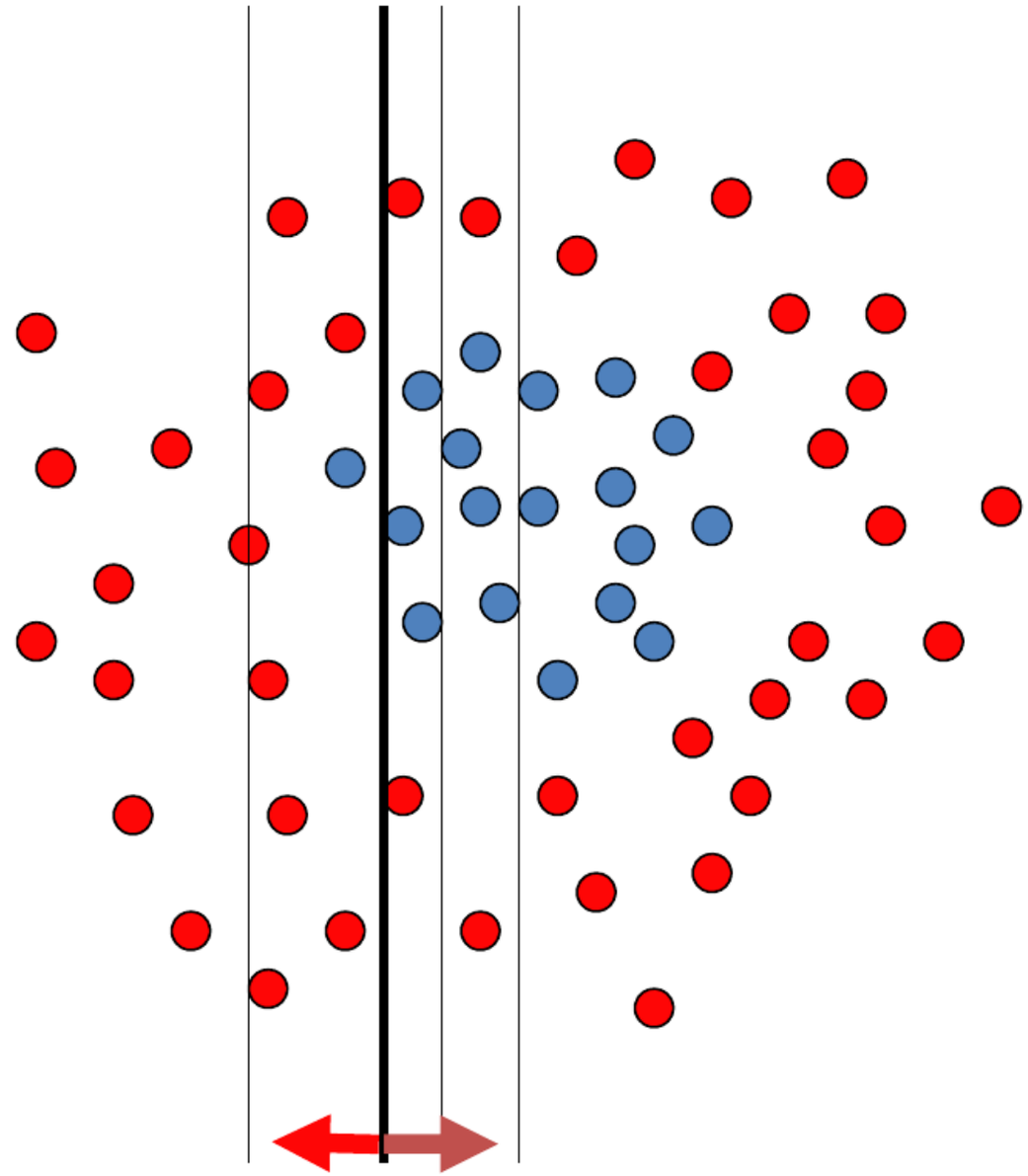
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

and a weight:

$w_t = 1$

h => p(error) = 0.5 it is at chance

Toy example



Each data point has a class label:

$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

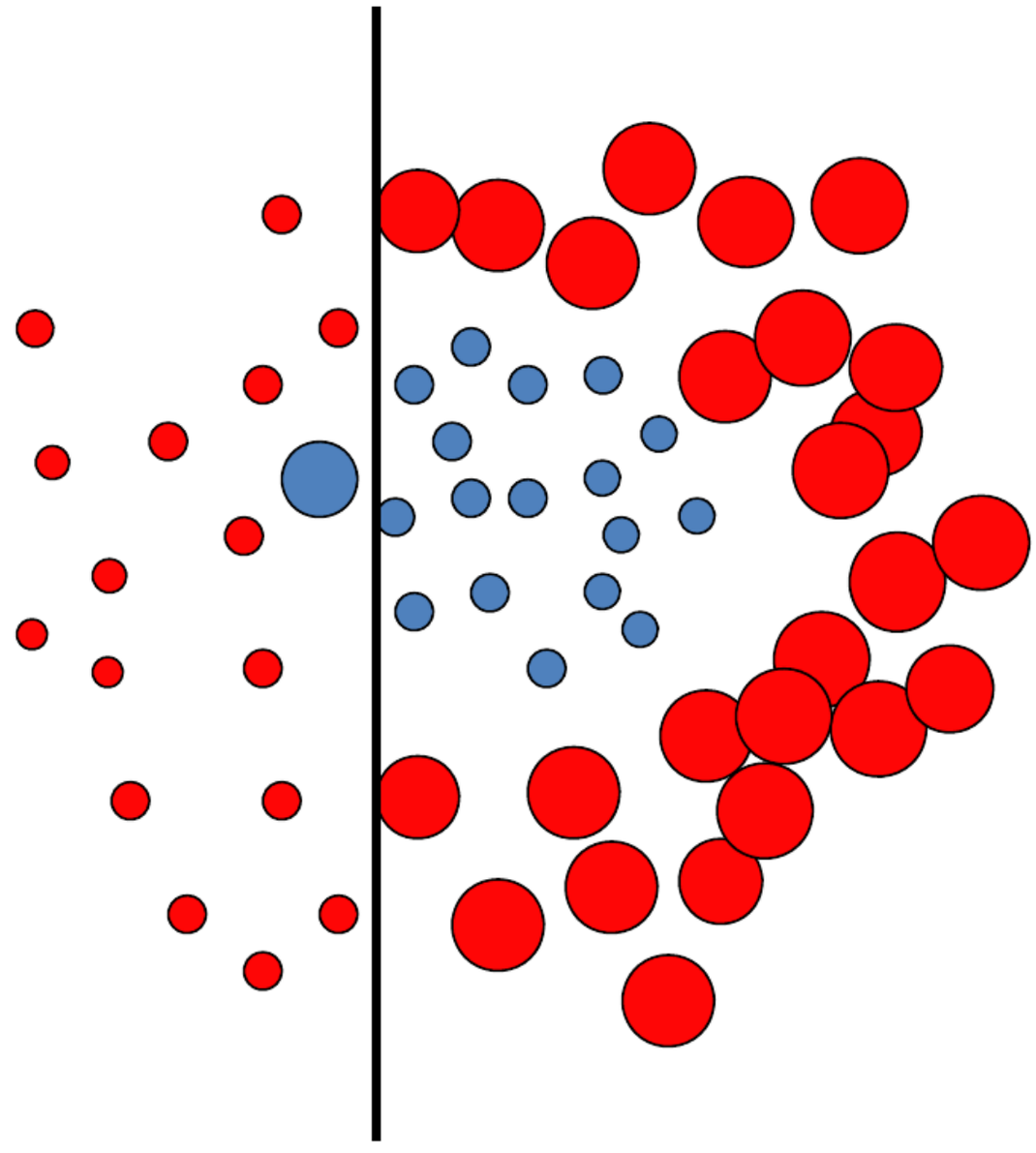
and a weight:

$w_t = 1$

This one seems to be the best

This is a '**weak classifier**': It performs slightly better than chance.

Toy example



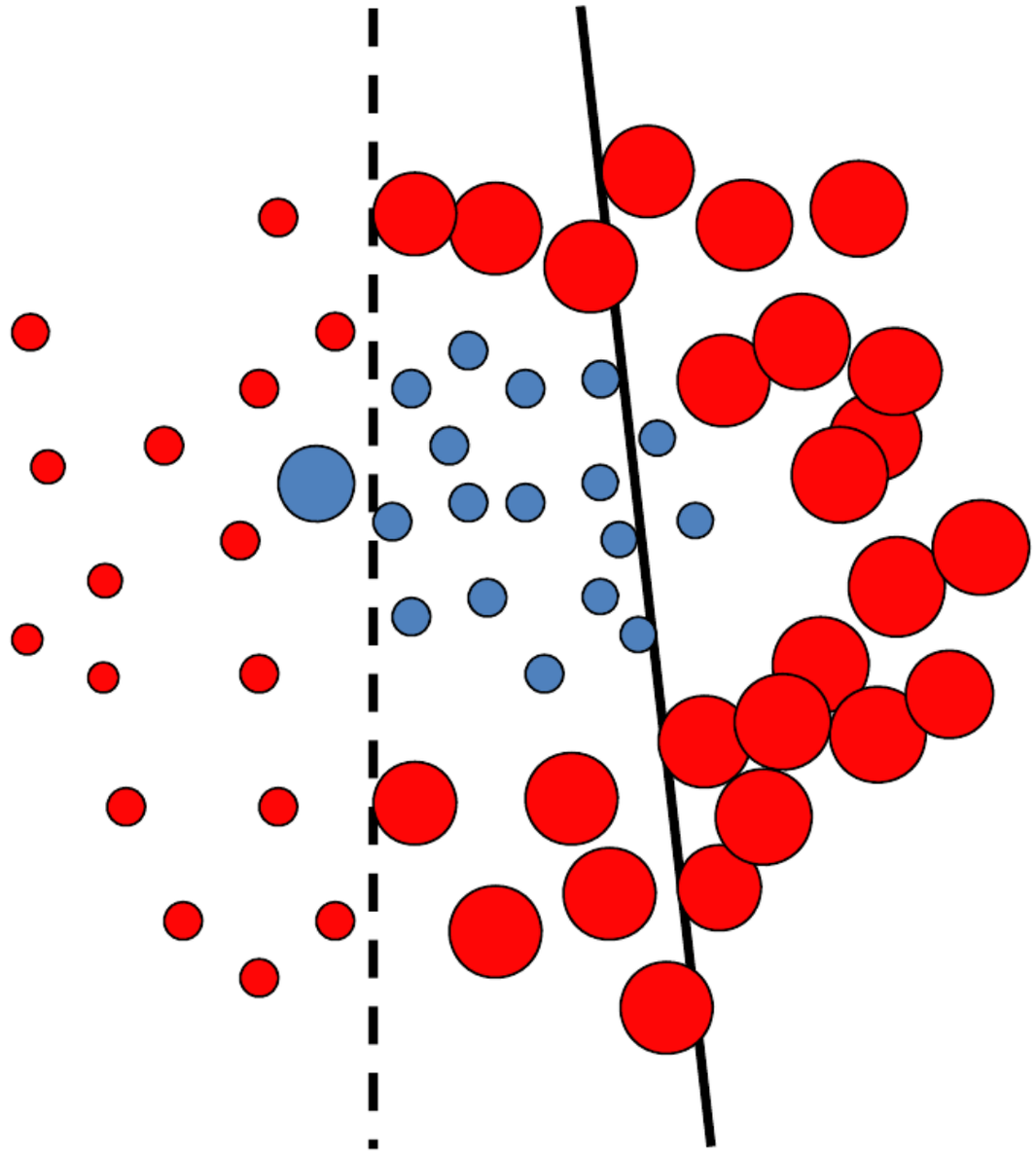
Each data point has a class label:

$$y_t = \begin{cases} +1 & (\\ -1 & (\end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Toy example



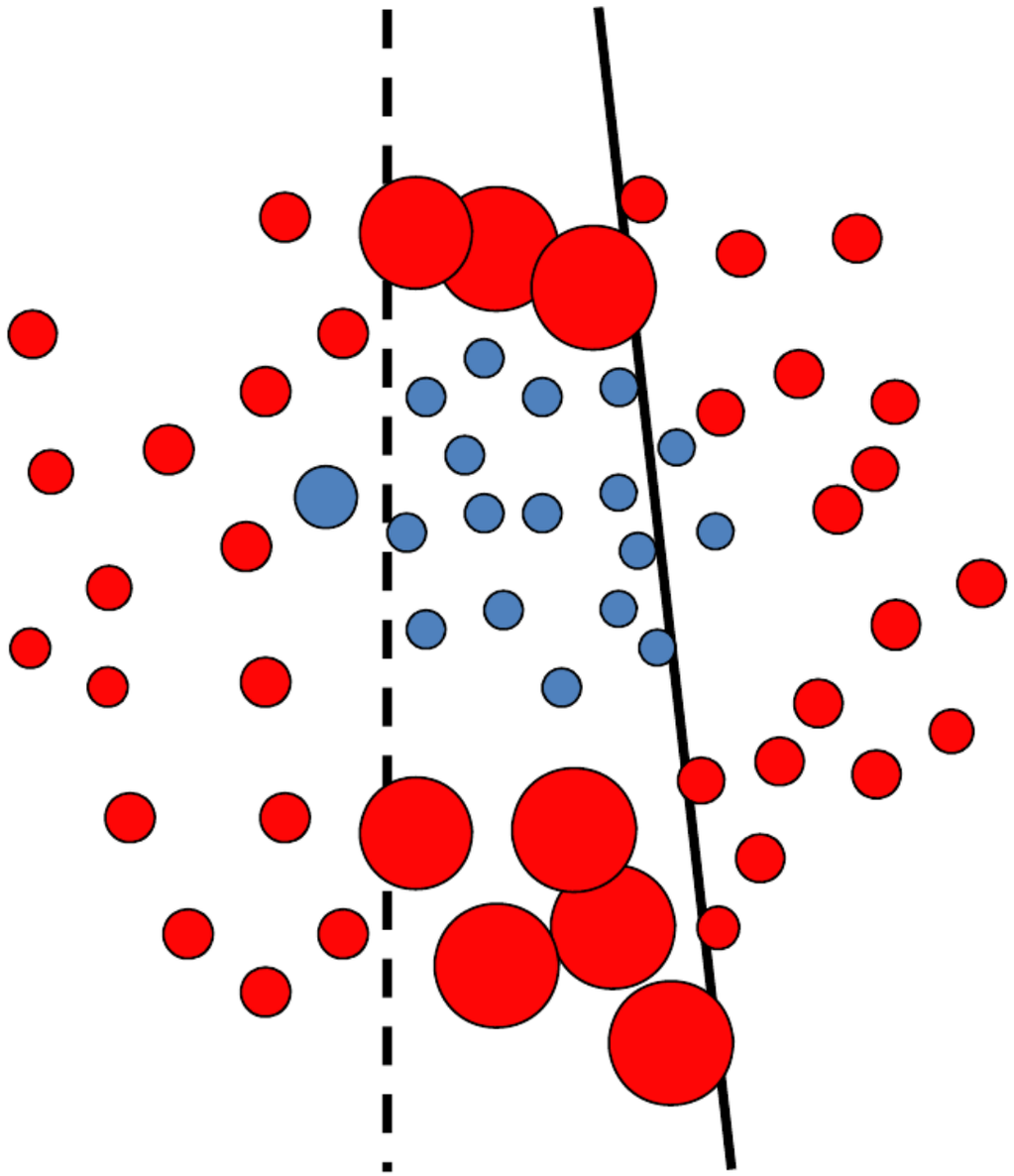
Each data point has a class label:

$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Toy example



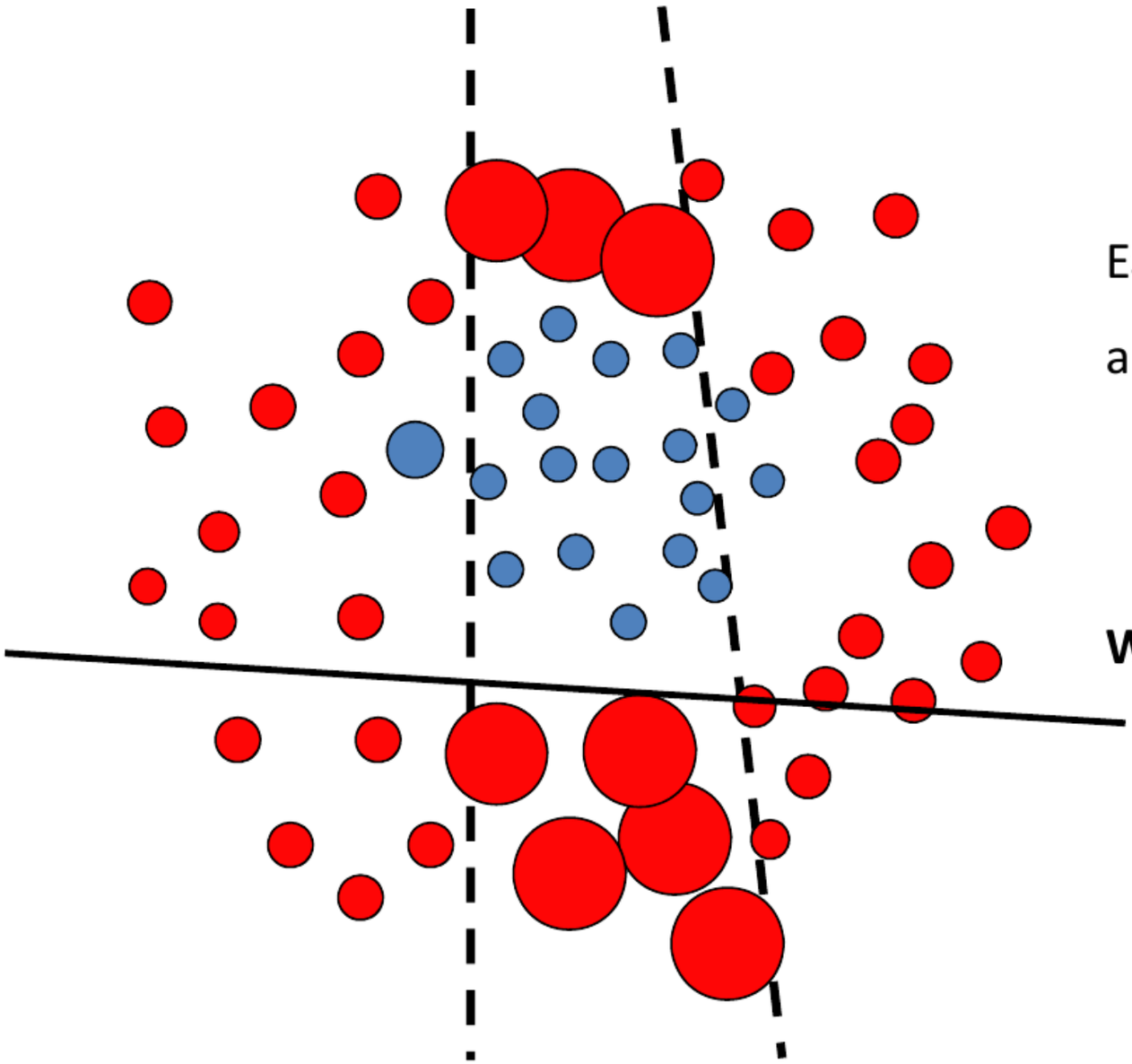
Each data point has a class label:

$$y_t = \begin{cases} +1 \text{ (} \bullet \text{)} \\ -1 \text{ (} \bullet \text{)} \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Toy example



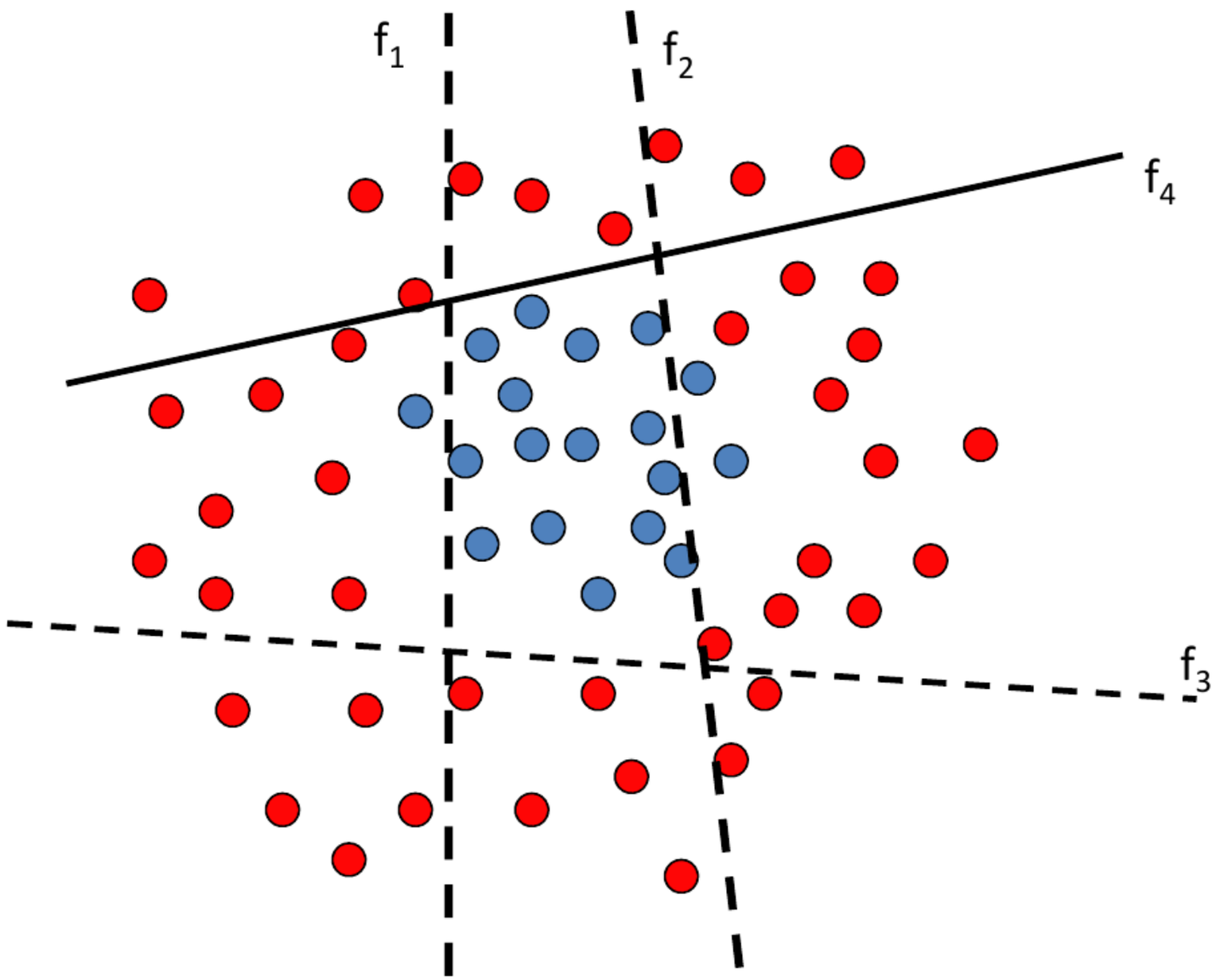
Each data point has a class label:

$$y_t = \begin{cases} +1 \text{ (} \bullet \text{)} \\ -1 \text{ (} \bullet \text{)} \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Toy example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Boosting

- Defines a classifier using an additive model:

$$h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) + \cdots$$

Strong classifier

Features vector

Weight

Weak classifier

Boosting

- Defines a classifier using an additive model:

$$h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) + \cdots$$

Strong classifier

Features vector

Weight

Weak classifier

- We need to define a family of weak classifiers

$h_k(x)$ form a family of weak classifiers

Why boosting?

- A simple algorithm for learning robust classifiers
  - Freund & Shapire, 1995
  - Friedman, Hastie, Tibshhirani, 1998

- Provides efficient algorithm for sparse visual feature selection
  - *Tieu & Viola, 2000*
  - *Viola & Jones, 2003*

- Easy to implement, not requires external optimization tools.

Boosting - mathematics

- Weak learners

value of rectangle feature

$$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

threshold

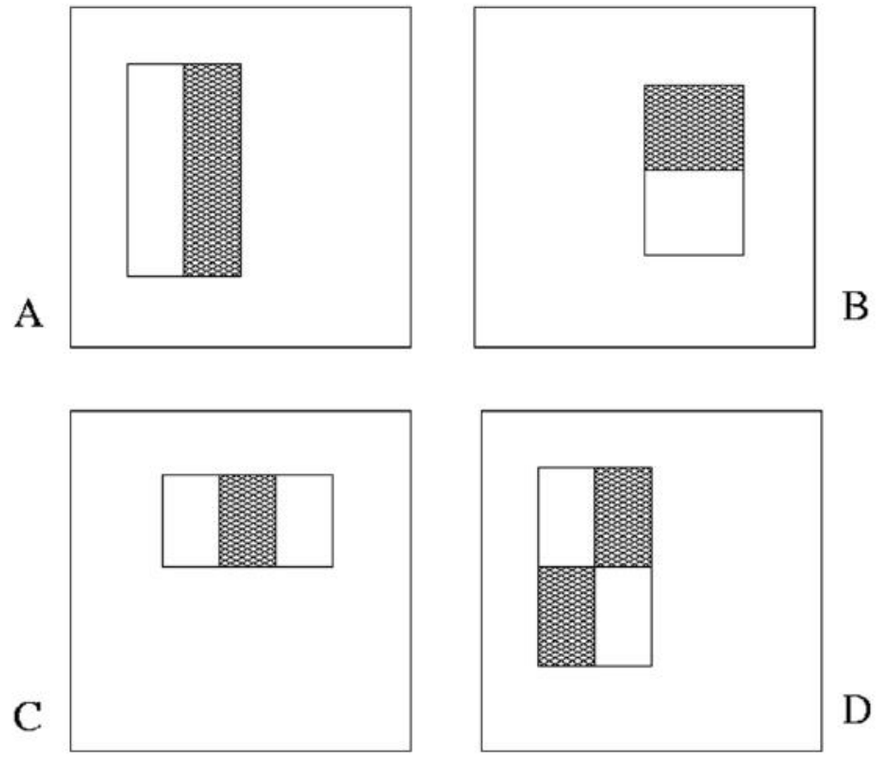
- Final strong classifier

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Weak classifier

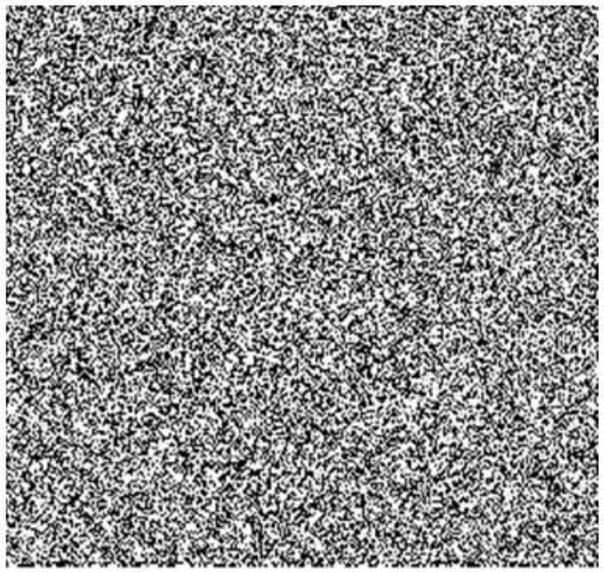
- 4 kind of Rectangle filters
- *Value =*

∑ (pixels in white area) –
∑ (pixels in black area)



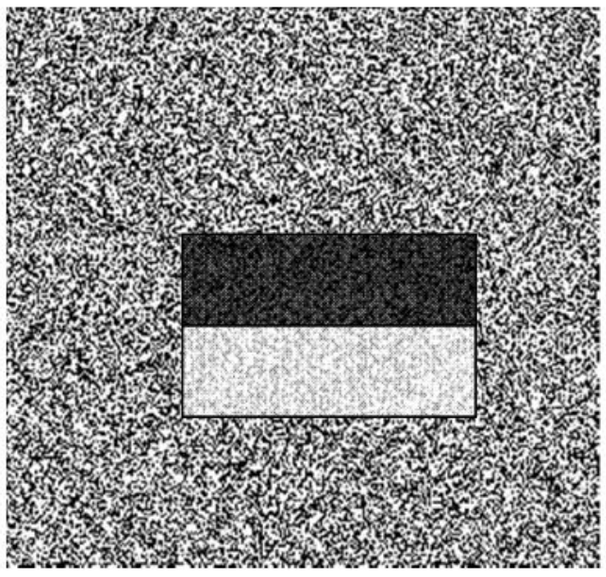
Credit slide: S. Lazebnik

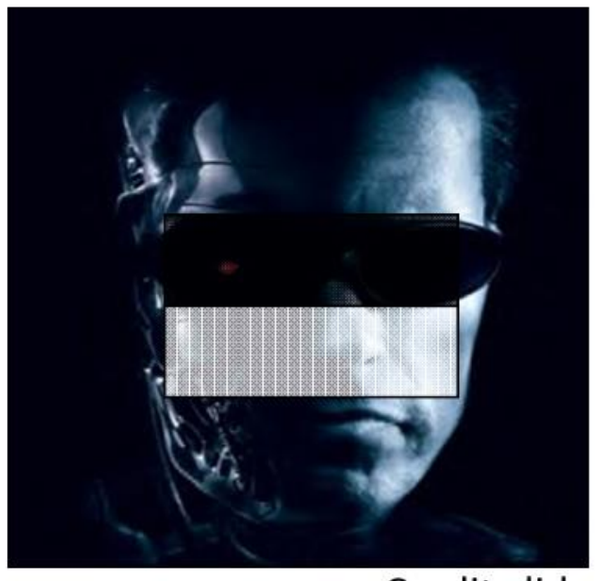
Weak classifier



Source

Result





Credit slide: S. Lazebnik

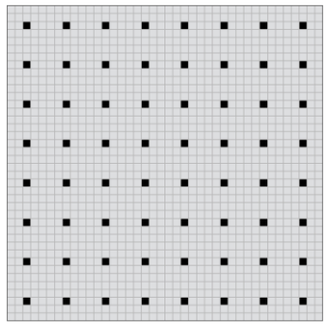
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder. International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

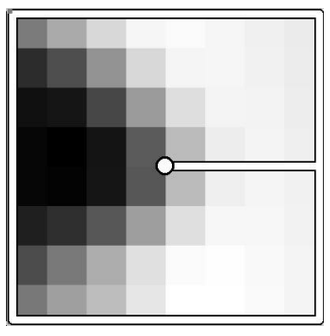
Get 40 x 40 image patch, subsample every 5th pixel

(low frequency filtering, absorbs localization errors)



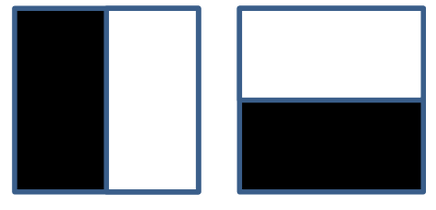
Subtract the mean, divide by standard deviation

(removes bias and gain)



Haar Wavelet Transform

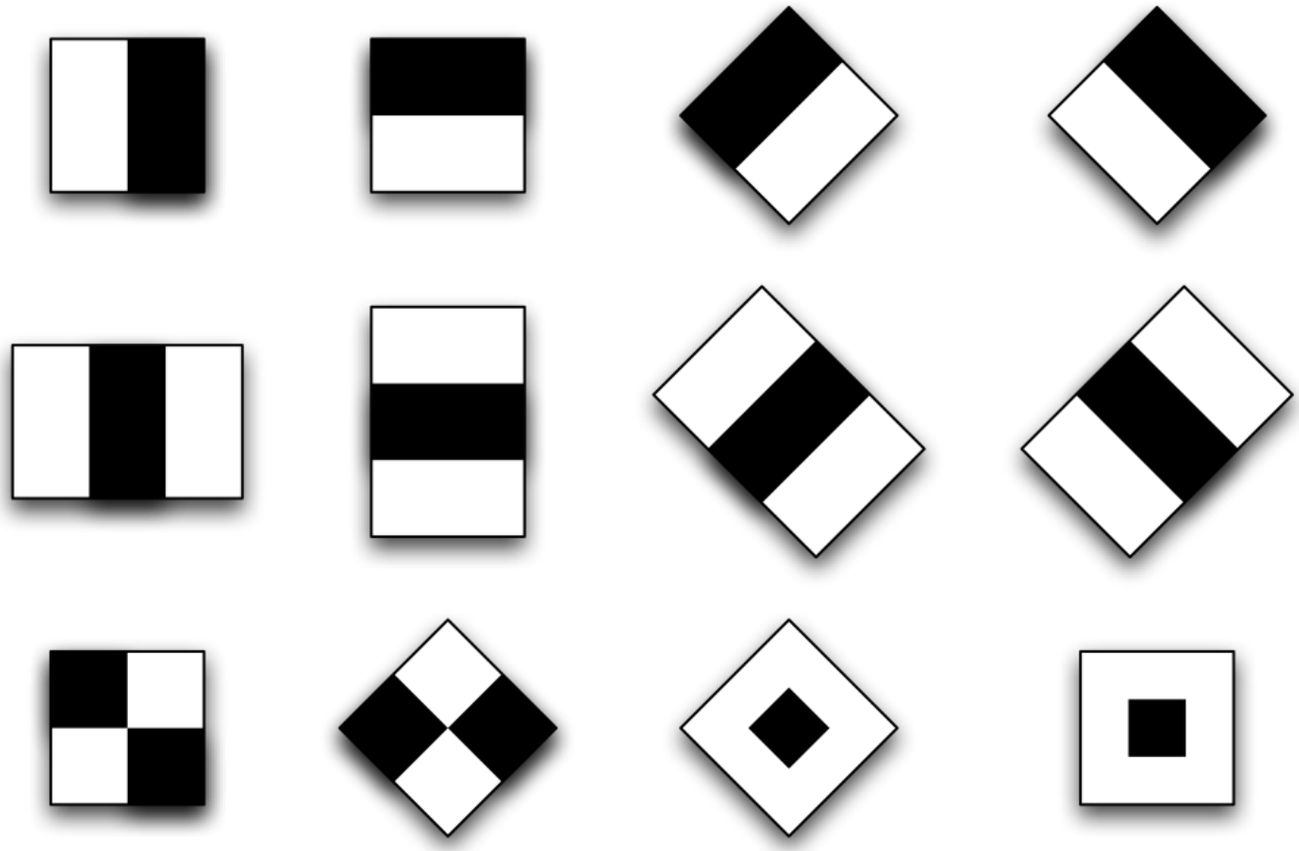
(low frequency projection)



Haar Wavelets

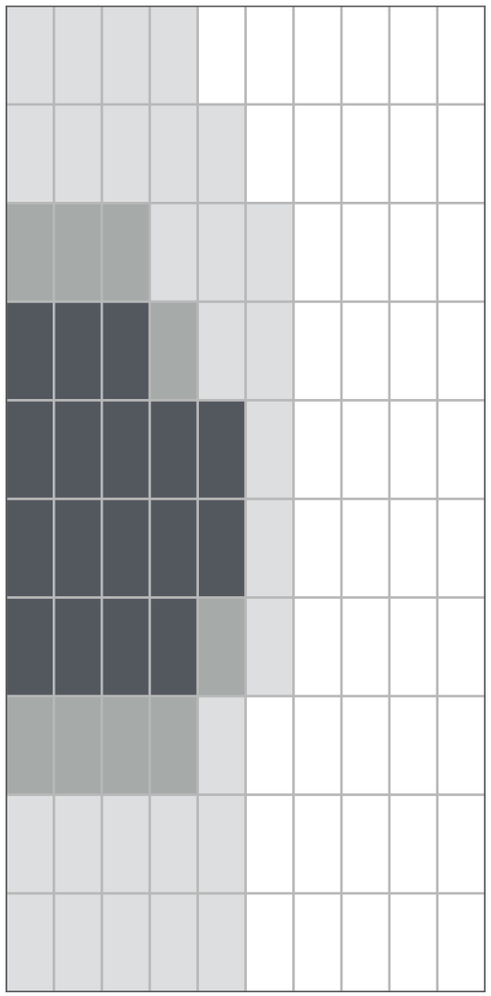
(actually, Haar-like features)

Use responses of a bank of filters as a descriptor

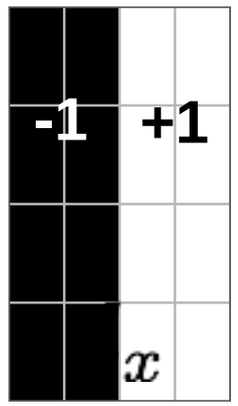


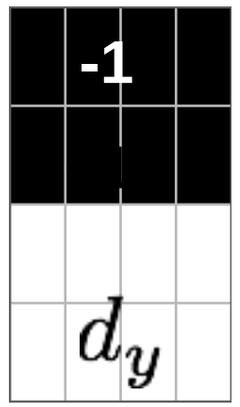
Haar wavelet responses can be computed with filtering

image patch



Haar wavelets filters





Haar wavelet responses can be computed **efficiently** (in constant time) with integral images

Integral Image

original image

$I(x, y)$

1	5	2
2	4	1
2	1	1

$A(x, y)$

1	6	8
3	12	15
5	15	19

integral image

$$A(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

Integral Image

original image

$I(x, y)$

1	5	2
2	4	1
2	1	1

$A(x, y)$

1	6	8
3	12	15
5	15	19

integral image

$$A(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

Can find the **sum** of any block using **3** operations

$$A(x_1, y_1, x_2, y_2) = A(x_2, y_2) - A(x_1, y_2) - A(x_2, y_1) + A(x_1, y_1)$$

What is the sum of the bottom right 2x2 square?

$$A(x_1, y_1, x_2, y_2) = A(x_2, y_2) - A(x_1, y_2) - A(x_2, y_1) + A(x_1, y_1)$$

$I(x, y)$

1	5	2
2	4	1
2	1	1

image

$A(x, y)$

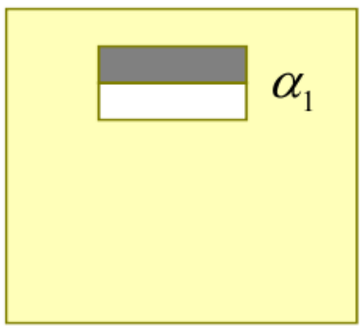
1	6	**8**
3	12	15
5	15	**19**

integral image

$$\begin{aligned} A(1,1,3,3) &= A(3,3) - A(1,3) - A(3,1) + A(1,1) \\ &= 19 - 8 - 5 + 1 \\ &= 7 \end{aligned}$$

Viola & Jones algorithm

1. Evaluate each rectangle filter on each example



$(x_1, 1)$ $(x_2, 1)$ $(x_3, 0)$ $(x_4, 0)$ $(x_5, 0)$ $(x_6, 0)$

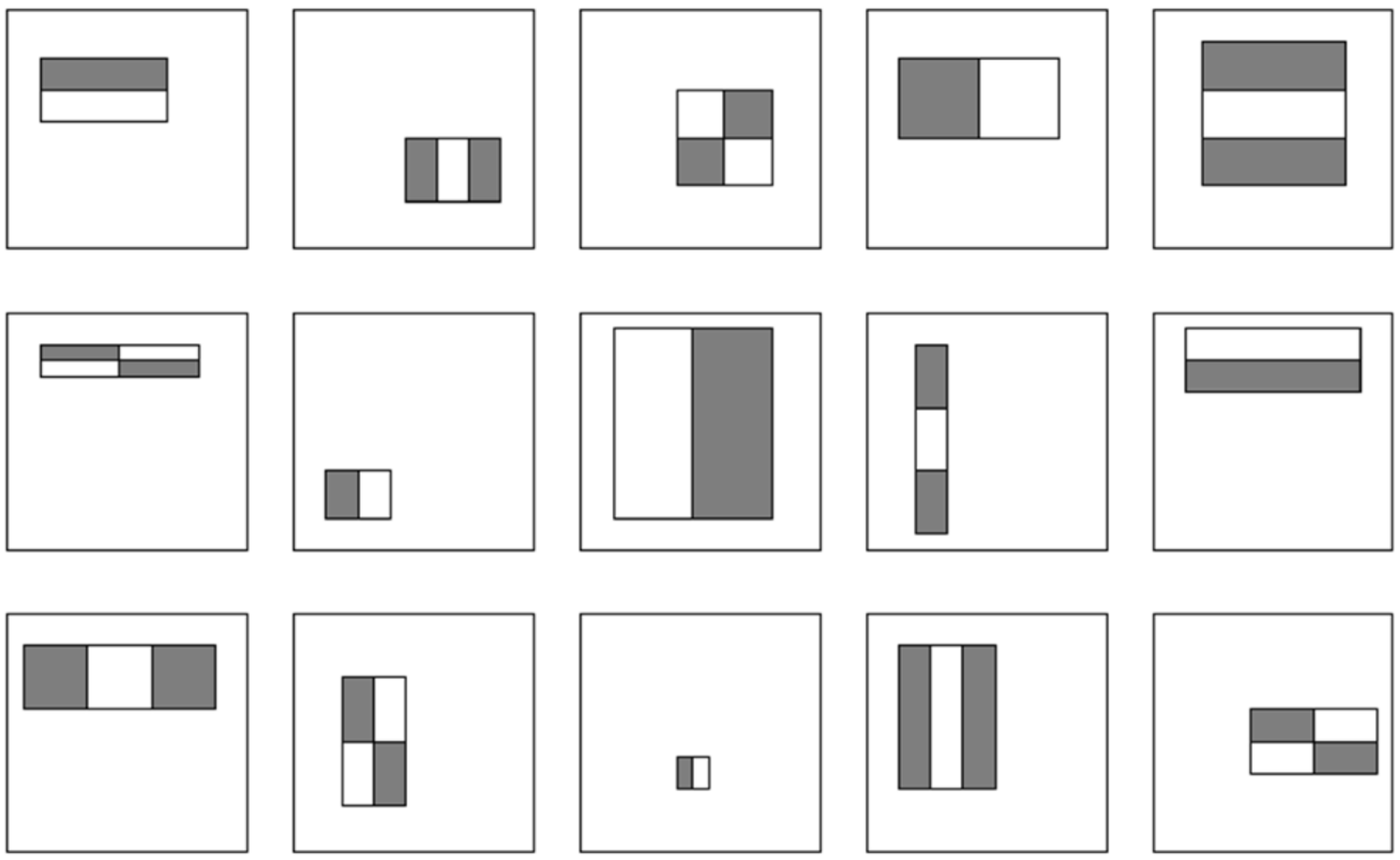
0.8 0.7 0.2 0.3 0.8 0.1 ····· (x_n, y_n)

Weak classifier $h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$ ← threshold

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Viola & Jones algorithm

- For a 24x24 detection region,



P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Viola & Jones algorithm

2. Select best filter/threshold combination

a. Normalize the weights $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

$$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

b. For each feature, j $$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

c. Choose the classifier, h_t with the lowest error ε_t

3. Reweight examples

$$w_{t+1,i} = w_{t,i} \beta_t^{1-|h_t(x_i)-y_i|}$$

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Viola & Jones algorithm

4. The final strong classifier is

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_t = \log \frac{1}{\beta_t}$$

The final hypothesis is a weighted linear combination of the T hypotheses where the weights are inversely proportional to the training errors

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Viola & Jones algorithm

- A “paradigmatic” method for real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Credit slide: S. Lazebnik

The implemented system

- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

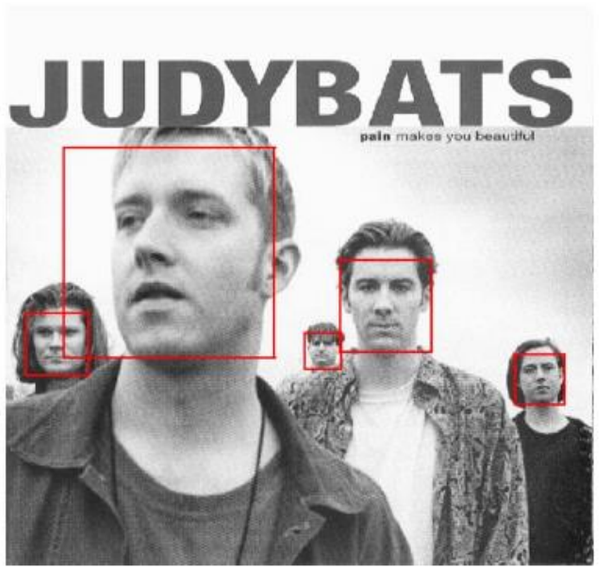
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

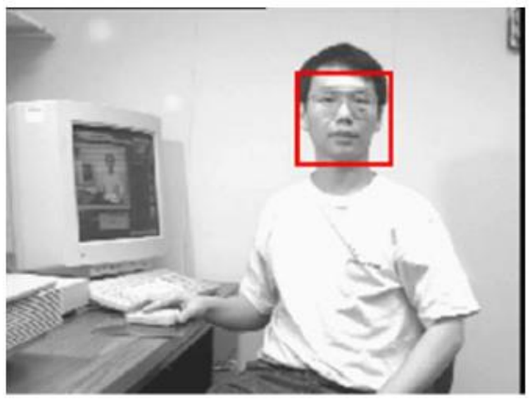
System performance

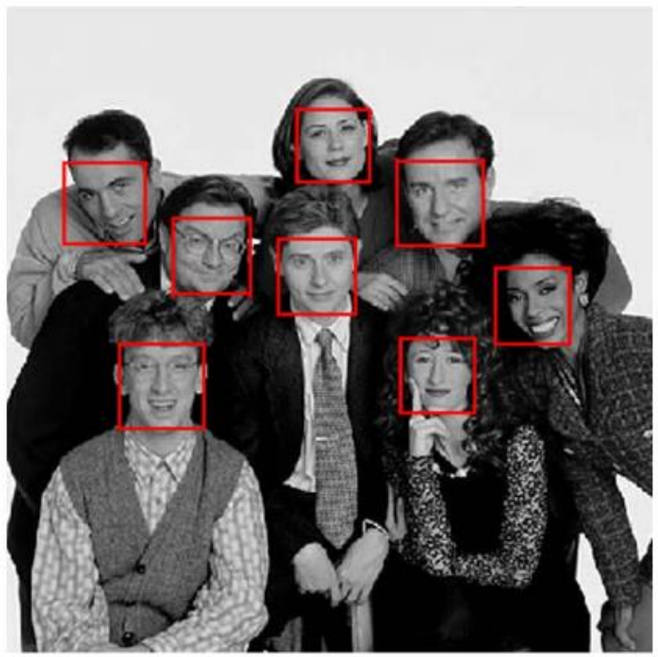
- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Output of Face Detector on Test Images



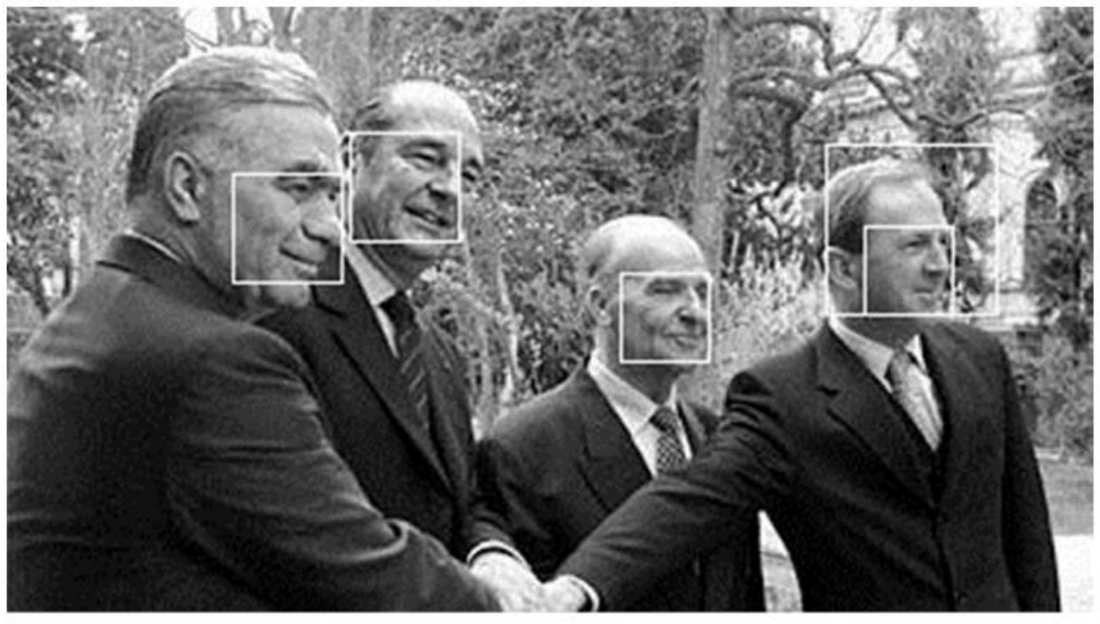




P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

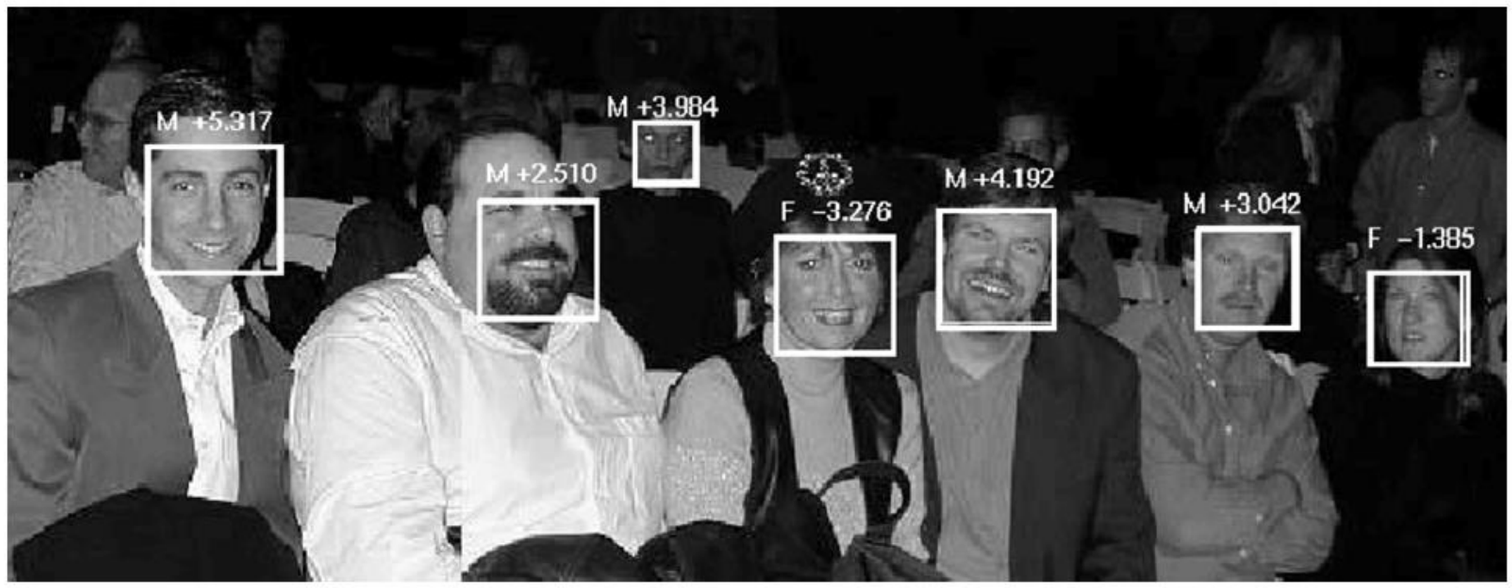
Other detection tasks

Facial Feature Localization

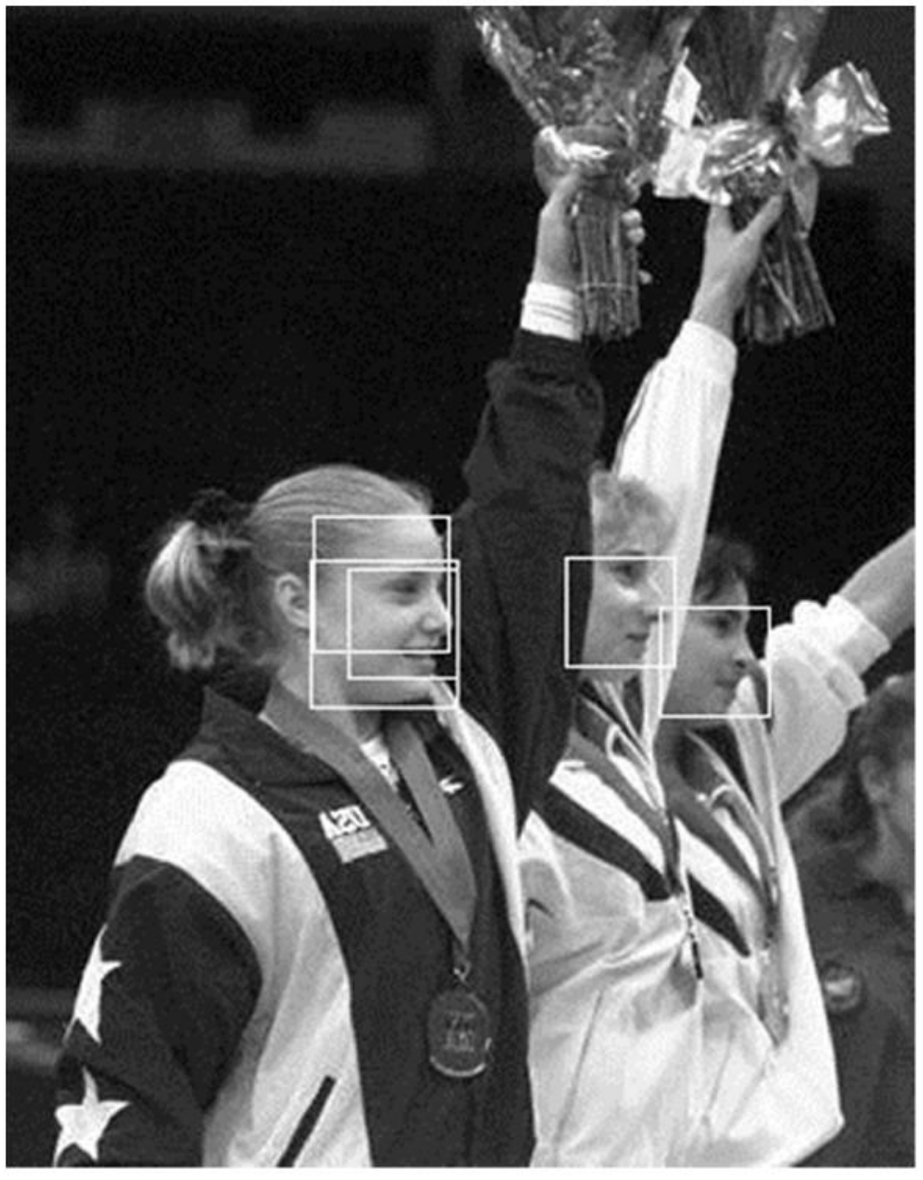


Profile Detection

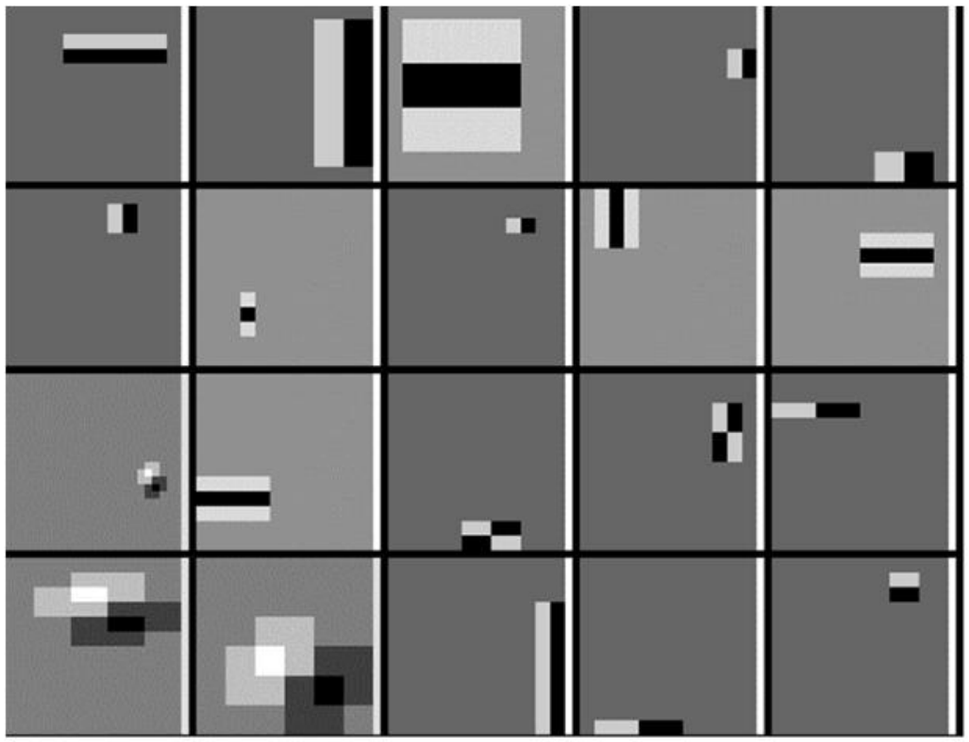
Male vs. female



Profile Detection



Profile Features



Face Image Databases

- Databases for face recognition can be best utilized as training sets
  - Each image consists of an individual on a uniform and uncluttered background
- Test Sets for face detection
  - MIT, CMU (frontal, profile), Kodak

Experimental Results

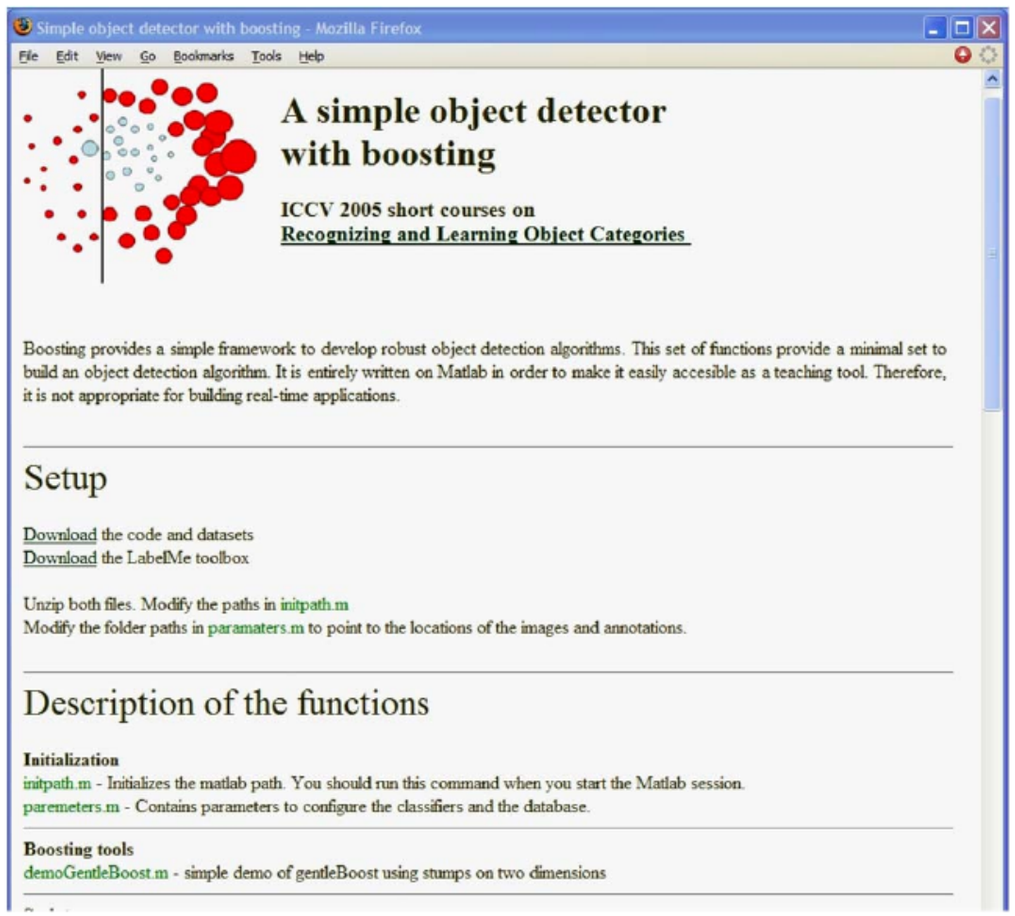
- Test dataset
  - MIT+CMU frontal face test set
  - 130 images with 507 labeled frontal faces

False detection	10	31	50	65	78	95	110	167	422
AdaBoost	78.3	85.2	88.8	89.8	90.1	90.8	91.1	91.8	93.7
Neural-net	83.2	86.0	-	-	-	89.2	-	90.1	89.9

MIT test set: 23 images with 149 faces
Sung & poggio: detection rate 79.9% with 5 false positive
AdaBoost: detection rate 77.8% with 5 false positives

Sharing features with Boosting

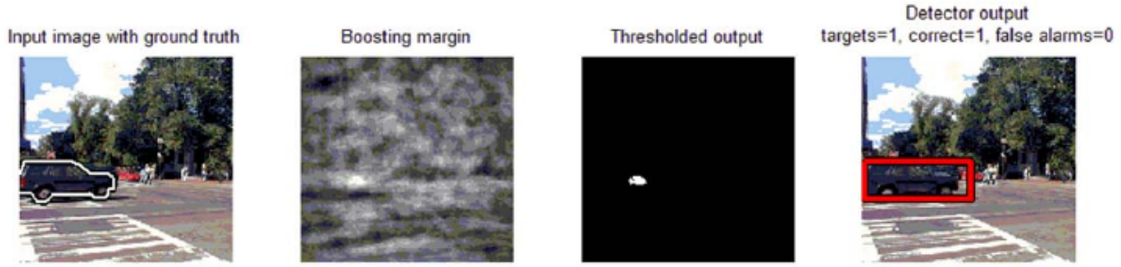
Sharing features: efficient boosting procedures for multiclass object detection
A. Torralba, K. P. Murphy and W. T. Freeman Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). pp 762-769, 2004.



http://people.csail.mit.edu/torralba/iccv2005/

Matlab code

- Gentle boosting
- Object detector using a part based model



References

Basic reading:

- Szeliski, Sections 14.1.1, 14.2.