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:

- Fei-Fei Li (Stanford University).
- Jia-Bin Huang (Virginia Tech).
What’s ‘recognition’?

vs.

Identification
What’s ‘recognition’?

Identification  vs.  Categorization
Yes, there are faces

Identification

Categorization
Yes, there is John Lennon
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

- Digital photography
Face Recognition

- Digital photography
- 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

[Images and text indicating surveillance application of face recognition]
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

1. PCA & Eigenfaces (Turk & Pentland, 1991)
2. LDA & Fisherfaces (Bellumeur et al. 1997)
3. AdaBoost (Viola & Jones, 2001)
Simple idea for face recognition

1. Treat face image as a vector of intensities

   ![Face Image]

   $\mathbf{x}$

2. Recognize face by nearest neighbor in database

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

   $k = \arg\min_k ||\mathbf{y}_k - \mathbf{x}||$
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 than faces!
The Space of Faces

- An image is a point in a high dimensional space
  - If represented in grayscale intensity, an $N \times M$ image is a point in $\mathbb{R}^{NM}$
  - 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

- An image is a point in a high dimensional space
- If represented in grayscale intensity, an $N \times M$ image is a point in $\mathbb{R}^{NM}$
- 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
• Maximize the scatter of the training images in 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

Training images: $x_1, \ldots, x_N$
Top eigenvectors: $\phi_1, \ldots, \phi_k$

Mean: $\mu$
Visualization of eigenfaces

Principal component (eigenvector) $\phi_k$

$\mu + 3\sigma_k \phi_k$

$\mu - 3\sigma_k \phi_k$
Eigenface algorithm

• Training

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

   Note that each image is formulated into a long vector!

2. Compute average face

\[ \frac{1}{N} \sum_{i=1}^{N} x_i \]

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

\[
X_c = X - \mu 1^T = X - \frac{1}{n}X11^T = X\left(I - \frac{1}{n}11^T\right)
\]
4. Compute the covariance matrix

\[
\Sigma = \frac{1}{n} \begin{bmatrix}
0 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & 0
\end{bmatrix} = \frac{1}{n} X_c X_c^T
\]

4. Compute the eigenvectors of the covariance matrix \( \Sigma \)

5. Compute each training image \( x_i \) 's projections as

\[
x_i \rightarrow \begin{pmatrix} x_i^c \cdot 1, x_i^c \cdot 2, \ldots, x_i^c \cdot K \end{pmatrix} \equiv (a_1, a_2, \ldots, a_K)
\]

\[
x_i = a_1 f_1 + a_2 f_2 + \ldots + a_K f_K
\]

6. Visualize the estimated training face \( x_i \)
Eigenface algorithm

\[ x_i \rightarrow (x_i^c \cdot a_1, x_i^c \cdot a_2, \ldots, x_i^c \cdot a_K) \equiv (a_1, a_2, \ldots, a_K) \]

\[ x_i + a_1 f_1 + a_2 f_2 + \ldots + a_K f_K \]

Reconstructed training face
Eigenvalues (variance along eigenvectors)
Reconstruction and Errors

- Only selecting the top $K$ eigenfaces $\rightarrow$ 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

$$t \rightarrow ((t \times f_1), (t \times f_2), \ldots, (t \times f_K)) \equiv (w_1, w_2, \ldots, w_K)$$

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 $m = 1$):

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

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

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

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

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

- We want a projection that maximizes:

$$J(w) = \max \frac{\text{between class scatter}}{\text{within class scatter}}$$
Fischer’s Linear Discriminant Analysis

Between class scatter

Within class scatter

good projection

bad projection
LDA with 2 variables

The following objective function:

\[ J(w) = \max \frac{\text{between class scatter}}{\text{within class scatter}} \]

Can be written as

\[ J(w) = \frac{(E_{Z|Y}[Z|Y=1] - E_{Z|Y}[Z|Y=0])^2}{\text{var}[Z|Y=1] + \text{var}[Z|Y=0]} \]
LDA with 2 variables

- We can write the between class scatter as:

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

\[
= w^T [\mu_1 - \mu_0] \left[ [\mu_1 - \mu_0]^T w \right]
\]

- Also, the within class scatter becomes:

\[
\text{var}[Z \mid Y = i] = E_{Z|Y} \left\{ (z - E_{Z|Y}[Z \mid Y = i])^2 \mid Y = i \right\}
\]

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

\[
= E_{Z|Y} \left\{ w^T [x - \mu_i] [x - \mu_i]^T w \mid Y = i \right\}
\]

\[
= w^T \Sigma_i w
\]
LDA with 2 variables

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

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

\[ S_B = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T \]
\[ S_W = (\Sigma_1 + \Sigma_0) \]

- And our objective becomes:

\[ J(w) = \frac{\left( E_{Z|Y}[Z|Y=1] - E_{Z|Y}[Z|Y=0] \right)^2}{\text{var}[Z|Y=1] + \text{var}[Z|Y=0]} \]
\[ = \frac{w^T (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T (\Sigma_1 + \Sigma_0) w} \]
LDA with 2 variables

- The scatter variables

\[
S_B = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T \\
S_W = (\Sigma_1 + \Sigma_0)
\]
$S_W = S_1 + S_2$

Within class scatter

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
  \[ L = w^T S_B w - \lambda(w^T S_W w - K) \]
- And maximize with respect to both w and \( \lambda \)
Linear Discriminant Analysis (LDA)

- Setting the gradient of
  \[ L = w^T (S_B - \lambda S_W)w + \lambda K \]

  With respect to \( w \) to zeros we get
  \[ \nabla_w L = 2(S_B - \lambda S_W)w = 0 \]
  \[ S_B w = \lambda S_W w \]

- This is a generalized eigenvalue problem

- The solution is easy when
  \[ S_w^{-1} = (\Sigma_1 + \Sigma_0)^{-1} \] 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: \( \{x_1, \ldots, x_N\} \)

- 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 $z$:

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

• 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:

$$W_{opt} = \arg \max_w \frac{\tilde{S}_B}{\tilde{S}_W} = \arg \max_w \frac{W^T S_B W}{W^T S_W W}$$
Mathematical Formulation

 Wir opt = arg \( \max W \left| \frac{W^T S_B W}{W^T S_W W} \right| \)

- Solve generalized eigenvector problem:

\[ S_B w_i = \lambda_i S_W w_i \quad i = 1, \ldots, m \]
Mathematical Formulation

• Solution: Generalized Eigenvectors

\[ S_B w_i = \lambda_i S_W w_i \quad i = 1, \ldots, m \]

• Rank of \( W_{opt} \) is limited
  – \( \text{Rank}(S_B) \leq |C| - 1 \)
  – \( \text{Rank}(S_W) \leq N - C \)
PCA vs. LDA

- Eigenfaces exploit the max scatter of the training images in face space.
- Fisherfaces attempt to maximise the \textbf{between class scatter}, while minimising the \textbf{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

With glasses | Without glasses | 3 Lighting conditions | 5 expressions
---|---|---|---

![Images of faces showing variation in facial expression, eyewear, and lighting conditions.](image-url)
Eigenfaces vs. Fisherfaces

Graph showing error rate against number of principal components for Eigenfaces and Fisherfaces. The graph indicates that Fisherfaces have a lower error rate compared to Eigenfaces, particularly when using fewer principal components.
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
Tel Aviv University
Tel Aviv, Israel
{yaniv, mingyang, ranzato}@fb.com
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

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)

MegaFace Benchmark

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

1. PCA & Eigenfaces (Turk & Pentland, 1991)
2. LDA & Fisherfaces (Bellumeur et al. 1997)
3. AdaBoost (Viola & Jones, 2001)
Detecting foreground objects: A binary classification formulation
Linear classifiers

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

\[ x_i \text{ positive: } x_i \cdot w + b \geq 0 \]
\[ x_i \text{ negative: } x_i \cdot w + b < 0 \]
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:

\[ \Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]

lifting transformation

Slide credit: Andrew Moore
Boosting


Each data point has a class label:

\[ y_t = \begin{cases} 
+1 \ (\text{red}) \\
-1 \ (\text{blue}) 
\end{cases} \]

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:

\[ y_t = \begin{cases} 
+1 & ( \text{red} ) \\
-1 & ( \text{blue} ) 
\end{cases} \]

and a weight:

\[ w_t = 1 \]

\[ h \Rightarrow p(\text{error}) = 0.5 \text{ it is at chance} \]
Each data point has a class label:

\[ y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\oplus) \end{cases} \]

and a weight:

\[ w_t = 1 \]

This one seems to be the best

This is a ‘weak classifier’: It performs slightly better than chance.
Each data point has a class label:

\[ y_t = \begin{cases} 
+1 (\text{red}) \\
-1 (\text{blue}) 
\end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
Toy example

Each data point has a class label:

\[ y_t = \begin{cases} 
+1 & \text{ (red) } \\
-1 & \text{ (blue) } 
\end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
Toy example

Each data point has a class label:

\[ y_t = \begin{cases} 
+1 & (\text{red}) \\
-1 & (\text{blue}) 
\end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
Toy example

Each data point has a class label:

\[ y_t = \begin{cases} 
+1 & (\text{red}) \\
-1 & (\text{blue}) 
\end{cases} \]

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:

\[ h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) + \cdots \]
Boosting

• Defines a classifier using an additive model:

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

• 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

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

- 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 =

$$\sum \text{ (pixels in white area) } - \sum \text{ (pixels in black area) }$$

Credit slide: S. Lazebnik
Weak classifier

Source

Result

Credit slide: S. Lazebnik
Multi-Scale Oriented Patches (MOPS)


Given a feature \((x, y, s, \theta)\)

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.

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

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

original image

\[
\begin{bmatrix}
1 & 5 & 2 \\
2 & 4 & 1 \\
2 & 1 & 1 \\
\end{bmatrix}
\]

integral image

\[
\begin{bmatrix}
1 & 6 & 8 \\
3 & 12 & 15 \\
5 & 15 & 19 \\
\end{bmatrix}
\]
Integral Image

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?

\[ A(x_1, y_1, x_2, y_2) = A(x_2, y_2) - A(x_1, y_2) - A(x_2, y_1) + A(x_1, y_1) \]

\[
\begin{array}{ccc}
1 & 5 & 2 \\
2 & 4 & 1 \\
2 & 1 & 1 \\
\end{array}
\]

image

\[
\begin{array}{ccc}
1 & 6 & 8 \\
3 & 12 & 15 \\
5 & 15 & 19 \\
\end{array}
\]

integral image

\[
A(1, 1, 3, 3) = A(3, 3) - A(1, 3) - A(3, 1) + A(1, 1) \\
= 19 - 8 - 5 + 1 \\
= 7
\]
Viola & Jones algorithm

1. Evaluate each rectangle filter on each example

\[
\begin{align*}
(x_1, 1) & & (x_2, 1) & & (x_3, 0) & & (x_4, 0) & & (x_5, 0) & & (x_6, 0) & & \cdots & & (x_n, y_n) \\
0.8 & & 0.7 & & 0.2 & & 0.3 & & 0.8 & & 0.1 & & \cdots
\end{align*}
\]

Weak classifier

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

Viola & Jones algorithm

- For a 24x24 detection region,

Viola & Jones algorithm

2. Select best filter/threshold combination

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

b. For each feature, \( j \)
   \[ \varepsilon_j = \sum_i w_i |h_j(x_i) - y_i| \]

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

c. Choose the classifier, \( h_t \) with the lowest error \( \varepsilon_t \)

3. Reweight examples

   \[ w_{t+1,i} = w_{t,i} \beta_t^{1 - |h_t(x_i) - y_i|} \]
   \[ \beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t} \]

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.

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

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

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)

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

<table>
<thead>
<tr>
<th>False detection</th>
<th>10</th>
<th>31</th>
<th>50</th>
<th>65</th>
<th>78</th>
<th>95</th>
<th>110</th>
<th>167</th>
<th>422</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost</td>
<td>78.3</td>
<td>85.2</td>
<td>88.8</td>
<td>89.8</td>
<td>90.1</td>
<td>90.8</td>
<td>91.1</td>
<td>91.8</td>
<td>93.7</td>
</tr>
<tr>
<td>Neural-net</td>
<td>83.2</td>
<td>86.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>89.2</td>
<td>-</td>
<td>90.1</td>
<td>89.9</td>
</tr>
</tbody>
</table>

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

Matlab code
- Gentle boosting
- Object detector using a part based model

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

Basic reading:
• Szeliski, Sections 14.1.1, 14.2.