Adaboost for Face Detection

Slides adapted from P. Viola and Tai-Wan Yue
The Task of Face Detection
Basic Idea

Slide a window across image and evaluate a face model at every location; no. of locations where face is present is very very small
1. **hypothesize:**
try all possible rectangle locations, sizes

2. **test:**
classify if rectangle contains a face (and only the face)

Note: 1000's more false windows then true ones.
Classification (Discriminative)

In some feature space
Challenges

- Slide a window across image and **evaluate a face model at every location**
- Sliding window detector **must evaluate tens of thousands of location/scale combinations**
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to **spend as little time as possible** on the non-face windows
  - A megapixel image has $\sim 10^6$ pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image, the **false positive rate** has to be **less than** $10^{-6}$
The Viola/Jones Face Detector

- A seminal approach to **real-time object detection**
- **Key ideas**
  - *Integral images* for **fast feature evaluation**
  - *Boosting* for **feature selection**
  - *Attentional cascade* for **fast rejection of non-face windows**


Image Features

**Rectangular filters**

\[ g(x) = \text{sum(WhiteArea)} - \text{sum(BlackArea)} \]

**Local features:** Subtract sum of pixels in white area from the sum of pixels in black area; 2-rectangle features (A and B), 3-rectangle feature (C) and 4-rectangle feature (D)
Image Features

Local features: Subtract sum of pixels in white area from the sum of pixels in black area.

Rectangular filters

Too many features
In a 24 x 24 patch with 4x4 detector, there are over 160,000 locations for rectangles.
Image Features

\[ F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + ... \]

\[ f_i(x) = \begin{cases} 1 & \text{if } g_i(x) > \theta_i \\ -1 & \text{otherwise} \end{cases} \]

Need to: (1) Select Features \( i=1..n \),
(2) Learn thresholds \( \theta_i \),
(3) Learn weights \( \alpha_i \)
Why rectangle features? (1)
The Integral Image

• The *integral image* computes a value at each pixel \((x,y)\) that is the sum of the pixel values above and to the left of \((x,y)\), inclusive.

• This can quickly be computed in one pass through the image.
Why rectangle features? (2)
Computing Sum within a Rectangle

• Let A, B, C, D be the values of the integral image at the corners of a rectangle

• Then the sum of original image values within the rectangle can be computed:

$$\text{sum} = A - B - C + D$$

• Only 3 additions are required for any size of rectangle!
  - This is now used in many areas of computer vision
Boosting

How to select the best features?

How to learn the classification function?

\[ F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \ldots \]
Boosting

- Defines a classifier using an additive model:

\[ F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots \]
**Image Features**

- **Integral Image**: An intermediate representation of the image for rapid calculation of rectangle features

- \( s(x,y) \) is the cumulative row sum, \( s(x,-1) = 0 \) and \( ii(-1, y) = 0 \); the integral image can be computed in one pass over the original image

\[
ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')
\]

\[
s(x, y) = s(x, y - 1) + i(x, y)
\]

\[
ii(x, y) = ii(x - 1, y) + s(x, y)
\]

- Reject non-face windows through a cascade of classifiers and boosting
Cascaded Classifier

• A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.

• A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative) – using data from previous stage.

• A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)
Each data point has a class label: $y_t = \begin{cases} +1 \ (\bullet) \\ -1 \ (\circ) \end{cases}$ and a weight: $w_t = 1$
Toy example

Weak learners from the family of lines

Each data point has a class label:

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

and a weight:

\[ w_t = 1 \]

\[ h \Rightarrow p(\text{error}) = 0.5 \text{ it is at chance} \]
This one seems to be the best

This is a ‘weak classifier’: It performs slightly better than chance.
We set a new problem for which the previous weak classifier performs at chance again.

Each data point has a class label:

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

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
We set a new problem for which the previous weak classifier performs at chance again.

Each data point has a class label:

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

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
We set a new problem for which the previous weak classifier performs at chance again.

Each data point has a class label:

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

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
We set a new problem for which the previous weak classifier performs at chance again.

Each data point has a class label:

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

We update the weights:

\[w_t \leftarrow w_t \exp\{-y_t H_t\}\]
The strong (non-linear) classifier is built as the combination of all the weak (linear) classifiers.
Face Detector

• Scan the input at many scales
• Starting at the base scale in which faces are detected at 24x24 pixels, a 384x288 pixel image is scanned at 12 scales each a factor 1.25 larger than the last
• Any rectangle feature can be evaluated at any scale and location in a few operations
• Face detection @15 fps for the entire image
Weak Learners for Face Detection

- Hypothesis: A very small no. of 160K (x4) features for each image sub-window can be formed to find an effective classifier (face vs. non-face)
- AdaBoost is used both to select the features and train the classifier
- Weak learner: a single rectangle feature that best separates positive and negative examples; so weak classifier is a thresholded single feature (can be viewed as a single node decision tree)

$$h_t(x) = \begin{cases} 
1 & \text{if } f_t(x) > \theta_t \\
0 & \text{otherwise}
\end{cases}$$
Cascade of classifiers

- Train each classifier with increasing number of features until the target false positive and detection rates are achieved.
- Use false positives from current stage as the negative training examples for the next stage.
- Classifiers are progressively more complex and have lower false positive rates.

False positive rate of cascade $F = \prod_{i=1}^{K} f_i$
Detection rate of cascade $D = \prod_{i=1}^{K} d_i$

ROC Curve

% False Pos

% Detection

0 50 100
Boosting

• Training set contains face and nonface examples
  • Initially, with equal weight
• For each round of boosting:
  • Evaluate each rectangle filter on each example
  • Select best threshold for each filter
  • Select best filter/threshold combination
  • Reweight examples
• Computational complexity of learning: \( O(MNK) \)
  • \( M \) rounds, \( N \) examples, \( K \) features
A variant of AdaBoost for aggressive feature selection

• Given example images \((x_1,y_1), \ldots, (x_n,y_n)\) where \(y_i = 0, 1\) for negative and positive examples respectively.

• Initialize weights \(w_{1,i} = 1/(2m), 1/(2l)\) for training example \(i\), where \(m\) and \(l\) are the number of negatives and positives respectively.

For \(t = 1 \ldots T\)

1) Normalize weights so that \(w_t\) is a distribution

2) For each feature \(j\) train a classifier \(h_j\) and evaluate its error \(\varepsilon_j\) with respect to \(w_t\).

3) Chose the classifier \(h_j\) with lowest error.

4) Update weights according to:

\[
\beta_t \varepsilon_i - = + 1, 1, 1 \begin{cases} w_{t+1,i} = w_{t,i} \beta_t \varepsilon_i \\ \text{where } e_i = 0 \text{ is } x_i \text{ is classified correctly, } 1 \text{ otherwise, and } \\ \beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t} \end{cases}
\]

• The final strong classifier is:

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

where \(\alpha_i = \log \left( \frac{1}{\beta_t} \right)\)
Algorithm for training a cascade of classifiers

• User selects values for $f$, the maximum acceptable false positive rate per layer and $d$, the minimum acceptable detection rate per layer.
• User selects target overall false positive rate $F_{\text{target}}$.
• $P =$ set of positive examples
• $N =$ set of negative examples
• $F_0 = 1.0; D_0 = 1.0; i = 0$

While $F_i > F_{\text{target}}$
  
  $i++$
  
  $n_i = 0; F_i = F_{i-1}$
  
  while $F_i > f \times F_{i-1}$
    
    $n_i++$
    
    o Use $P$ and $N$ to train a classifier with $n_i$ features using AdaBoost
    
    o Evaluate current cascaded classifier on validation set to determine $F_i$ and $D_i$
    
    o Decrease threshold for the $i$th classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects $F_i$)

$N = \emptyset$

If $F_i > F_{\text{target}}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set $N$. 
ROC Curves Cascaded Classifier to Monolithic Classifier

Speed of cascade classifier is 10 times faster
There is little difference between the two in terms of accuracy.

There is a big difference in terms of speed.

The cascaded classifier is nearly 10 times faster since its first stage throws out most non-faces so that they are never evaluated by subsequent stages.
Face Detection System

- **Training Data**
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 9500 million non-faces
  - Faces are normalized
    - Scale, translation

- **Many variations**
  - Across individuals
  - Illumination
  - Pose
Structure of the Detector Cascade

Combining successively more complex classifiers in cascade

- 32 stages
- included a total of 4297 features
Structure of the Detector Cascade

- All Sub-Windows
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
  - 8
  - 32

- 2 features, reject 50% non-faces, detect 100% faces
- 10 features, reject 80% non-faces, detect 100% faces
- 25 features
- 50 features
- by algorithm

Face

Reject Sub-Window
Feature Selection

Figure 5: The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.
Speed of the Final Detector

• On a 700 Mhz Pentium III processor, the face detector can process a $384 \times 288$ pixel image in about 0.067 seconds.

• 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

• Average of 8 features evaluated per window on test set
Output of Face Detector on Test Images