Face Detection

CSE 576
Face detection

State-of-the-art face detection demo
(Courtesy Boris Babenko)
Face detection and recognition

Detection

Recognition

“Sally”
Face detection

• Where are the faces?
Face Detection

• What kind of features?

• What kind of classifiers?
Image Features

“Rectangle filters”

\[ \text{Value} = \sum (\text{pixels in white area}) - \sum (\text{pixels in black area}) \]
Feature extraction

“Rectangular” filters

Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images scale features directly for same cost

Viola & Jones, CVPR 2001
Sums of rectangular regions

How do we compute the sum of the pixels in the red box?

After some pre-computation, this can be done in constant time for any box.

This “trick” is commonly used for computing Haar wavelets (a fundamental building block of many object recognition approaches.)
Sums of rectangular regions

The trick is to compute an "integral image." Every pixel is the sum of its neighbors to the upper left.

Sequentially compute using:

\[ I(x, y) = I(x, y) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1) \]
Sums of rectangular regions

Solution is found using:

\[ A + D - B - C \]

What if the position of the box lies between pixels?
Large library of filters

Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

Viola & Jones, CVPR 2001
Feature selection

• For a 24x24 detection region, the number of possible rectangle features is ~180,000!
• At test time, it is impractical to evaluate the entire feature set
• Can we create a good classifier using just a small subset of all possible features?
• How to select such a subset?
AdaBoost for feature+classifier selection

- Want to select the single rectangle feature and a corresponding threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error.

Resulting weak classifier:

\[ h_t(x) = \begin{cases} 
+1 \text{ if } f_t(x) > \theta_t \\
-1 \text{ otherwise} 
\end{cases} \]

For next round, reweight the examples according to errors, choose another filter/threshold combo.

Viola & Jones, CVPR 2001
AdaBoost: Intuition

Consider a 2-d feature space with positive and negative examples.

Each weak classifier splits the training examples with at least 50% accuracy.

Examples misclassified by a previous weak learner are given more emphasis at future rounds.

Figure adapted from Freund and Schapire

K. Grauman, B. Leibe
AdaBoost: Intuition

Weak Classifier 1

Weights Increased

Weak Classifier 2

K. Grauman, B. Leibe
AdaBoost: Intuition

Final classifier is combination of the weak classifiers

K. Grauman, B. Leibe
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 \( \alpha_t = \log \frac{1}{\beta_t} \)

Final classifier is combination of the weak ones, weighted according to error they had.
AdaBoost Algorithm

Start with uniform weights on training examples

\{x_1, \ldots, x_n\}

For T rounds

1. Normalize the weights,

\[ w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \]

so that \( w_t \) is a probability distribution.

2. For each feature, \( j \), train a classifier \( h_j \) which is restricted to using a single feature. The error is evaluated with respect to \( w_t \), \( \epsilon_j = \sum_i w_i |h_j(x_i) - y_i| \).

3. Choose the classifier, \( h_t \), with the lowest error \( \epsilon_t \).

4. Update the weights:

\[ w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_i} \]

where \( \epsilon_i = 0 \) if example \( x_i \) is classified correctly, \( \epsilon_i = 1 \) otherwise, and \( \beta_t = \frac{\epsilon_t}{1-\epsilon_t} \).

Find the best threshold and polarity for each feature, and return error.

Re-weight the examples:
Incorrectly classified \( \rightarrow \) more weight
Correctly classified \( \rightarrow \) less weight

- 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 \( w_{1,i} = \frac{1}{2m}, \frac{1}{2l} \) for \( y_i = 0, 1 \) respectively, where \( m \) and \( l \) are the number of negatives and positives respectively.
- For \( t = 1, \ldots, T \):

  1. Normalize the weights,

  \[ w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \]

  so that \( w_t \) is a probability distribution.

  2. For each feature, \( j \), train a classifier \( h_j \) which is restricted to using a single feature. The error is evaluated with respect to \( w_t \), \( \epsilon_j = \sum_i w_i |h_j(x_i) - y_i| \).

  3. Choose the classifier, \( h_t \), with the lowest error \( \epsilon_t \).

  4. Update the weights:

  \[ w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_i} \]

  where \( \epsilon_i = 0 \) if example \( x_i \) is classified correctly, \( \epsilon_i = 1 \) otherwise, and \( \beta_t = \frac{\epsilon_t}{1-\epsilon_t} \).
Picking the best classifier

Efficient single pass approach:

\[ f_t = \min ( S + (T - S), S + (T - S) ) \]

At each sample compute:

Find the minimum value of \( e \), and use the value of the corresponding sample as the threshold.

\( S = \) sum of samples below the current sample
\( T = \) total sum of all samples
# Measuring classification performance

- **Confusion matrix**

- **Accuracy**
  - \[ \frac{(TP+TN)}{(TP+TN+FP+FN)} \]

- **True Positive Rate=Recall**
  - \[ \frac{TP}{(TP+FN)} \]

- **False Positive Rate**
  - \[ \frac{FP}{(FP+TN)} \]

- **Precision**
  - \[ \frac{TP}{(TP+FP)} \]

- **F1 Score**
  - \[ \frac{2\times \text{Recall} \times \text{Precision}}{(\text{Recall} + \text{Precision})} \]

### Table: Confusion Matrix

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Class1</th>
<th>Class2</th>
<th>Class3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class1</td>
<td>40</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Class2</td>
<td>3</td>
<td>25</td>
<td>7</td>
</tr>
<tr>
<td>Class3</td>
<td>4</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

### Table: True Positive (TP) and False Positive (FP)

<table>
<thead>
<tr>
<th>Actual</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>True Positive</td>
<td>False Negative</td>
</tr>
<tr>
<td>Negative</td>
<td>False Positive</td>
<td>True Negative</td>
</tr>
</tbody>
</table>
Boosting for face detection

- First two features selected by boosting:

This feature combination can yield 100% detection rate and 50% false positive rate
Boosting for face detection

- A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084

Receiver operating characteristic (ROC) curve
Attentioanal cascade

• We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows

• Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on

• A negative outcome at any point leads to the immediate rejection of the sub-window
Attentional cascade

- Chain classifiers that are progressively more complex and have lower false positive rates:
Attentional cascade

- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages.
- A detection rate of 0.9 and a false positive rate on the order of $10^{-6}$ can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 ($0.3^{10} \approx 6 \times 10^{-6}$).
Training the cascade

• Set target detection and false positive rates for each stage

• Keep adding features to the current stage until its target rates have been met
  – Need to lower AdaBoost threshold to maximize detection
    (as opposed to minimizing total classification error)
  – Test on a validation set

• If the overall false positive rate is not low enough, then add another stage

• Use false positives from current stage as the negative training examples for the next stage
Viola-Jones Face Detector: Summary

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

K. Grauman, B. Leibe
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)
Many detections above threshold.
Non-maximal suppression (NMS)
Similar accuracy, but 10x faster
Viola-Jones Face Detector: Results

K. Grauman, B. Leibe
Viola-Jones Face Detector: Results
Viola-Jones Face Detector: Results

K. Grauman, B. Leibe
Detecting profile faces?

Detecting profile faces requires training separate detector with profile examples.
Viola-Jones Face Detector: Results
Summary: Viola/Jones detector

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows
Recall

- Classification
  - NN
  - Naïve Bayes
  - Logistic Regression
  - Boosting

- Face Detection
  - Simple Features
  - Integral Images
  - Boosting