Viola-Jones Type Face Detection

lecturer: Jiří Matas, matas@cmp.felk.cvut.cz

authors: Jiří Matas, Ondřej Drbohlav

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
16/May/2016

Last update: 15/May/2016
The task is not simple
What is/is not a face?
Viola and Jones Suggested a Brute-Force Search
Viola – Jones (2001)

**Breakthrough #1**

- Speed depends on negative examples only

- This is addressed by sequential decision making

---

![Diagram](image)

- **Input Signal (Image Window)**
- **Stage 1**: 1 Weak Classifier
  - 60% to 60%
  - Class 2 (Non-Face)
  - 40%
- **Stage 2**: 5 Weak Classifiers
  - 36% to 60%
  - ... (Multiple stages)
  - 40%
- **Stage N**: 1200 Weak Classifiers
  - 40%
- **Class 1 (Face)**
Breakthrough #2 – bootstrap (hard negative mining)

- Classifiers further down the cascade are trained on samples which had not been already discarded by previous classifiers (by being classified as a non-face). Their training set is thus harder but orders of magnitude smaller than training set faced by preceding classifiers. The classifiers thus can be designed to be increasingly more complex.

T: classified as a face (true positives and false positives), goes further down the cascade

F: classified as a non-face (true and false negatives), dropped from the pipeline

![Receiver operating characteristic (ROC)](image)
Face Detector, Hard Negative Examples

Images classified as faces by early cascade components
Breakthrough #3: Fast features

- Gabor filters had been commonly used as features of choice; they are nice but expensive to compute.

- Viola-Jones have approximated Gabors by piecewise constant functions - Haar wavelets.

Example:

\[ \psi(t) = \begin{cases} 
1 & \text{for } 0 \leq t < 0.5 \\
-1 & \text{for } 0.5 \leq t < 1 \\
0 & \text{otherwise} 
\end{cases} \]
Fast Calculation of Haar Wavelets

Row sum: \[ s(x, y) = s(x-1, y) + i(x, y) \]
Integral image: \[ ss(x, y) = ss(x, y-1) + s(x, y) \]

MATLAB: \[
ss = \text{cumsum}(
\text{cumsum}(\text{double}(i)), 2);
\]
Fast Calculation of Haar Wavelets

- values at A, B, C, D are read out form the integral image

- Sum of the intensities within the rectangle is equal to:
  \[ \text{sum} = A - B - C + D \]

- Each rectangle requires 3 addition/subtraction operations!
Breakthrough #4

• VJ have employed AdaBoost (Schapire a Freund, 1997) which both trains the classifier and selects the features

• Pros of Adaboost:
  • Well understood
  • Good detection rate (in many applications)
  • Easy to implement (“just 10 lines of code” [R. Schapire])
AdaBoost: Algorithm

Input: \((x_1, y_1), \ldots, (x_L, y_L)\), where \(x_i \in \mathcal{X}\) and \(y_i \in \{-1, +1\}\)

Initialize weights \(D_1(i) = 1/L\).

For \(t = 1, \ldots, T\):

- Find \(h_t = \arg \min_{h \in B} \epsilon_t\); \(\epsilon_t = \sum_{i=1}^{L} D_t(i) [y_i \neq h(x_i)]\) (WeakLearn)
  \[\text{[true]} \overset{\text{def}}{=} 1, \quad \text{[false]} \overset{\text{def}}{=} 0\]

- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)

- Update

\[
D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}, \quad Z_t = \sum_{i=1}^{L} D_t(i)e^{-\alpha_t y_i h_t(x_i)},
\]

where \(Z_t\) is a normalization factor chosen so that \(D_{t+1}\) is a distribution.

Output the final classifier:

\[
H(x) = \text{sign}(f(x)), \quad f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)
\]
### AdaBoost – Example 1

<table>
<thead>
<tr>
<th>Data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$D_1$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Taken from “A Tutorial on Boosting” by Yoav Freund and Rob Schapire
Example 1 – Iteration 1

<table>
<thead>
<tr>
<th>Data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( D_2 \cdot Z_2 )</td>
<td>0.07</td>
<td>0.07</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
</tbody>
</table>

\[ Z_2 = 0.92 \]

\( \epsilon_1 \) ... error

\[ D_2 \approx D_1 \sqrt{\frac{\epsilon_1}{(1 - \epsilon_1)}} \] for corr. class.

\[ \alpha_1 = \frac{1}{2} \log \left( \frac{1 - \epsilon_1}{\epsilon_1} \right) \]

\[ D_2 \approx \frac{D_1 \sqrt{(1 - \epsilon_1)}}{\epsilon_1} \] for wrongly class.
Example 1 – Iteration 2

<table>
<thead>
<tr>
<th>Data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$D_3 \cdot Z_3$</td>
<td>0.04</td>
<td>0.04</td>
<td>0.09</td>
<td>0.09</td>
<td>0.09</td>
<td>0.04</td>
<td>0.14</td>
<td>0.14</td>
<td>0.14</td>
<td>0.04</td>
</tr>
</tbody>
</table>

$Z_3 = 0.82$

$\epsilon_2 \ldots$ error

$\alpha_2 = 1/2 \log(1 - \epsilon_2)/\epsilon_2$

$D_3 \approx D_2 \sqrt{\epsilon_2/(1 - \epsilon_2)}$ for corr. class.

$D_3 \approx D_2 \sqrt{(1 - \epsilon_2)/\epsilon_2}$ for wrongly class.
Example 1 – Iteration 3

<table>
<thead>
<tr>
<th>Data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$D_4 \cdot Z_4$</td>
<td>0.11</td>
<td>0.11</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.11</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.02</td>
</tr>
</tbody>
</table>

$Z_4 = 0.68$

$\epsilon_3$ ... error

$D_4 \approx D_3 \sqrt{\epsilon_3/(1 - \epsilon_3)}$ for corr. class.

$\alpha_3 = 1/2 \log (1 - \epsilon_3)/\epsilon_3$  $D_4 \approx D_3 \sqrt{(1 - \epsilon_3)/\epsilon_3}$ for wrongly class.

$\epsilon_3 = 0.14$

$\alpha_3 = 0.92$
Example 1 – Final Classifier after Iter. 3

\[
H_{\text{final}} = \text{sign} \left( \begin{array}{c}
0.42 \\
+ 0.65 \\
+ 0.92 
\end{array} \right)
\]
Dataset: \((x_1, y_1), \ldots, (x_L, y_L)\), where \(x_i \in \mathcal{X}\) and \(y_i \in \{-1, +1\}\).

The two class distributions do not overlap (Bayes error is 0). The class distributions are not known to AdaBoost.
Example 2, Weak Classifier

Dataset: \((x_1, y_1), \ldots, (x_L, y_L)\), where \(x_i \in \mathcal{X}\) and \(y_i \in \{-1, +1\}\).

The two class distributions do not overlap (Bayes error is 0). The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

\[ h_{w,b}(x) = \text{sign}(w \cdot x + b), \]

where \(w\) is the projection direction vector and \(b\) is the bias.
Example 2, Weak Classifier Set

**Dataset:** \((x_1, y_1), \ldots, (x_L, y_L)\), where \(x_i \in \mathcal{X}\) and \(y_i \in \{-1, +1\}\).

The two class distributions do not overlap (Bayes error is 0). The class distributions are not known to AdaBoost.

**Weak classifier:** a linear classifier

\[ h_{w,b}(x) = \text{sign}(w \cdot x + b), \]

where \(w\) is the projection direction vector and \(b\) is the bias.

**Weak classifier set** \(\mathcal{B}\):

\[ \{h_{w,b} \mid w \in \{w_1, w_2, \ldots, w_N\}, b \in \mathbb{R}\} \]

- \(N\) is the number of projection directions used
Example 2, Weak Classifier Set

Dataset: \((x_1, y_1), \ldots, (x_L, y_L)\), where \(x_i \in \mathcal{X}\) and \(y_i \in \{-1, +1\}\).

The two class distributions do not overlap (Bayes error is 0). The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

\[ h_{w,b}(x) = \text{sign}(w \cdot x + b), \]

where \(w\) is the projection direction vector and \(b\) is the bias.

Weak classifier set \(\mathcal{B}\):

\[ \{ h_{w,b} \mid w \in \{w_1, w_2, \ldots, w_N\}, b \in \mathbb{R} \} \]

- \(N\) is the number of projection directions used
- for each projection direction \(w\), varying bias \(b\) results in different training errors \(\epsilon\).
Example 2, Iteration 1

\[ t = 1 \]

- \( h_1 \) selected (note \( \bullet \) in the polar plot). \( N = 36 \) directions \( w \) are used.
- \( \epsilon_1 < 0.5 \), continue
- \( \alpha_1 = \frac{1}{2} \log(\frac{1-\epsilon_1}{\epsilon_1}) \)
- re-weighting \( D \) puts more weight to mis-classified samples in the \( \bullet \) class
- \( f_1(x) = \alpha_1 h_1(x) \)
- \( f_1'(x) = f_1(x)/\alpha_1 \)
- \( H_1(x) = \text{sign}(f_1(x)) \)
Example 2, Iteration 2

$t = 2$

- minimum errors $\epsilon_2(w)$ for all weak classifier directions $w$ are equal. Everything is classified as $-1$ (●)

- this essentially re-weights the classes; gives more weight to class 1 (●)

- $f_2(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x)$

- $f'_2(x) = \frac{f_2(x)}{\alpha_1 + \alpha_2}$

- $H_2(x) = \text{sign}(f_2(x))$

Quiz question:
- What is the difference between $f_2(x)$ and the previous $f_1(x)$? (Note that all points are classified as $-1$)
Example 2, Iteration 3

\[ t = 3 \]

- \( h_3 \) selected which minimizes \( \epsilon_3(w) \) (note \( \bullet \) in the polar plot).
- \( \alpha_3 = \frac{1}{2} \log \left( \frac{1-\epsilon_3}{\epsilon_3} \right) \)
- distribution re-weighted
- \( f_3(x) = \sum_{q=1}^{3} \alpha_q h_q(x) \)
- \( f'_3(x) = \frac{f(x)}{\sum_{q=1}^{3} \alpha_q} \)
- \( H_3(x) = \text{sign}(f_3(x)) \)
Example 2, Iteration 8

\[ f_8'(x) \]

\[ H_8(x) = \text{sign}(f_8(x)) \]

\[ \epsilon_8(w) \text{ at optimal } b \]

\[ \alpha_8 = \frac{1}{2} \log\left( \frac{1 - \epsilon_8}{\epsilon_8} \right) \]

- \( h_8 \) selected which minimizes \( \epsilon_8(w) \) (note \( \bullet \) in the polar plot).

- Distribution re-weighted

\[ f_8(x) = \sum_{q=1}^{8} \alpha_q h_q(x) \]

\[ f_8'(x) = \frac{f(x)}{\sum_{q=1}^{8} \alpha_q} \]

\[ H_8(x) = \text{sign}(f_8(x)) \]

\[ t = 8 \]

\[ \epsilon_8(w) \text{ at optimal } b \]

\[ \text{training error of } H_t \]

\[ \text{iteration } t \]
Example 2, Iteration 9

\[ t = 9 \]

- \( h_9 \) selected which minimizes \( \epsilon_9(w) \) (note in the polar plot).

- \( \alpha_9 = \frac{1}{2} \log\left(\frac{1-\epsilon_9}{\epsilon_9}\right) \)

- distribution re-weighted

- \( f_9(x) = \sum_{q=1}^{9} \alpha_q h_q(x) \)

- \( f'_9(x) = \frac{f(x)}{\sum_{q=1}^{9} \alpha_q} \)

- \( H_9(x) = \text{sign}(f_9(x)) \)

\(\epsilon_9(w)\) at optimal \( b \)

\begin{align*}
\begin{array}{c}
\text{training error of } H_t \\
\end{array}
\end{align*}

\begin{align*}
\begin{array}{c}
\text{iteration } t \\
\end{array}
\end{align*}
Example 2, Iteration 15

\[ t = 15 \]

- \( h_{15} \) selected which minimizes \( \epsilon_{15}(w) \) (note in the polar plot).
- \( \alpha_{15} = \frac{1}{2} \log\left(\frac{1-\epsilon_{15}}{\epsilon_{15}}\right) \)
- distribution re-weighted
- \( f_{15}(x) = \sum_{q=1}^{15} \alpha_q h_q(x) \)
- \( f'_{15}(x) = \frac{f(x)}{\sum_{q=1}^{15} \alpha_q} \)
- \( H_{15}(x) = \text{sign}(f_{15}(x)) \)
Example 2, Iteration 21

\[ t = 21 \]

- \( h_{21} \) selected which minimizes \( \epsilon_{21}(w) \) (note \( \bullet \) in the polar plot).
- \( \alpha_{21} = \frac{1}{2} \log\left(\frac{1-\epsilon_{21}}{\epsilon_{21}}\right) \)
- Distribution re-weighted
- \( f_{21}(x) = \sum_{q=1}^{21} \alpha_q h_q(x) \)
- \( f'_{21}(x) = \frac{f(x)}{\sum_{q=1}^{21} \alpha_q} \)
- \( H_{21}(x) = \text{sign}(f_{21}(x)) \)

\( \epsilon_{21}(w) \) at optimal \( b \)

- Training error of \( H_t \)


Example 2, Iteration 100

\[ f'_{100}(x) \]

\[ H_{100}(x) = \text{sign}(f_{100}(x)) \]

\[ \epsilon_{100}(w) \text{ at optimal } b \]

\[ \text{training error of } H_t \]

\[ t = 100 \]

- \( h_{100} \) selected which minimizes \( \epsilon_{100}(w) \) (note \( \bullet \) in the polar plot).

- \( \alpha_{100} = \frac{1}{2} \log\left(\frac{1 - \epsilon_{100}}{\epsilon_{100}}\right) \)

- distribution re-weighted

- \( f_{100}(x) = \sum_{q=1}^{100} \alpha_q h_q(x) \)

- \( f'_{100}(x) = \frac{f(x)}{\sum_{q=1}^{100} \alpha_q} \)

- \( H_{100}(x) = \text{sign}(f_{100}(x)) \)
Example 2, \( N=3 \) Proj. Directions, Iteration 100

\[ t = 100 \]

- \( h_{100} \) selected which minimizes \( \epsilon_{100}(w) \) (note \( \bullet \) in the polar plot).
- \( \alpha_{100} = \frac{1}{2} \log(\frac{1-\epsilon_{100}}{\epsilon_{100}}) \)
- distribution re-weighted
- \( f_{100}(x) = \sum_{q=1}^{100} \alpha_q h_q(x) \)
- \( f'_{100}(x) = \frac{f(x)}{\sum_{q=1}^{100} \alpha_q} \)
- \( H_{100}(x) = \text{sign}(f_{100}(x)) \)
Example 3 - Adaboost Detector

- The first two selected classifiers:

The two features have 100% detection rate and 50% false alarm rate
Not every image sub-window must be tested by the classifier. It is sufficient to use:

- shifts by cca 10% window side
- window side size increments of 15%
- window rotation by +/- 15 deg

**Note:**
Total number of sub-windows (thus speed) is determined by the size of smallest face to be detected. Total detection time is the geometric series sum with \( q = 1/1.15^2 \); \( s \approx 4t_0 \)
Historical perspective

- Many improvements since then.
- In 2009, implemented in many digital cameras.
- E.g. Waldboost (developed here on CTU) improves the method by addressing the problem of trade-off between speed and accuracy of the Adaboost classifier.