Imagine this scenario

Instead,

Instead,

Instead,
Instead, individually, none of them can get off the island but…

Strong and Weak learners

- Strong Learner:

Self-Operating Napkin

- Weak Learner:

Boosting:

Combining weak learners
Boosting: An example

Using voting

Final Decision = majority decision of classifiers

Using confidence measures

Final Decision = weighted majority decision of classifiers

But we really aren't making the best use of the classifiers!!

Better way, AdaBoost

Correctly classified data: Multiply by $a_i$

Incorrectly classified data: Multiply by $1/a_i$

where $a_i < 1$
Training Data → Classifier 1 → Classifier 2 → Classifier 3

Correctly classified data:
Multiply by $a_1$
Incorrectly classified data:
Multiply by $1/a_1$

where $a_1 < 1$

**Better way, AdaBoost**

Formalizing the concept

- Training data:
  \[(x_i, y_i)\]
  where $x_i$ is the input to the classifier
  and $y_i$ is the correct output, either +1 or -1

- Set of classifiers: $h_1, h_2, h_3, \ldots, h_T$
  where $h_i(x)$ classifies the input $x$ as +1 or -1
  $y^* h_i(x) = +1$ if correctly classified and -1 if incorrectly classified

Formalizing the concept

- Voting
  - $f(x) = \sum_i a_i h_i(x)$
  - $H(x) = \text{sign}(f(x)) = \text{sign}(\sum_i h_i(x))$

Boosting
Boosting

- Voting
  - \( f(x) = \sum h(x) \)
  - Classifier \( H(x) = \text{sign}(f(x)) = \text{sign}(\sum h(x)) \)

- Using weights:
  - \( f(x) = \sum \alpha_i h(x) \)
  - Classifier \( H(x) = \text{sign}(f(x)) = \text{sign}(\sum \alpha_i h(x)) \)
  - The weight \( \alpha_i \) for any \( h(x) \) is a measure of our trust in \( h(x) \)

AdaBoost

- As before:
  - \( y_i \) is either +1 or -1
  - \( H(x) \) is also either +1 or -1

- Correctly classified: \( y_i H(x) = +1 \)
- Incorrectly classified: \( y_i H(x) = -1 \)

Error function: \( \frac{1}{2} (1 - y_i H(x)) \)
- Correct classification: 0
- Incorrect classification: 1

The AdaBoost Algorithm

- Initialize \( D_1(x) = \frac{1}{N} \)
- For \( t = 1, ..., T \)
  - Train a weak classifier \( h_t \) using distribution \( D_t \)
  - Compute total error on training data
    - \( c_t = \sum (D_t(x) \cdot (1 - y_i h_t(x))) \)
  - Set \( \alpha_t = \frac{1}{2} \ln \left( \frac{1 - c_t}{c_t} \right) \)
  - For \( i = 1, ..., N \)
    - set \( D_{t+1}(x) = D_t(x) \exp(-\alpha_t y_i h_t(x)) \)
  - Normalize \( D_{t+1} \) to make it a distribution
- The final classifier is
  - \( H(x) = \text{sign}(\sum \alpha_i h_i(x)) \)

AdaBoost Example: Face detection

- Training data:
  - \( 0.3 \text{ E1} - 0.6 \text{ E2} \)
  - \( 0.5 \text{ E1} - 0.5 \text{ E2} \)
  - \( 0.7 \text{ E1} - 0.1 \text{ E2} \)
  - \( 0.6 \text{ E1} - 0.4 \text{ E2} \)
  - \( 0.2 \text{ E1} + 0.4 \text{ E2} \)
  - \( 0.2 \text{ E1} - 0.3 \text{ E2} \)
  - \( 0.4 \text{ E1} - 0.9 \text{ E2} \)
  - \( 0.2 \text{ E1} + 0.5 \text{ E2} \)
  - \( \text{ Image: } a \cdot \text{ E1} + b \cdot \text{ E2} \rightarrow a = \text{ Image: E1/|Image|} \)
- Face detection with Eigen faces
- Step 0: Derive top 2 Eigen faces from training data
- Step 1: Represent all Eigen faces in development set with these Eigen faces
Train a weak classifier

The AdaBoost Algorithm

1. Initialize \( D_1(x) = \frac{1}{N} \)
2. For \( t = 1, \ldots, T \)
   a. Train a weak classifier \( h_t \) using distribution \( D_t \)
   b. Compute total error on training data
      \[ e_t = \sum_{i=1}^{N} D_t(i) \cdot 1/2 (1 - y_i \cdot h_t(x_i)) \]
   c. Set \( \alpha_t = \frac{1}{2} \ln \left( \frac{1 - e_t}{e_t} \right) \)
   d. For \( i = 1 \ldots N \)
      - set \( D_{t+1}(i) = D_t(i) \cdot \exp(-\alpha_t \cdot y_i \cdot h_t(x_i)) \)
   e. Normalize \( D_{t+1} \) to make it a distribution
3. The final classifier is
   \[ H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t \cdot h_t(x) \right) \]

The E1 “Stump”

Classifier based on E1:
If \( \text{sign} \cdot \text{wt}(E1) > \text{thresh} > 0 \)
   face = true

Sign = +1, error = 3/8
Sign = -1, error = 5/8

The E1 “Stump”

Classifier based on E1:
If \( \text{sign} \cdot \text{wt}(E1) > \text{thresh} > 0 \)
   face = true

Sign = +1, error = 2/8
Sign = -1, error = 6/8
The E1 “Stump”

Classifier based on E1: if (sign*wt(E1) > thresh) > 0) face = true

Sign = +1 or if (sign*wt(E1) > thresh) > 0)

Classifier based on E1:

Sign = +1 or if (sign*wt(E1) > thresh) > 0)

Classifier based on E1:

Sign = +1 or if (sign*wt(E1) > thresh) > 0)

Classifier based on E1:

Sign = +1 or if (sign*wt(E1) > thresh) > 0)

Classifier based on E1:

Sign = +1 or if (sign*wt(E1) > thresh) > 0)

Classifier based on E1:

Sign = +1 or if (sign*wt(E1) > thresh) > 0)

The E2 “Stump”

Classifier based on E2: if (sign*wt(E2) > thresh) > 0) face = true

Sign = +1 or if (sign*wt(E2) > thresh) > 0)

Classifier based on E2:

Sign = +1 or if (sign*wt(E2) > thresh) > 0)

Classifier based on E2:

Sign = +1 or if (sign*wt(E2) > thresh) > 0)

Classifier based on E2:

Sign = +1 or if (sign*wt(E2) > thresh) > 0)

Classifier based on E2:

Sign = +1 or if (sign*wt(E2) > thresh) > 0)

Classifier based on E2:

Sign = +1 or if (sign*wt(E2) > thresh) > 0)

Classifier based on E2:

Sign = +1 or if (sign*wt(E2) > thresh) > 0)

Classifier based on E2:

Sign = +1 or if (sign*wt(E2) > thresh) > 0)
The Best “Stump”

The Best Error

NOTE: THE ERROR IS THE SUM OF THE WEIGHTS OF MISCLASSIFIED INSTANCES

Computing Alpha

The Boosted Classifier Thus Far

The AdaBoost Algorithm

<table>
<thead>
<tr>
<th>T</th>
<th>0.5ln((1 – e_t)/e_t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5ln((1 – 0.2)/0.2) = 0.5 ln(4) = 1.39</td>
</tr>
<tr>
<td>2</td>
<td>0.5ln((1 – 0.6)/0.6) = 0.5 ln(2) = 0.5</td>
</tr>
<tr>
<td>3</td>
<td>0.5ln((1 – 0.3)/0.3) = 0.5 ln(3) = 0.89</td>
</tr>
<tr>
<td>4</td>
<td>0.5ln((1 – 0.2)/0.2) = 0.5 ln(4) = 1.39</td>
</tr>
<tr>
<td>5</td>
<td>0.5ln((1 – 0.5)/0.5) = 0.5 ln(2) = 0.5</td>
</tr>
<tr>
<td>6</td>
<td>0.5ln((1 – 0.7)/0.7) = 0.5 ln(3) = 0.89</td>
</tr>
<tr>
<td>7</td>
<td>0.5ln((1 – 0.2)/0.2) = 0.5 ln(4) = 1.39</td>
</tr>
<tr>
<td>8</td>
<td>0.5ln((1 – 0.5)/0.5) = 0.5 ln(2) = 0.5</td>
</tr>
<tr>
<td>9</td>
<td>0.5ln((1 – 0.7)/0.7) = 0.5 ln(3) = 0.89</td>
</tr>
</tbody>
</table>

\[ h_1(x) = \text{sign}(\alpha_1 h(X)) \]

\[ H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)) \]
**The AdaBoost Algorithm**

- Initialize \( D_1(x) = \frac{1}{N} \)
- For \( t = 1, \ldots, T \)
  - Train a weak classifier \( h_t \) using distribution \( D_t \)
  - Compute total error on training data
    - \( e_t = \text{Sum} (D_t(x) \cdot y \cdot h_t(x)) \)
    - Set \( \alpha_t = \frac{1}{2} \ln \left( \frac{1 - e_t}{e_t} \right) \)
  - For \( i = 1 \ldots N \)
    - set \( D_{t+1}(x) = D_t(x) \cdot \exp( - \alpha_t \cdot y \cdot h_t(x)) \)
  - Normalize \( D_{t+1} \) to make it a distribution
- The final classifier is
  \( H(x) = \text{sign} (\sum \alpha_t \cdot h_t(x)) \)

**The Best Error**

\[
D_{t+1}(x) = D_t(x) \cdot \exp(- \alpha_t \cdot y \cdot h_t(x))
\]

Multiply the correctly classified instances by 0.38
Multiply incorrectly classified instances by 2.63
### The Best E1 classifier

<table>
<thead>
<tr>
<th>F</th>
<th>E</th>
<th>H</th>
<th>A</th>
<th>G</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Sign</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Classifier based on E1:**

If \( \text{sign}(\text{wt}(E1) > \text{thresh}) > 0) \)

\( \text{face} = \text{true} \)

Sign = +1, error = 0.222

**Threshold:**

0.074

### The Best E2 classifier

<table>
<thead>
<tr>
<th>F</th>
<th>E</th>
<th>H</th>
<th>G</th>
<th>A</th>
<th>B</th>
<th>D</th>
<th>C</th>
<th>Sign</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Classifier based on E2:**

If \( \text{sign}(\text{wt}(E2) > \text{thresh}) > 0) \)

\( \text{face} = \text{true} \)

Sign = +1, error = 0.074

**Threshold:**

0.2

### The Best Classifier

<table>
<thead>
<tr>
<th>F</th>
<th>E</th>
<th>H</th>
<th>G</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Sign</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Classifier based on E1:**

If \( \text{wt}(E1) > 0.45 \)

\( \text{face} = \text{true} \)

Sign = +1, error = 0.074

**Threshold:**

0.48

### The Boosted Classifier Thus Far

<table>
<thead>
<tr>
<th>F</th>
<th>E</th>
<th>H</th>
<th>G</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Sign</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( h_1(X) = \text{sign}(0.97 \cdot h_1(X) + 1.26 \cdot h_2(X)) \)

\( h_2(X) = \text{sign}(0.97 \cdot h_1(X) + 1.26 \cdot h_2(X)) \)

\( h_3(X) = \text{sign}(0.97 \cdot h_1(X) + 1.26 \cdot h_2(X)) \)

\( h(X) = \text{sign}(0.97 \cdot h_1(X) + 1.26 \cdot h_2(X)) \)
Reweighting the Data

<table>
<thead>
<tr>
<th>ID</th>
<th>E1</th>
<th>E2</th>
<th>Class</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.3</td>
<td>-</td>
<td>+1</td>
<td>0.48*0.28</td>
</tr>
<tr>
<td>B</td>
<td>0.5</td>
<td>-</td>
<td>+1</td>
<td>0.074*0.28</td>
</tr>
<tr>
<td>C</td>
<td>0.7</td>
<td>-</td>
<td>+1</td>
<td>0.074*0.28</td>
</tr>
<tr>
<td>D</td>
<td>0.6</td>
<td>-</td>
<td>+1</td>
<td>0.074*0.28</td>
</tr>
<tr>
<td>E</td>
<td>0.2</td>
<td>0.4</td>
<td>-1</td>
<td>0.074*0.28</td>
</tr>
<tr>
<td>F</td>
<td>-</td>
<td>0.1</td>
<td>-1</td>
<td>0.074*0.28</td>
</tr>
<tr>
<td>G</td>
<td>0.4</td>
<td>0.9</td>
<td>-1</td>
<td>0.074*3.5</td>
</tr>
<tr>
<td>H</td>
<td>0.2</td>
<td>0.5</td>
<td>-1</td>
<td>0.074*0.28</td>
</tr>
</tbody>
</table>

Threshold: Sign = +1, error = 0.074

RENORMALIZE

NOTE: THE WEIGHT OF “G”, WHICH WAS MISCLASSIFIED BY THE SECOND CLASSIFIER, IS NOW SUDDENLY HIGH

- Initialize \( D_1(x) = \frac{1}{N} \)
- For \( t = 1, \ldots, T \)
  - Train a weak classifier \( h_t \) using distribution \( D_t \)
  - Compute total error on training data
    \[ e_t = \frac{1}{N} \sum \left( D_t(x_i) \cdot \frac{1}{2} (1 - y_i h_t(x_i)) \right) \]
  - Set \( \alpha_t = \frac{1}{2} \ln \left( \frac{1 - e_t}{e_t} \right) \)
  - For \( i = 1, \ldots, N \)
    - Set \( D_{t+1}(x_i) = D_t(x_i) \exp(-\alpha_t y_i h_t(x_i)) \)
  - Normalize \( D_{t+1} \) to make it a distribution

The final classifier is

\[ H(x) = \text{sign}(\sum \alpha_t h_t(x)) \]

The AdaBoost Algorithm

ADA Boost

- The final classifier is
  \[ H(x) = \text{sign}(\sum \alpha_t h_t(x)) \]
- Not all classifiers need to use \( E_1 \)
- It is often a good idea to keep adding classifiers even after the training accuracy reaches 100%

Face Detection: History

- Historically, tried complex models that tried to find eyes, nose, ears, etc…
Face Detection: History

Historically, tried complex models that tried to find eyes, nose, ears, etc…

Viola-Jones burst onto the scene with boosted cascade classifiers

The problem of face detection

1. Defining Features
   - Should we be searching for noses, eyes, eyebrows etc.?
     - Nice, but expensive
     - Or something simpler

2. Selecting Features
   - Of all the possible features we can think of, which ones make sense

3. Classification: Combining evidence
   - How does one combine the evidence from the different features?

Viola Jones Method

- Very simple features
- Modified AdaBoost algorithm
- Cascading classifiers

Features

“Integral” Features

- Each checkerboard has the following parameters
  - Length
  - Width
  - Type (specifies type of the pattern)
- Viola and Jones only used the above 4 patterns.

Dot products

- Compute dot products between the checkerboard patterns and the images

Feature = (sum of pixels in light region) – (sum of pixels in dark region)
**Integral images**

- Summed area tables

- For each pixel store the sum of ALL pixels to the left of and above it.

**Fast computation of Integral sums**

- Store pixel table for every pixel in the image
- The sum of all pixel values to the left of and above the pixel
- Let A, B, C, D, E, F be the pixel table values at the locations shown
  - Total pixel value of black area = \(D + A - B - C\)
  - Total pixel value of white area = \(F + C - D - E\)
  - Feature value = \((F + C - D - E) - (D + A - B - C)\)

**A Fast Way to Compute the Feature**

- Store pixel table for every pixel in the image
  - The sum of all pixel values to the left of and above the pixel
- Let A, B, C, D, E, F be the pixel table values at the locations shown
  - Total pixel value of black area = \(D + A - B - C\)
  - Total pixel value of white area = \(F + C - D - E\)
  - Feature value = \((F + C - D - E) - (D + A - B - C)\)

**How many features?**

- Each checker board of width \(P\) and height \(H\) can start at
  - \((0,0)\), \((0,1)\), \((0,2)\), \ldots \((0,N-P)\)
  - \((1,0)\), \((1,1)\), \((1,2)\), \ldots \((1,N-P)\)
  - \ldots
  - \((M-H,0)\), \((M-H,1)\), \((M-H,2)\), \ldots \((M-H,N-P)\)
- \((M-H) \times (N-P)\) possible starting locations
- Each is a unique checker feature
  - E.g. at one location it may measure the forehead, at another the chin

**No. of features**

- Each possible checkerboard gives us one feature
- A total of up to 180000 features derived from a 24x24 image!
- Every 24x24 image is now represented by a set of 180000 numbers
- This is the set of features we will use for classifying if it is a face or not!
AdaBoost

Correctly classified data: Multiply by $a_i$

Incorrectly classified data: Multiply by $1/a_i$

Pick best feature and threshold

AdaBoost: The weak learner

Threshold

Face detected!

Problems

- This only classifies images as face/non-face
- There can be multiple faces in an image
- Faces can be bigger than 24 x 24

Fixing the multiple face problem
Fixing the multiple face problem

But this is still too slow 😞
Not all classifiers are equally fast

All animals are equal. Some animals are more equal than others
- George Orwell's Animal Farm

Basic Idea

- Simple Classifier 1
- More complex Classifier 2

Modifying the classifier

- The normal AdaBoost trained classifier tries to optimize for accuracy
- We want it to errr on the side of never missing a face

Modifying the classifier

- False rejection: Failing to detect a face
- False detection: Detecting a face where there is none

Modifying the classifier

- False rejection: Failing to detect a face
- False detection: Detecting a face where there is none

Classifier:
- Standard boosted classifier: $H(x) = \text{sign}(\sum_i \omega_i h_i(x))$
- Modified classifier $H(x) = \text{sign}(\sum_i \omega_i h_i(x) + Y)$

Biasing the classifier towards False Detection

Retrain the system using AdaBoost with the system this way
Testing

Sample results using the Viola-Jones Detector

- Notice detection at multiple scales

38 classifiers in sequence with over 6000 features

15x times faster than best systems at that time !!!

Sometimes, the weak can beat the strong