

Chapter 8: Object Recognition and Detection

Andre L. C. Barczak

Computer Science
Massey University, Albany

Contents

Feature Extraction

Real-time Object Detection

Beyond Viola-Jones Method

Features

Information that describes the image, not necessarily in human terms

Feature Extraction

E.g., a mathematical method to extract numbers out of parts of the image. We could carry out edge detection, and get the best straight line, the coefficients would be the features.

Feature taxonomy:

- invariance (invariant to which transformations?)
- accuracy (robustness to noise?)
- response (real-time?)
- correlation (does it correlate to a certain class?)

Example with letters:

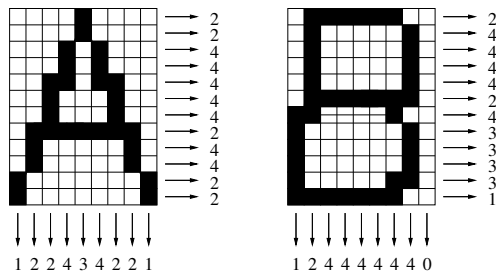


Figure 1: Simple feature extraction for binary images of Latin characters.

Invariant Features

moments

- Moment invariants - Hu 1962
- fast, but not very robust, prone to numerical instabilities
- E.g., used in OCR
- other moments: Zernike, Chebyshev, etc

Geometric Moment Invariants

moments

Given a digital image $i(x, y)$, the 2D moment of order $(p + q)$ is:

$$m_{pq} = \sum_x \sum_y x^p y^q i(x, y) \quad (1)$$

The values \bar{x} and \bar{y} are given by:

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \bar{y} = \frac{m_{01}}{m_{00}} \quad (2)$$

The central moment μ_{pq} is defined by:

$$\mu_{pq} = \sum_x \sum_y (\bar{x} - x)^p (\bar{y} - y)^q i(x, y) \quad (3)$$

Geometric Moment Invariants

invariant moments

And the normalised central moment η_{pq} is given by:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad (4)$$

Where $\gamma = \frac{p+q+2}{2}$

η_{pq} are invariant to scaling and translation. Hu came with a solution for moments to become also invariant to rotation.

Hu's invariants

invariant moments

$$\phi_1 = \eta_{20} + \eta_{02} \tag{5}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{6}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \tag{7}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \tag{8}$$

Hu's invariants

invariant moments

$$\begin{aligned}\phi_5 = & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3((\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (9)\end{aligned}$$

$$\begin{aligned}\phi_6 = & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - ((\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\ & (10)\end{aligned}$$

$$\begin{aligned}\phi_7 = & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3((\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (11)\end{aligned}$$

Hu's invariants

invariant moments

- Hu's invariants: up to 3rd order.
- Flusser (2000) recommended: ϕ_2 and ϕ_3 should not be used.
- Complex numbers solution: independent and complete moment invariants set.
- Published in 2000, but most libraries still use 7 Hu's invariants (including OpenCV...).

Real-time Object Detection

The revolution of Viola and Jones

- Haar-like features (with SATs).
- The training method: *Adaboost*.
- Cascades of classifiers.

Motivation

Introduction

For Detection and Recognition tasks, Feature Extraction Methods should be:

1. Invariant to geometric transformations
2. Invariant to illumination changes
3. Stable (numeric stability)
4. Fast to compute

What kind of solution

- Elegant solution?
- Safe solution?
- Fast solution?

Face Detection

Introduction

Digital cameras face detection based on *Viola-Jones method*.

How does it work?

- Haar-like features
- Cascade classifier (AdaBoost)

Four requirements?

1. Invariant to geometric transformations: **scaling**
2. Invariant to illumination changes: **contrast stretched features**
3. Stable (numeric stability): **stable, but inaccurate**
4. Fast to compute: **use SATs and cascades to speed up**

How your camera works

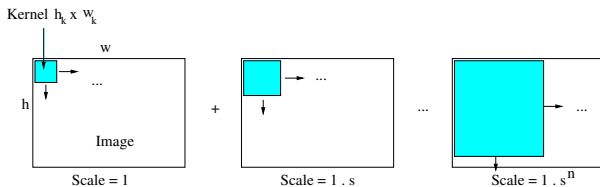


Figure 2: A scalable kernel.

Each sub-window is compared against a classifier.
The classifier uses Haar-like features.

How your camera works

This is what Haar-like features look like:

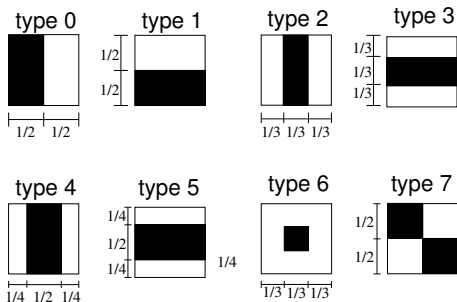


Figure 3: Examples of Haar-like Features.

Adding and subtracting pixels at different scales is really slow...

How your camera works

And this is how to compute Haar-like features really fast:

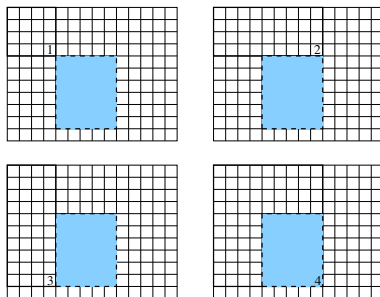


Figure 4: Summed-area Tables.

Sum of the blue area:

- $SAT(x_4, y_4) - SAT(x_3, y_3) - SAT(x_2, y_2) + SAT(x_1, y_1)$
- These Haar-like features only need 6, 8 or 9 lookups

The classifier: AdaBoost

Then, each sub-window runs through a binary classifier.

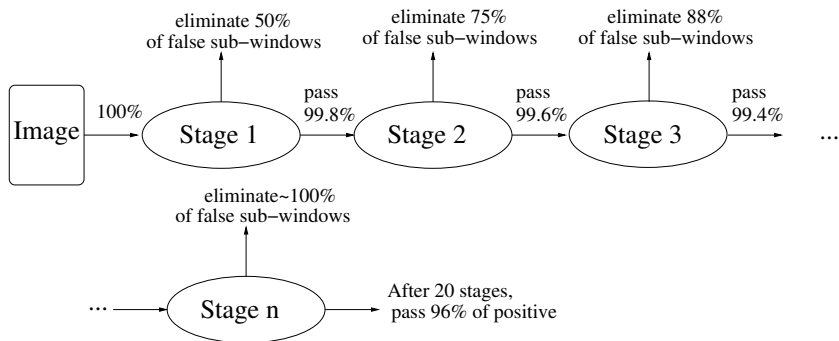


Figure 5: A cascade of classifiers filters negative samples.

AdaBoost

- Discrete AdaBoost
- Real AdaBoost
- LogitBoost
- Gentle AdaBoost
- ... and many other variations

Let's try to understand the principles behind AdaBoost

AdaBoost based on Binary Classifiers

- Weak Classifiers: slightly better than 50 %
- **Question:** how can we combine them to make a strong classifiers?
- Answer: AdaBoost, short for Adaptive Boosting
- AdaBoost uses a linear combination of weak classifiers
- Top 10 data mining algorithms (2007)
- Gödel prize (2003)

AdaBoost

Require: The input is a set of training examples $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where x_i is a feature $\in X$ and y_i is a class $\in \{-1, +1\}$, and a number of rounds T . It outputs a linear combination of weak classifiers $h_t(x_i)$, each a factor α_t .

Initialise weights for each element $D_1(i) = 1/(n)$ where n is the number of elements

for $t = 1, 2 \dots T$: **do**

Train a weak classifier h_t using the weights in D_t so that $h_t \in \{-1, +1\}$

Compute the error associated with the weak classifier:

$$error_t = \sum_i (D_t(i) h_t(x_i) y_i)$$

$$\text{Compute } \alpha_t = 0.5 \ln\left(\frac{1 - error_t}{error_t}\right)$$

Update the weights $D_{t+1}(i)$ such as D_{t+1} is a new distribution:

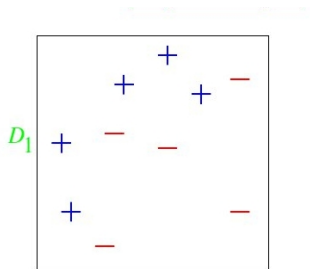
$$D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

$$\text{normalise the weights } D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_i D_{t+1}(i)}$$

end for

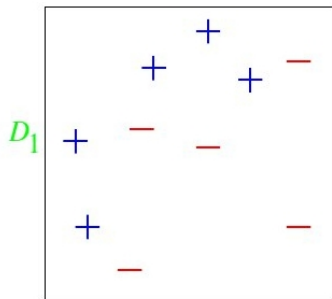
After T rounds the final classifier is: $H(x) = \text{sign}(\sum_t \alpha_t h_t(x))$

Weak Classifiers



Suppose we want to classify the following dataset, with distribution D_1 using only two features

Weak Classifiers



We start with the same weights to every sample $w_n = \frac{1}{N}$
(in this case, $w_n = 0.1$)

Weak Classifiers

We can exhaustively test all possible thresholds:

if ($F_1 \geq \textit{threshold}$) then $h(1) = 1$ (positive or +)

if ($F_1 < \textit{threshold}$) then $h(1) = 0$ (negative or -)

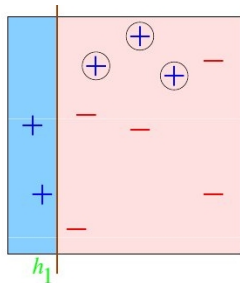
After testing several options,

- h_1 hypothesis depends on the threshold
- error e_1 depends on the distribution
- α is computed as a function of e_1

ϵ_1 and α_1

$\epsilon_1 = 0.3$ (3 positives misclassified, each with $w_n = 0.1$)

$$\alpha_1 = 0.5 \ln\left(\frac{1-0.3}{0.3}\right) = 0.4236$$

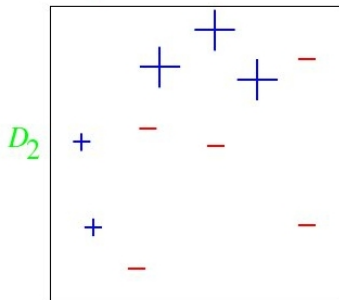


$$\epsilon_1 = 0.30$$

$$\alpha_1 = 0.42$$

Round 2

According to the algorithm's rules, we re-weight all samples, creating a new distribution D_2



Round 2

To compute e_2 and α_2 , we need all the individual weights w_n .
Updating the weights:

$$w_{misc} = 0.1 e^{0.42} = 0.152$$

$$w_{corr} = 0.1 e^{-0.42} = 0.066$$

But $\sum w_n \neq 1 \dots (\sum w_n = 0.9165)$

Normalising the weights:

$$w_{misc} = \frac{0.152}{0.9165} = 0.17$$

$$w_{corr} = \frac{0.066}{0.9165} = 0.07$$

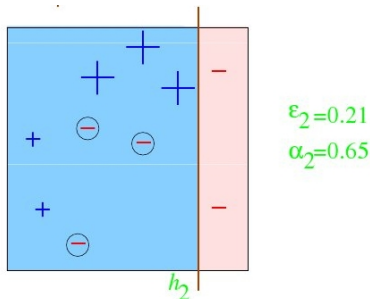
Round 2

$$w_{misc} = \frac{0.152}{0.9165} = 0.17 \quad w_{corr} = \frac{0.066}{0.9165} = 0.07$$

Find a threshold h_2 with 3 misclassified samples:

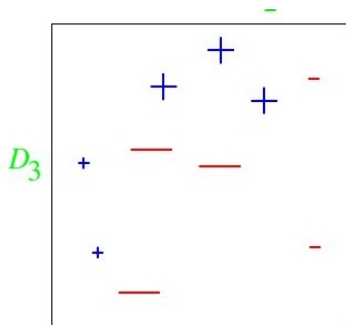
$$e_2 = 3 : 0.07 = 0.21$$

$$\alpha_2 = 0.5 \ln\left(\frac{1-0.21}{0.21}\right) = 0.65$$



Round 3

A new distribution D_3 is used. Now there are three different weights for different samples...



Round 3

Updating the weights:

$$w_{misc} = 0.07 e^{0.65} = 0.14 \text{ (3 samples)}$$

$$w_{corr1} = 0.07 e^{-0.65} = 0.04 \text{ (4 samples)}$$

$$w_{corr2} = 0.17 e^{-0.65} = 0.09 \text{ (3 samples)}$$

Normalising the weights:

$$w_{misc} = 0.17$$

$$w_{corr1} = 0.045$$

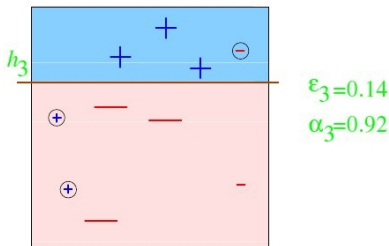
$$w_{corr2} = 0.11$$

Round 3

Compute e_3 and α_3 : In round 3 we have to find a threshold h_3 that give the smallest error.

$$e_3 = 0.045 \cdot 3 = 0.14$$

$$\alpha_3 = 0.5 \ln\left(\frac{1-0.14}{0.14}\right) = 0.92$$



Strong Classifier

Combining the three weak classifiers using their answers with the α values:

$$= \text{sign} \left(0.42 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{blue} & \text{blue} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} \right)$$

$$= \begin{array}{|c|c|c|c|} \hline \text{blue} & + & + & - \\ \hline & + & + & \\ \hline + & - & - & - \\ \hline + & - & & \\ \hline \end{array}$$

Strong Classifier

8 possible combinations (as we have 3 weak classifiers):

$$+ h_1 + h_2 + h_3 \qquad + h_1 + h_2 - h_3$$

$$+ h_1 - h_2 + h_3 \qquad + h_1 - h_2 - h_3$$

$$- h_1 + h_2 + h_3 \qquad - h_1 + h_2 - h_3$$

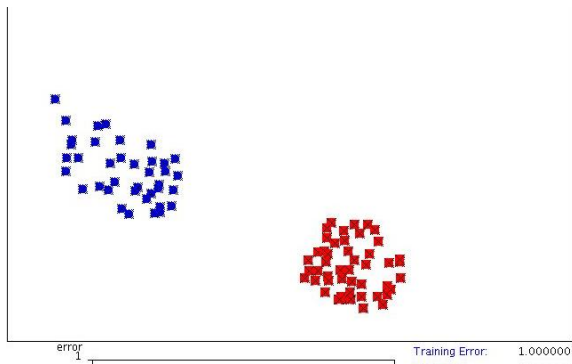
$$- h_1 - h_2 + h_3 \qquad - h_1 - h_2 - h_3$$

This divides the hyperspace (2 dimensions) into 6 regions

AdaBoost example 1

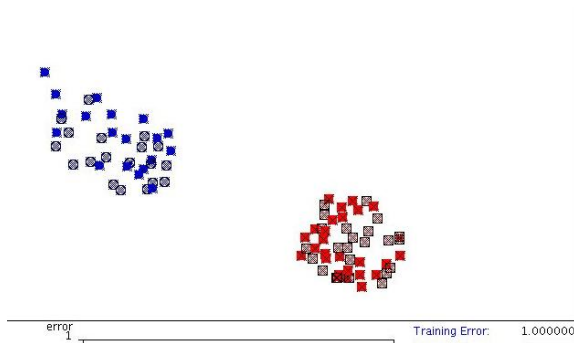
Suppose we have a set of samples.

Blue is positive, red is negative. Note that the sets are easily separable with a straight line.



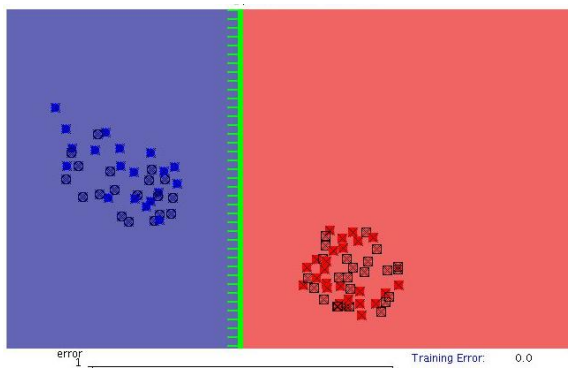
AdaBoost example 1

We need to split the samples into **training** and **test** sets.



AdaBoost example 1

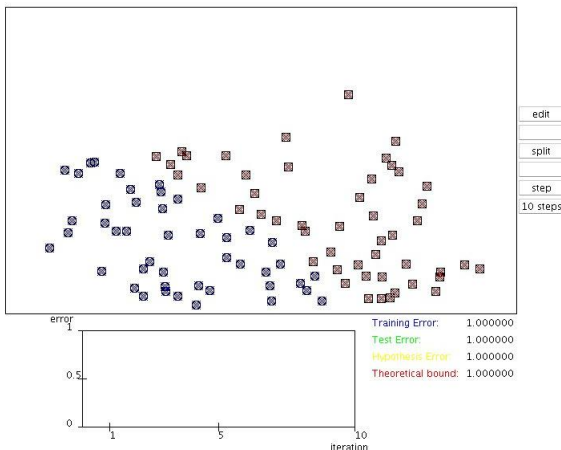
Finding a classifier is easy, a simple threshold will do.
The blue region covers all the positives, while the pink region covers all the negatives.



AdaBoost example 2

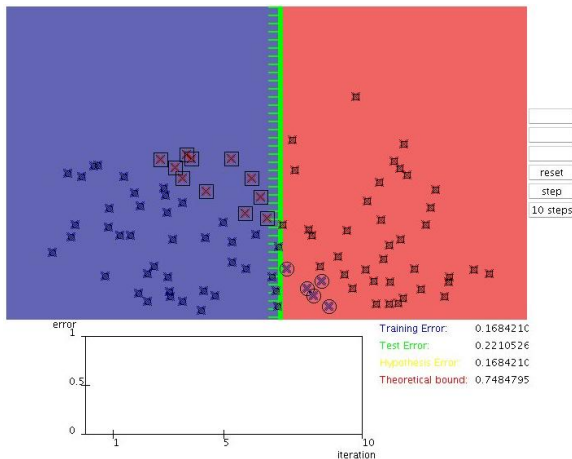
Suppose now that we have a set, still easily separable with a straight line, but not with a single threshold.

This set is also split into **training** and **test** sets.



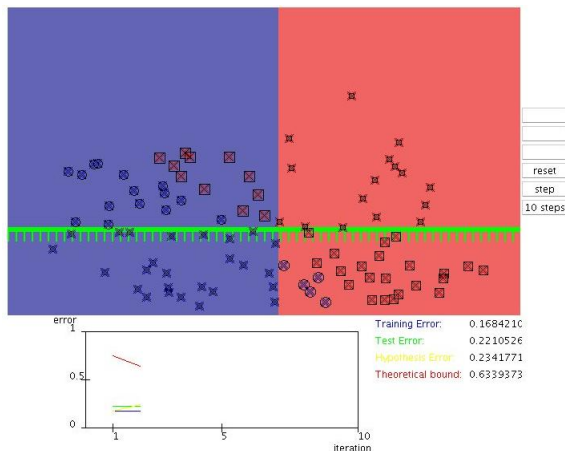
AdaBoost example 2

The first weak classifier (a threshold) gets a training error of 0.1684 and a test error of 0.2210. The mistakes are shown as larger samples on both positives and negatives.



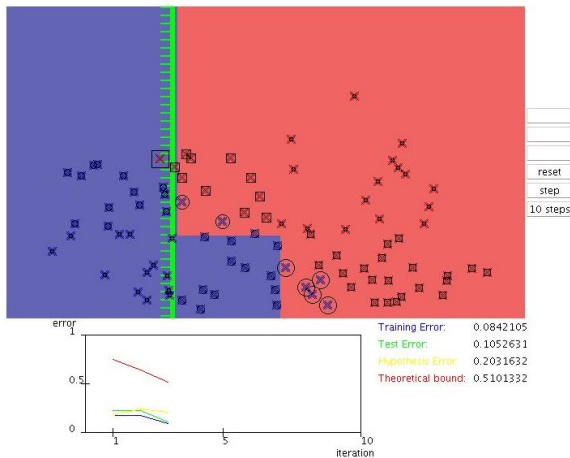
AdaBoost example 2

The second weak classifier is added, but that does not change the errors yet.



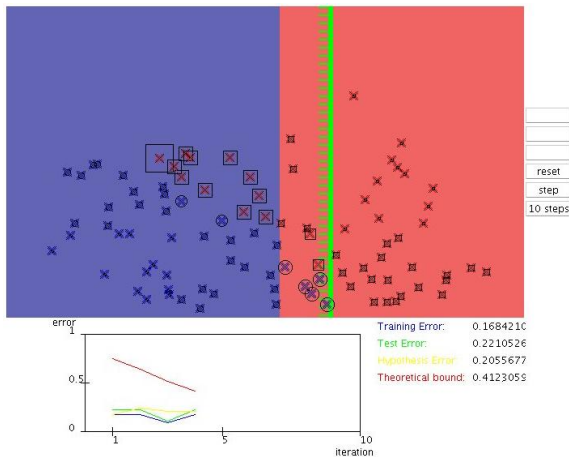
AdaBoost example 2

The third weak classifier...



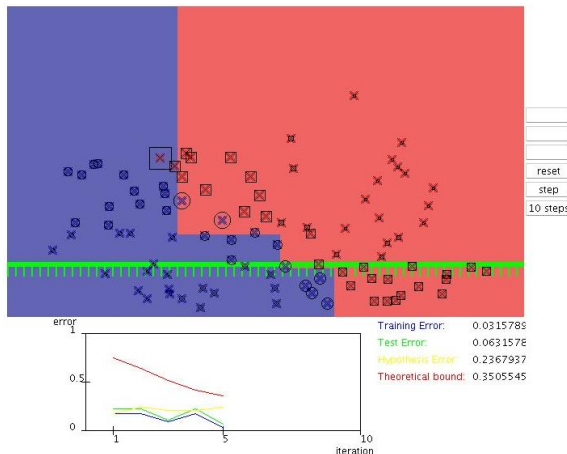
AdaBoost example 2

The fourth weak classifier...



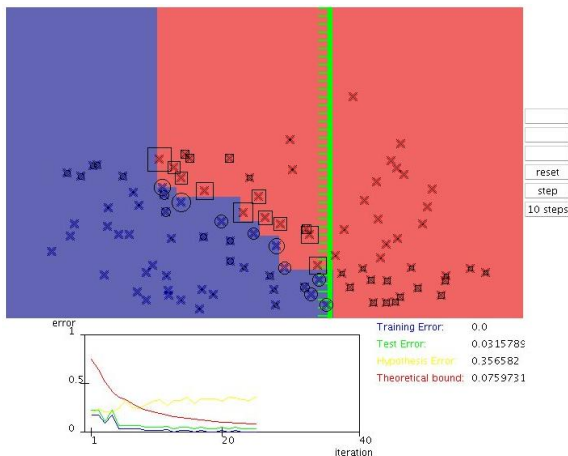
AdaBoost example 2

The fifth weak classifier...Note that the training and test errors are getting smaller, and that the region is being divided according to the sample's location.



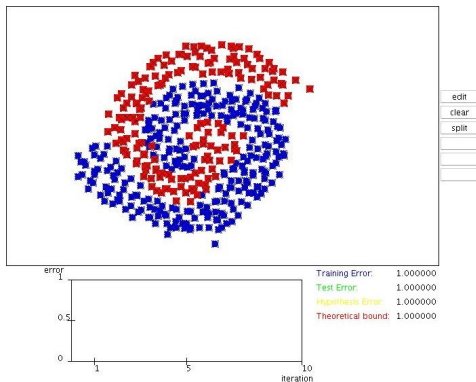
AdaBoost example 2

After 25 weak classifiers, the training error reached 0.0. The test error is low, but not zero.



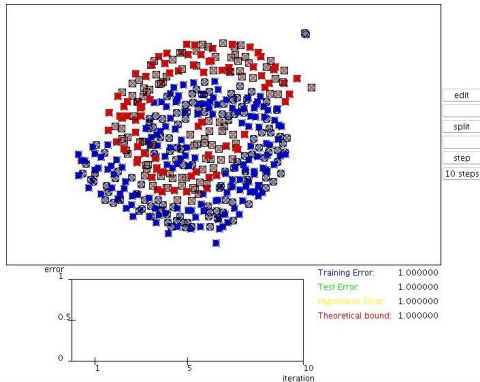
AdaBoost example 3

This set is not separable by a straight line. Let us see how AdaBoost resolves this case.



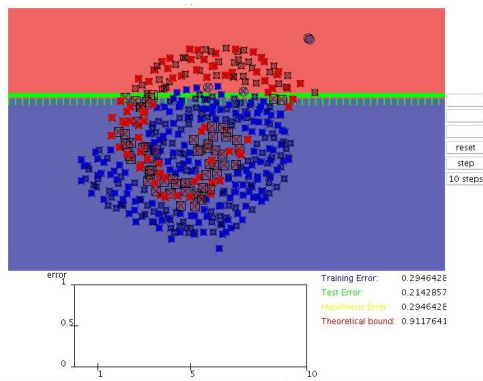
AdaBoost example 3

Split into training and test sets.



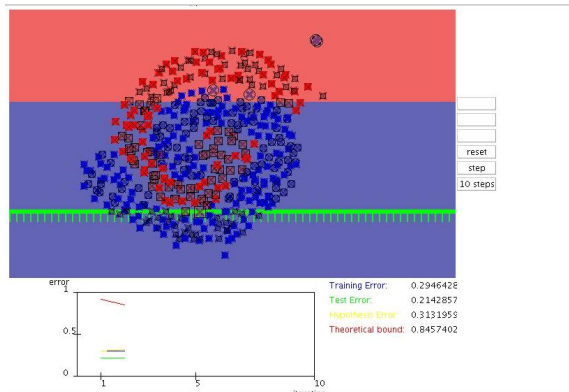
AdaBoost example 3

First weak classifier...



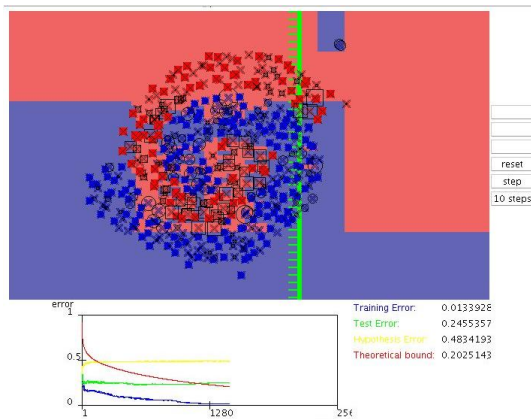
AdaBoost example 3

Second weak classifier...

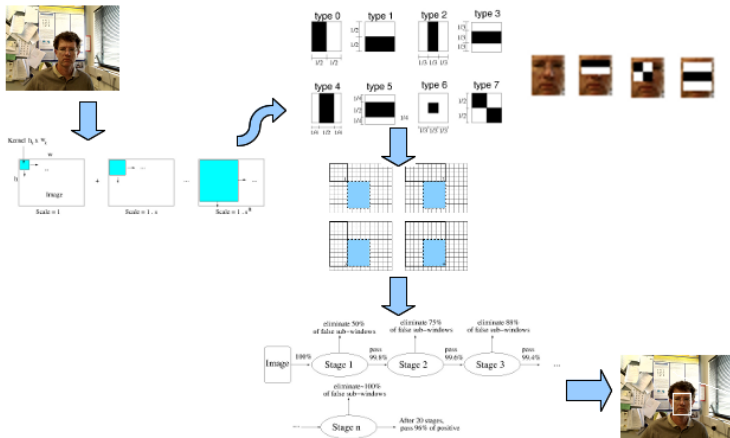


AdaBoost example 3

After many weak classifiers, the training error gets to 0.013, but the test set still is 0.245. If there are outliers, the training set can get to zero, but the test set cannot be improved.



How your camera works



In summary, this is how digital cameras find human faces...

Why does it sometimes fail?

- Rotated faces
- Illuminations that were not trained
- Translation factor
- Scaling factor

Rotated Objects

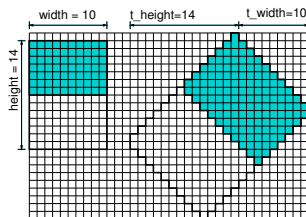


Figure 6: Converting OpenCV face detection.

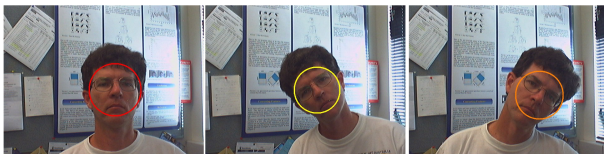
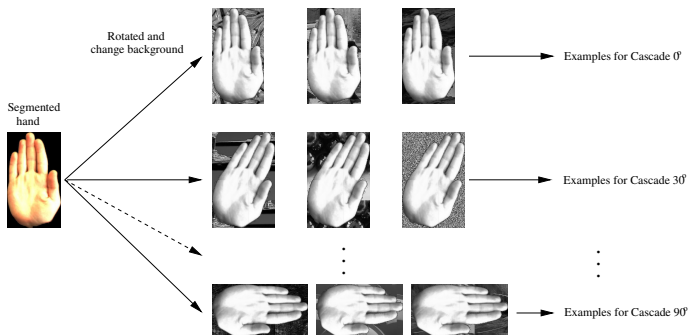


Figure 7: Converting OpenCV face detection.

Some experiments

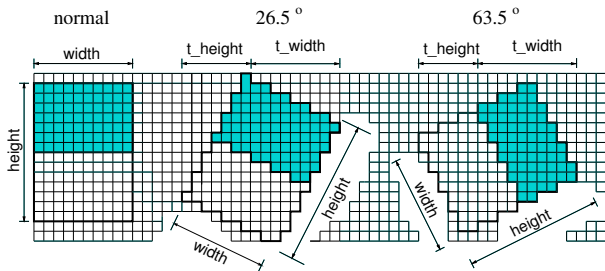
Haar-like features are non-invariant to rotation



Training

- One classifier per angle per gesture.
- Looong time to train...

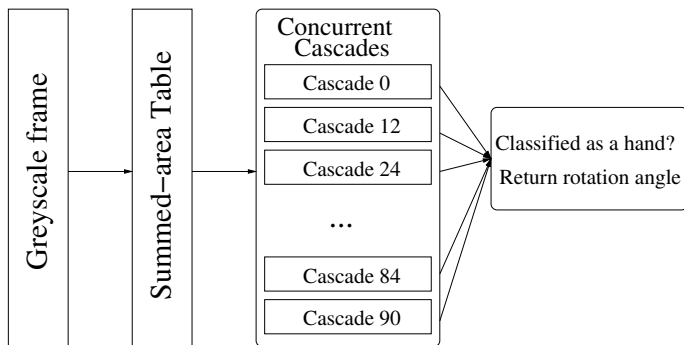
Haar-like Features with other angles



Performance consequences:

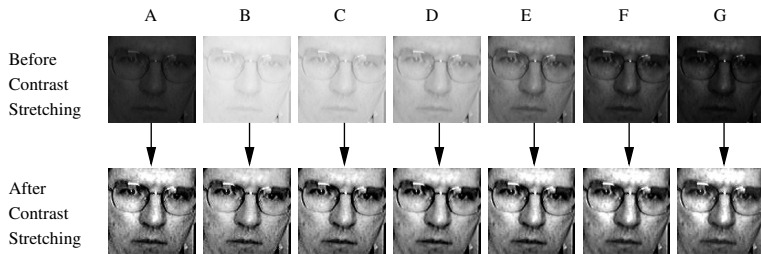
- Extra SATs
- Difficult to implement, e.g., needs to guarantee alignments
- Needs a separate classifier per angle

Detection



Single SAT for all cascades, more training
or multiple SATs with less training

Illumination Changes



Local Contrast Stretching:

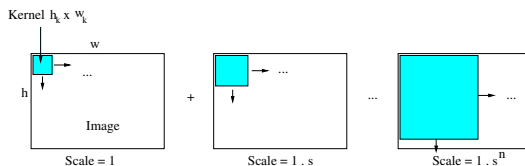
$$\bar{i}(x, y) = \frac{255(i(x, y) - \mu + c\sigma)}{2c\sigma} \quad (12)$$

The μ is found using the same upright SAT.

The σ can be found using an extra SAT of squares of pixels.

Translation and Scaling Factors

More CPU needed for more sub-windows...






Translation factor t and Scaling factor s :

- Number of sub-windows:

$$n_{sub} = \sum_{i=1}^n \frac{(w - w_k \cdot s^i)(h - h_k \cdot s^i)}{t^2} \quad (13)$$

Bibliography

(partial)

-  A. L. C. Barczak and M. J. Johnson, “A new rapid feature extraction method for computer vision based on moments,” in *International Conference in Image and Vision Computing NZ (IVCNZ 2006)*, (Auckland, NZ), pp. 395–400, November 2006.
-  Y. Freund and R. E. Schapire, “A short introduction to boosting,” *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.
-  P. Viola and M. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, pp. 137–154, May 2004.