Advanced Machine Learning

Bogdan Alexe,

bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2019-2020

Final exam date

- I fixed the date for the final exam = date for the deadline of assignment 2 (Sunday, 21st of June).

- **Take into account for the final grade only the grades for the 2 assignments (no final exam) – so each of the 2 assignments will worth 5 points (instead of 3.5 points)**

Recap - Weak learnability

Definition (γ-Weak-Learnability)
A learning algorithm, A, is a γ-weak-learner for a class $\mathcal{H}$ if there exists a function $m_{\mathcal{H}}:(0,1)\rightarrow N$ such that:

- for every $\delta > 0$ (*confidence*)
- for every labeling $f \in \mathcal{H}, f: \mathcal{X} \rightarrow \{-1,+1\}$ (*realizability case*)
- for every distribution $\mathcal{D}$ over $\mathcal{X}$

when we run the learning algorithm A on a training set, consisting of $m \geq m_{\mathcal{H}}(\delta)$ examples sampled i.i.d. from $\mathcal{D}$ and labeled by *f,* the algorithm A returns a hypothesis *h* (*h* might not be from $\mathcal{H}$ - improper learning) such that, with probability at least $1-\delta$ (over the choice of examples), $L_{\mathcal{D},f}(h) \leq 1/2 - \gamma$.

A hypothesis class $\mathcal{H}$ is γ-weak-learnable if there exists a γ-weak-learner for that class.

Recap - Weak vs Strong learnability

Strong learnability implies the ability to find an arbitrarily good classifier (with error rate at most ε for an arbitrarily small $\varepsilon > 0$).

In weak learnability, however, we only need to output a hypothesis whose error rate is at most $1/2-\gamma$, namely, whose error rate is slightly better than what a random labeling would give us.

The hope is that it may be easier to come up with ***efficient*** weak learners than with efficient (strong) PAC learners. If we have access to an ***efficient*** weak learner, can we use it to build an ***efficient*** strong learner?

We will see that strong learnability $\Leftrightarrow$ weak learnability

$\Rightarrow$ easy to show, based on the definition

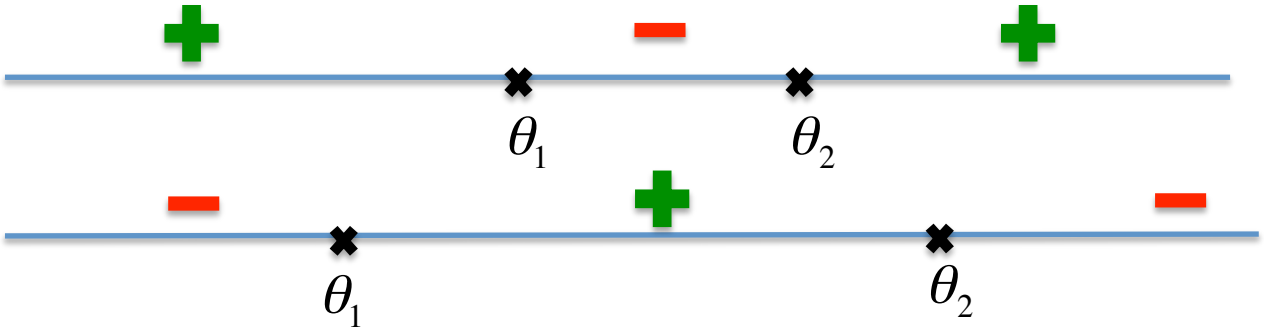
$\Leftarrow$ use boosting (improper learning algorithm) that combines weak learners to obtain a strong learner.

If the learning problem is hard, boosting cannot help as we can't find efficient weak learners.

Recap - Weak learnability - example

Let $\mathcal{X}$ = R, $\mathcal{H}$ is the class of 3-piece classifiers (signed intervals):

$$H = \{h_{\theta_1,\theta_2,b} \,|\, \theta_1,\theta_2 \in R, \theta_1 < \theta_2, b \in \{-1,+1\}\} \quad h_{\theta_1,\theta_2,b}(x) = \begin{cases} +b, \text{ if } x < \theta_1 \text{ or } x > \theta_2 \\ -b, \text{ if } \theta_1 \le x \le \theta_2 \end{cases}$$

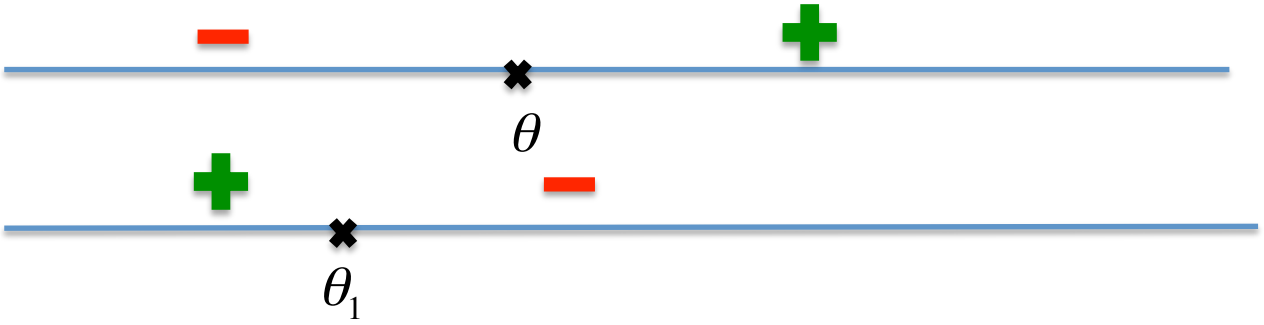


b = 1

b = -1

Consider $\mathcal{B}$ the class of Decision Stumps = class of 1-node decision trees

$\mathcal{B} = \{h_{\theta,b}: \mathbf{R} \to \{-1,1\}, h_{\theta,b}(x) = \text{sign}(x - \theta) \times b, \theta \in \mathbf{R}, b \in \{-1,+1\}\}$.



b = 1

b = -1

$\text{ERM}_{\mathcal{B}}$ is a γ-weak learner for $\mathcal{H}$, for $\gamma < 1/6$.

Efficient implementation of ERM for Decision Stumps

In practice, we use the following base hypothesis class of decision stumps over $\mathbf{R}^d$ for weak learners:

$\mathcal{H}_{DS}^d = \{h_{i,\theta,b}: \mathbf{R}^d \rightarrow \{-1,1\}, h_{i,\theta,b}(\mathbf{x}) = \text{sign}(\theta - x_i) \times b, 1 \leq i \leq d, \theta \in \mathbf{R}, b \in \{-1,+1\}\}$

-pick a coordinate i (from 1 to d), project the input $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ on the i-th coordinate and obtain x_i, if $x_i \leq$ threshold θ label the example with b, else with –b

Efficient implementation of the ERM rule for the class $\mathcal{H}_{DS}^d$ in $O(d \times m \times \log_2 m)$.

Today's lecture: Overview

- Boosting
- Face detection with AdaBoost

A formal description of boosting

Given:

- training set S = (($\mathbf{x_1}$, y_1), …, ($\mathbf{x_m}$, y_m)) of size m, $y_i \in \{-1, +1\}$
- weak learner
- T – number of rounds

For t = 1,...,T (number of rounds):

- construct distribution $\mathbf{D}^{(t)}$ on {1,..., m}
- find weak classifier ("rule of thumb") $h_t : \mathcal{X} \rightarrow \{-1,+1\}$ with error ε_t on $\mathbf{D}^{(t)}$:

$$\varepsilon_t = \Pr_{i \sim D^{(t)}}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D^{(t)}(i) \times 1_{[h_t(x_i) \neq y_i]}$$

- output final/combined classifier h_{final} using all the learned weak classifiers h_t

Each round involves building the distribution $\mathbf{D}^{(t)}$ as well as a single call to the weak learner. Therefore, if the weak learner can be implemented efficiently (as happens in the case of ERM with respect to decision stumps – improper learning) then the total training process will be efficient. Different variants of boosting comes from constructing distribution $\mathbf{D}^{(t)}$ + obtaining the final classifier h_{final}

First boosting algorithms

- [Schapire ‘89]:
  - first provable boosting algorithm
- [Freund ‘90]:
  - “optimal” algorithm that “boosts by majority”
- [Drucker, Schapire & Simard ‘92]:
  - first experiments with boosting
  - limited by practical drawbacks
- [Freund & Schapire ‘95]:
  - introduced “AdaBoost” algorithm
  - strong practical advantages over previous boosting algorithms

AdaBoost

- construct distribution $\mathbf{D}^{(t)}$ on {1,..., m}:
  - $\mathbf{D}^{(1)}(i) = 1/m$
  - given $\mathbf{D}^{(t)}$ and h_t: $D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}}{Z_{t+1}}$

where Z_{t+1} normalization factor ($\mathbf{D}^{(t+1)}$ is a distribution): $Z_{t+1} = \sum_{i=1}^{m} D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}$

w_t is a weight: $w_t = \frac{1}{2} \ln(\frac{1}{\varepsilon_t} - 1) > 0$ as the error $\varepsilon_t < 0.5$

ε_t is the error of h_t on $\mathbf{D}^{(t)}$: $\varepsilon_t = \Pr_{i \sim D^{(t)}}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D^{(t)}(i) \times 1_{[h_t(x_i) \neq y_i]}$

If example $\mathbf{x}_i$ is correctly classified then $h_t(\mathbf{x}_i) = y_i$ so at the next iteration t+1 its importance (probability distribution) will be decreased to:

$$D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{-w_t}}{Z_{t+1}} = \frac{D^{(t)}(i) \times e^{-\frac{1}{2}\ln(\frac{1}{\varepsilon_t}-1)}}{Z_{t+1}} = \frac{D^{(t)}(i) \times (\frac{1}{\varepsilon_t} - 1)^{-\frac{1}{2}}}{Z_{t+1}} = \frac{D^{(t)}(i) \times \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}}{Z_{t+1}}$$

Yoav Freund and Robert E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting**. Journal of Computer and System Sciences, 55(1):119–139, August 1997

AdaBoost

- construct distribution $\mathbf{D}^{(t)}$ on {1,..., m}:
  - $\mathbf{D}^{(1)}(i) = 1/m$
  - given $\mathbf{D}^{(t)}$ and h_t: $D^{(t+1)}(i) = \dfrac{D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}}{Z_{t+1}}$

where Z_{t+1} normalization factor ($\mathbf{D}^{(t+1)}$ is a distribution): $Z_{t+1} = \sum_{i=1}^{n} D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}$

w_t is a weight: $w_t = \dfrac{1}{2} \ln(\dfrac{1}{\varepsilon_t} - 1) > 0$ as the error $\varepsilon_t < 0.5$

ε_t is the error of h_t on $\mathbf{D}^{(t)}$: $\varepsilon_t = \Pr_{i \sim D^{(t)}}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D^{(t)}(i) \times 1_{[h_t(x_i) \neq y_i]}$

If example x_i is misclassified then $h_t(x_i) \neq y_i$ so at the next iteration t+1 its importance (probability distribution) will be increased to:

$$D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{w_t}}{Z_{t+1}} = \frac{D^{(t)}(i) \times e^{\frac{1}{2}\ln(\frac{1}{\varepsilon_t}-1)}}{Z_{t+1}} = \frac{D^{(t)}(i) \times (\frac{1}{\varepsilon_t} - 1)^{\frac{1}{2}}}{Z_{t+1}} = \frac{D^{(t)}(i) \times \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}}{Z_{t+1}}$$

AdaBoost

- construct distribution $\mathbf{D}^{(t)}$ on {1,..., m}:
  - $\mathbf{D}^{(1)}(i) = 1/m$
  - given $\mathbf{D}^{(t)}$ and h_t: $D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}}{Z_{t+1}}$

where Z_{t+1} normalization factor ($\mathbf{D}^{(t+1)}$ is a distribution): $Z_{t+1} = \sum_{i=1}^{n} D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}$

w_t is a weight: $w_t = \frac{1}{2} \ln(\frac{1}{\varepsilon_t} - 1) > 0$ as the error $\varepsilon_t < 0.5$

ε_t is the error of h_t on $\mathbf{D}^{(t)}$: $\varepsilon_t = \Pr_{i \sim D^{(t)}}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D^{(t)}(i) \times 1_{[h_t(x_i) \neq y_i]}$

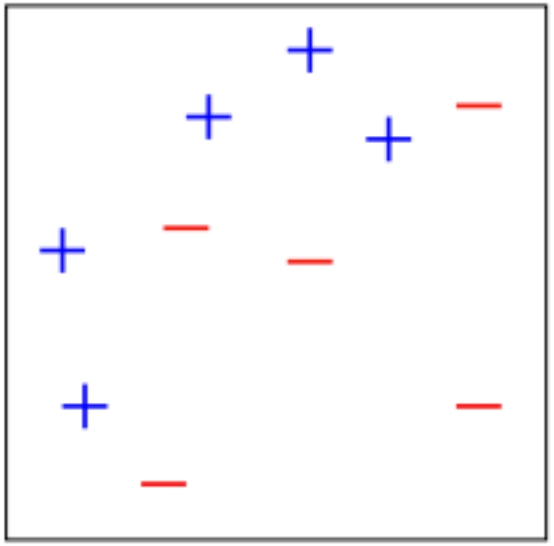
If example $\mathbf{x}_i$ is correctly classified then $h_t(\mathbf{x}_i) = y_i$ so at the next iteration t+1 its importance (probability distribution) will be decreased to $D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{-w_t}}{Z_{t+1}}$

If example x_i is misclassified then $h_t(x_i) \neq y_i$ so at the next iteration t+1 its importance (probability distribution) will be increased to $D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{w_t}}{Z_{t+1}}$

- output final/combined classifier h_{final}: $h_{final}(x) = sign(\sum_{t=1}^{T} w_t h_t(x))$

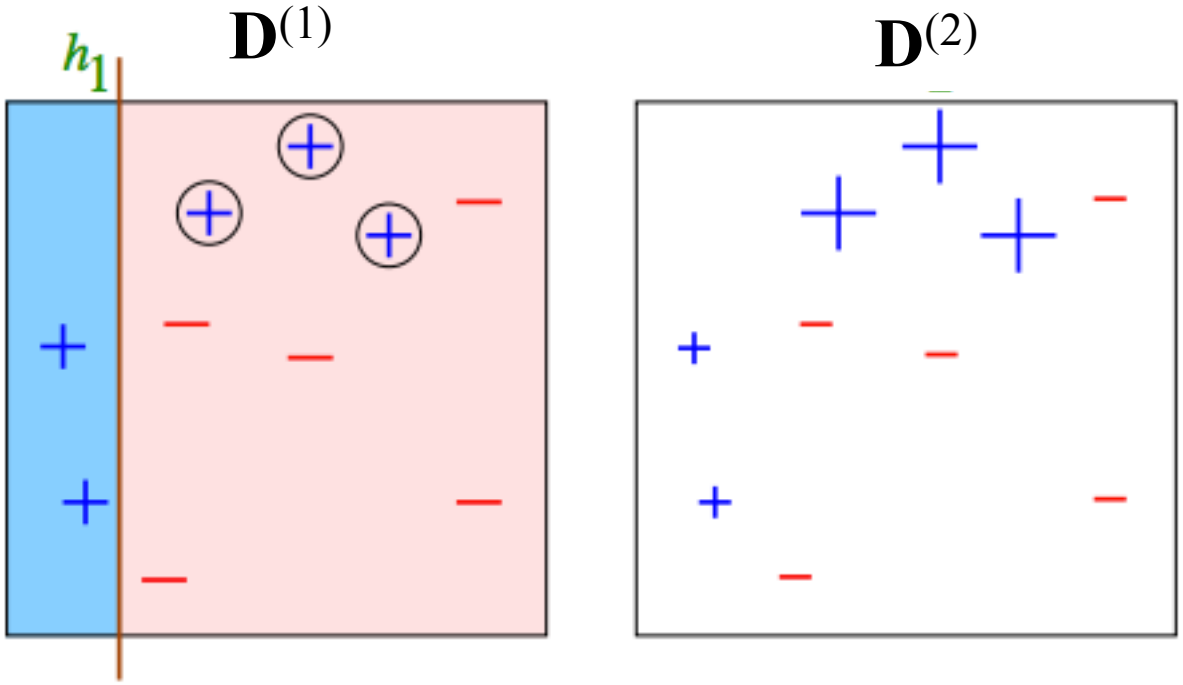
Toy example

$\mathbf{D}^{(1)}$



Weak classifiers = vertical or horizontal half- planes = hypothesis from $\mathcal{H}_{DS}{}^{2}$

Toy example – Round 1

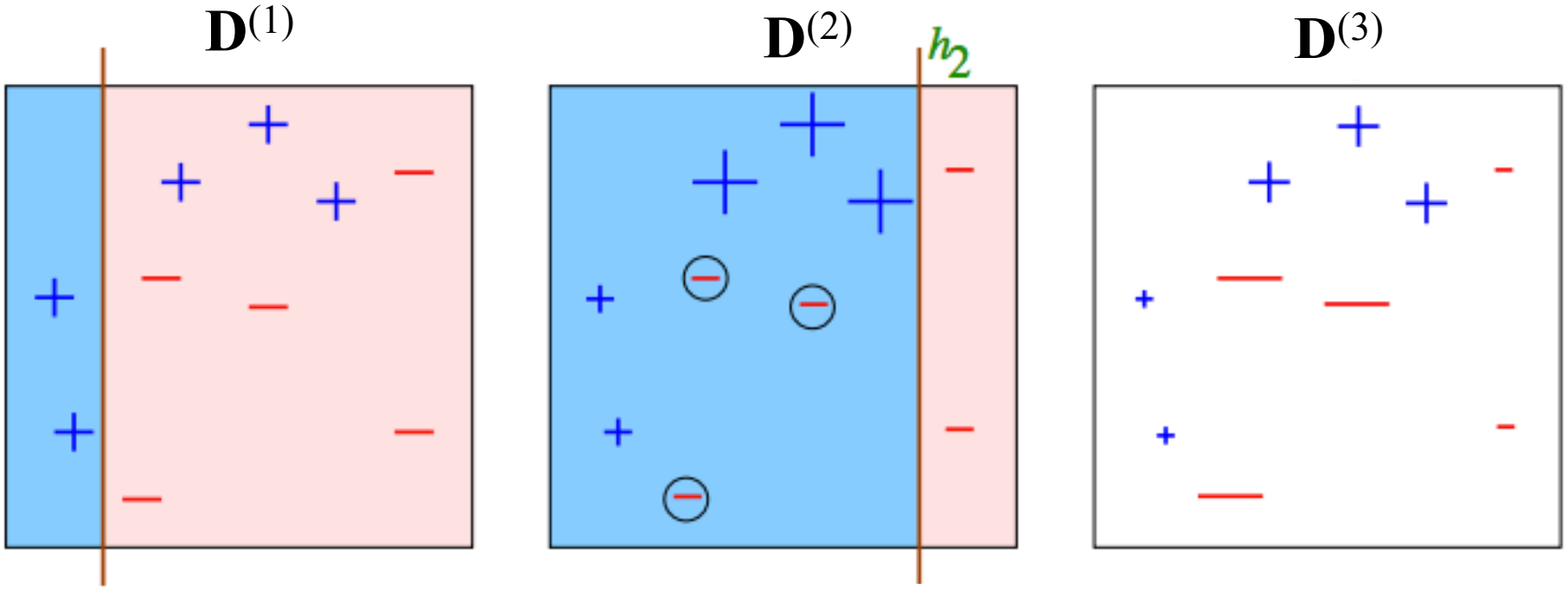


$\varepsilon_1 = 0.30$

$w_1 = 0.42$

Weak classifiers = vertical or horizontal half- planes = hypothesis from $\mathcal{H}_{DS}{}^2$

Toy example – Round 2

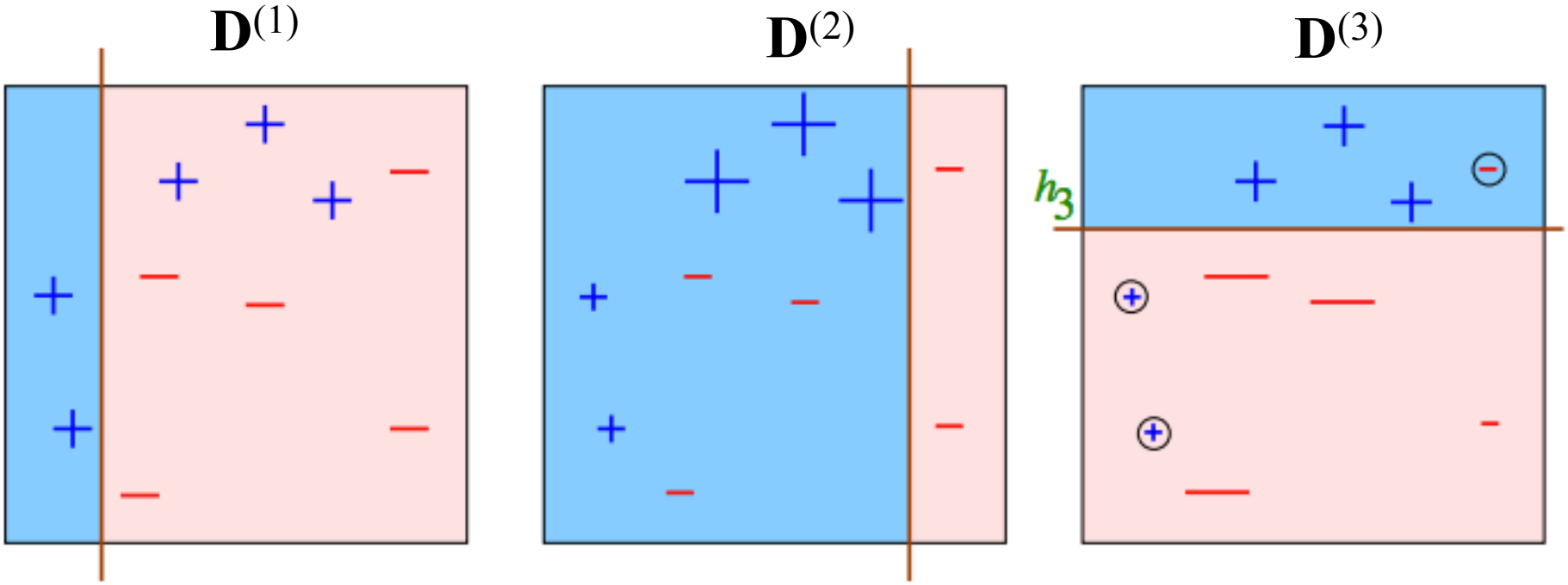


$\varepsilon_2 = 0.21$

$w_2 = 0.65$

Weak classifiers = vertical or horizontal half- planes = hypothesis from $\mathcal{H}_{DS}{}^2$

Toy example – Round 3

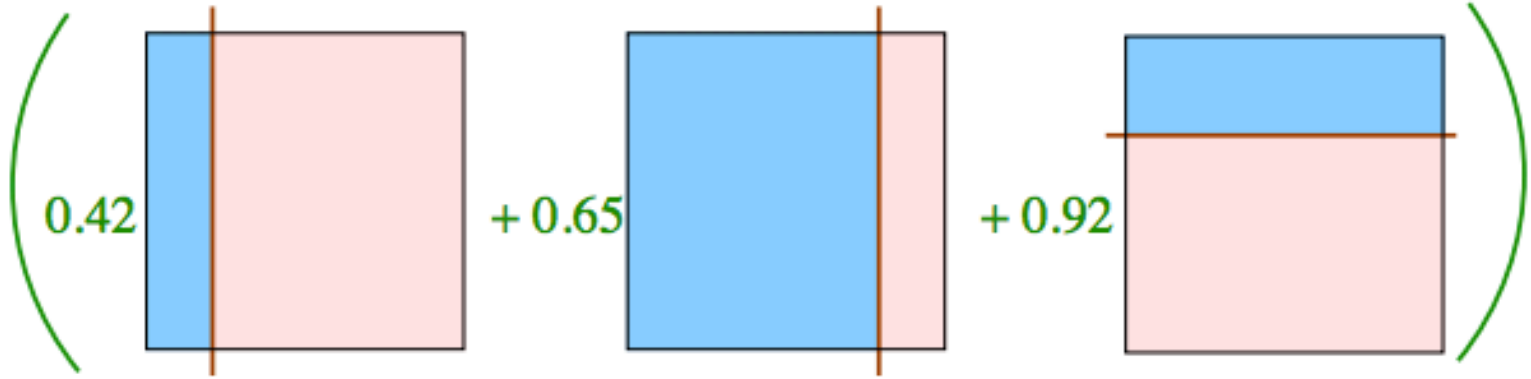


$\varepsilon_3 = 0.14$
$w_3 = 0.92$

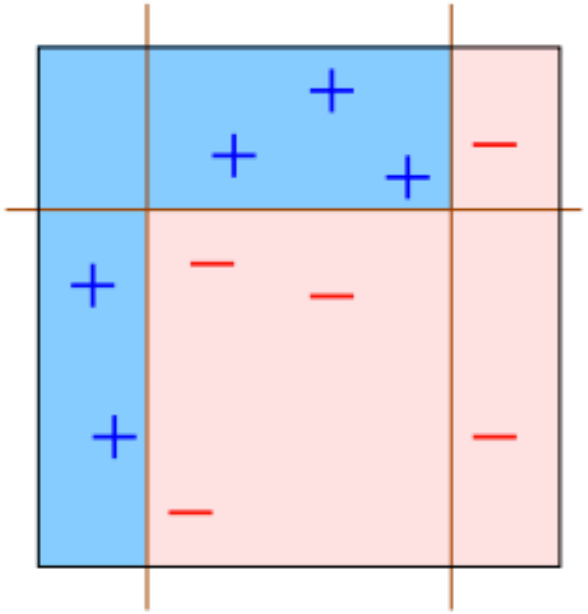
Weak classifiers = vertical or horizontal half- planes = hypothesis from $\mathcal{H}_{DS}^2$

Toy example – final classifier

$H_{\text{final}} = \text{sign}\left(0.42 \; + 0.65 \; + 0.92 \right)$



=



Bound on the training error

Theorem

Let S be a training set and assume that at each iteration of AdaBoost, the weak learner returns a hypothesis for which the error $\varepsilon_t \leq 1/2 - \gamma$. Then, the training error of the output hypothesis of AdaBoost after T rounds is at most:

$$L_S(h_{final}) \leq e^{-2\gamma^2 T}$$

Proof

-in the book, apply some mathematical tricks that do not present interest for us to discuss

So we can asymptotically obtain zero training error, provided we make many rounds T and we are able to find weak learners that return a hypothesis for which the error $\varepsilon_t \leq 1/2 - \gamma$, where $\gamma > 0$.

Of course, we are interested in the generalization error.

Generalization error for AdaBoost

Remember from lecture 7 (use of the Sauer lemma):

for every $\mathcal{D}$ and every $\delta \in (0,1)$, with probability of at least $1-\delta$ over the choice of $S \sim \mathcal{D}^m$ we have:

$$\left|L_D(h) - L_S(h)\right| \leq \frac{4+\sqrt{\log(\tau_H(2m))}}{\delta\sqrt{2m}} \leq \frac{4+\sqrt{d\log(2em/d)}}{\delta\sqrt{2m}} \leq \frac{2\sqrt{d\log(2em/d)}}{\delta\sqrt{2m}}$$

where d = VCdim($\mathcal{H}$), $\mathcal{H}$ is the class from which h_{final} comes.

As T grows, we have seen that $L_S(h_{final}) \rightarrow 0$.

As T grows, what happens with the VCdim($\mathcal{H}$) ?

Generalization error for AdaBoost

A popular approach for constructing a weak learner is to apply the ERM rule with respect to a base hypothesis class $\mathcal{B}$ (e.g., ERM over $\mathcal{H}_{DS}^{d}$ = decision stumps in *d* dimensions).

Given a base hypothesis class $\mathcal{B}$ (e.g., decision stumps), the output of AdaBoost will be a member of the following class:

$$L(\mathcal{B},T) = \{h: \mathbf{R}^d \rightarrow \{-1,1\},\ h(x) = sign(\sum_{t=1}^{T} w_t h_t(x))\ ,\ w \in \mathbf{R}^T,\ h \in \mathcal{B}\}$$

Remember the bias-complexity tradeoff: decompose the error of a predictor that chooses h_S from a restricted class $\mathcal{H}$ into two components:

$$L_D(h_S) = \varepsilon_{app} + \varepsilon_{est}$$

Generalization error for AdaBoost

$$L_D(h_S) = \varepsilon_{app} + \varepsilon_{est}$$

The approximation error (ε_{app})

the minimum risk achievable by a predictor in the hypothesis class $\mathcal{H}$

it is determined only by $\mathcal{H}$, enlarging it decreases the approximation error

$$\varepsilon_{app} = \min_{h \in H} L_D(h)$$

the expressiveness of $L(\mathcal{B}, T)$ grows with T

in other words, the approximation error decreases with T

The estimation error (ε_{est})

it measures how well our particular sample let us estimate the best classifier

the quality of this estimation depends on the training set size and on the size (complexity) of $\mathcal{H}$

we'll show that the estimation error increases with T

therefore, the parameter T of AdaBoost enables us to control the bias-complexity trade-off

Generalization error for AdaBoost

Lemma

Let $\mathcal{B}$ be a base class and let L($\mathcal{B}$,T) be as defined as previously:

$$L(\mathcal{B},T) = \{h: \mathbf{R}^d \rightarrow \{-1,1\},\ h(x) = sign(\sum_{t=1}^{T} w_t h_t(x)),\ w \in \mathbf{R}^T,\ h \in \mathcal{B}\}$$

Assume that both T and VCdim($\mathcal{B}$) are at least 3. Then, we have that:

$$VCdim(L(\mathcal{B},T)) \leq T \times (VCdim(\mathcal{B})+1) \times (3 \times \log(T \times (VCdim(\mathcal{B})+1))+2).$$

Proof

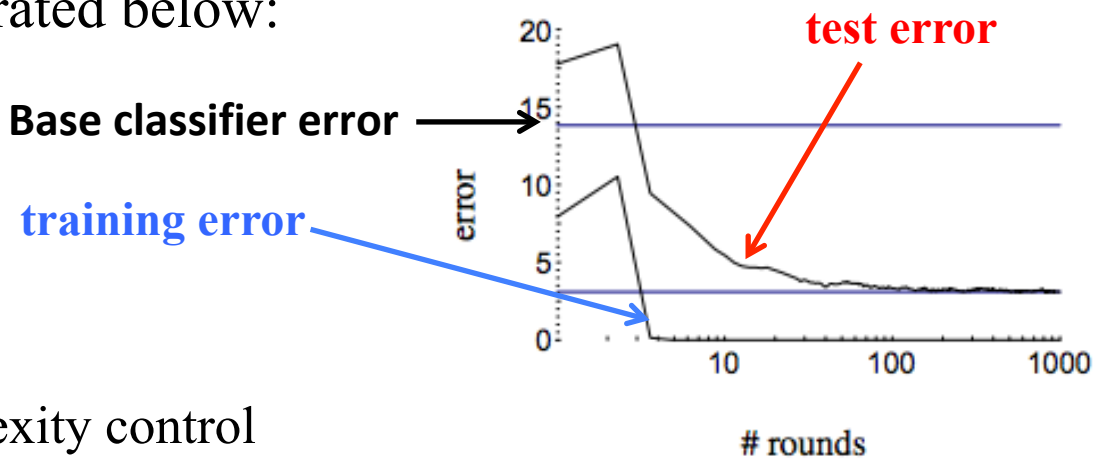
-in the book

Generalization error for AdaBoost

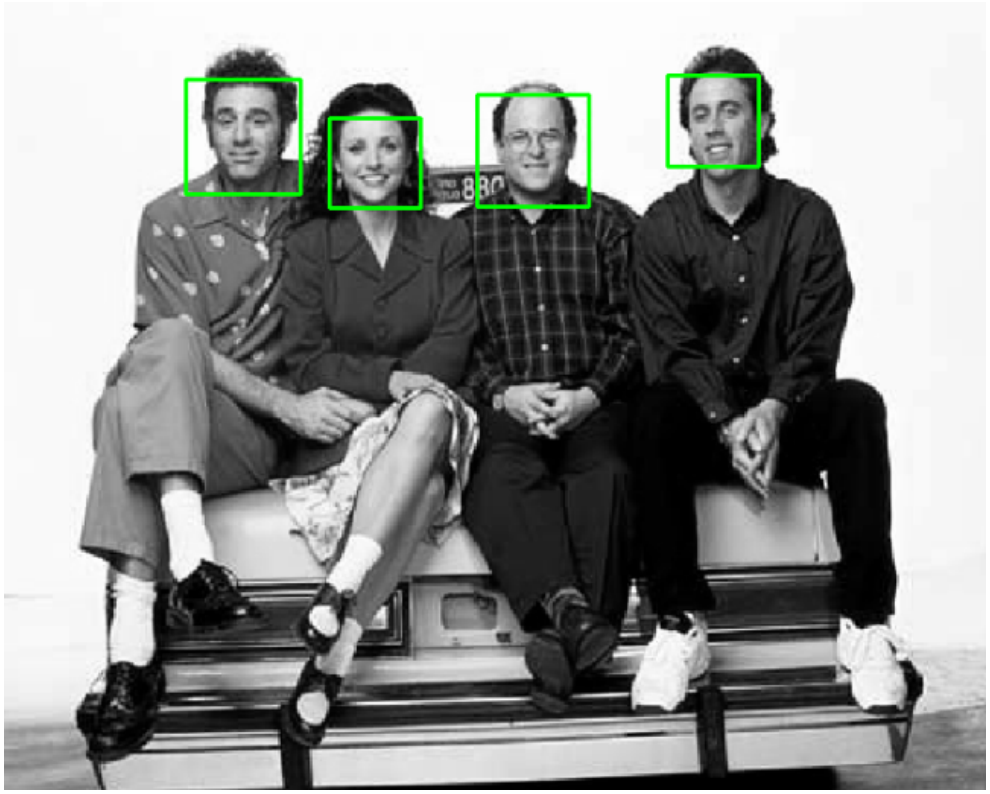
$VCdim(L(\mathcal{B},T)) \leq T \times (VCdim(\mathcal{B})+1) \times (3 \times \log(T \times (VCdim(\mathcal{B})+1))+2)$.

The upper bound grows as $O(T \times VCdim(\mathcal{B}) \times \log(T \times VCdim(\mathcal{B})))$, thus, the bound suggests that AdaBoost could overfit for large values of T, and indeed this can occur.

However, in many cases, it has been observed empirically that the generalization error of AdaBoost decreases as a function of the number of rounds of boosting T, as illustrated below:



- number of rounds *T* is complexity control
- use validation set + "early stopping" to select *T*

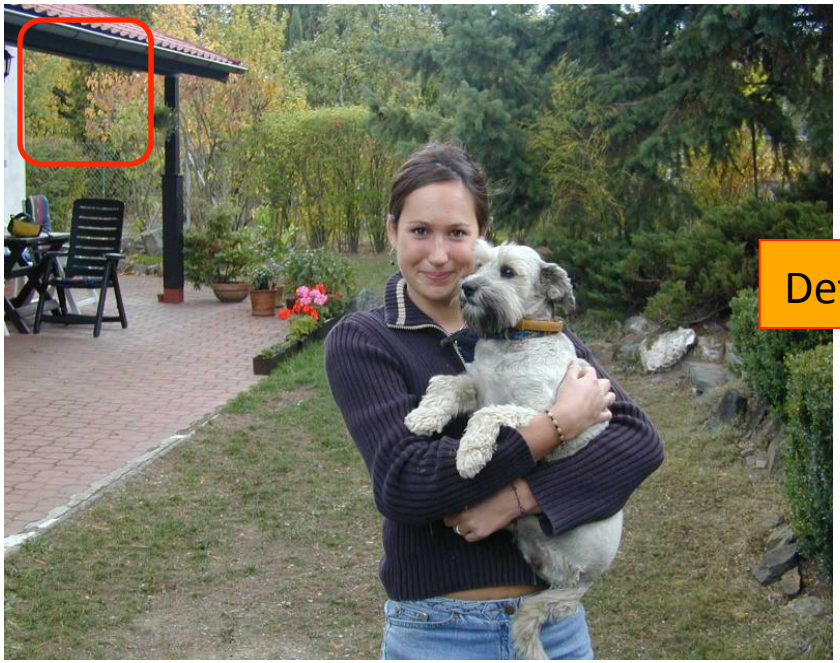


Face detection with AdaBoost

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Face detection vs face recognition



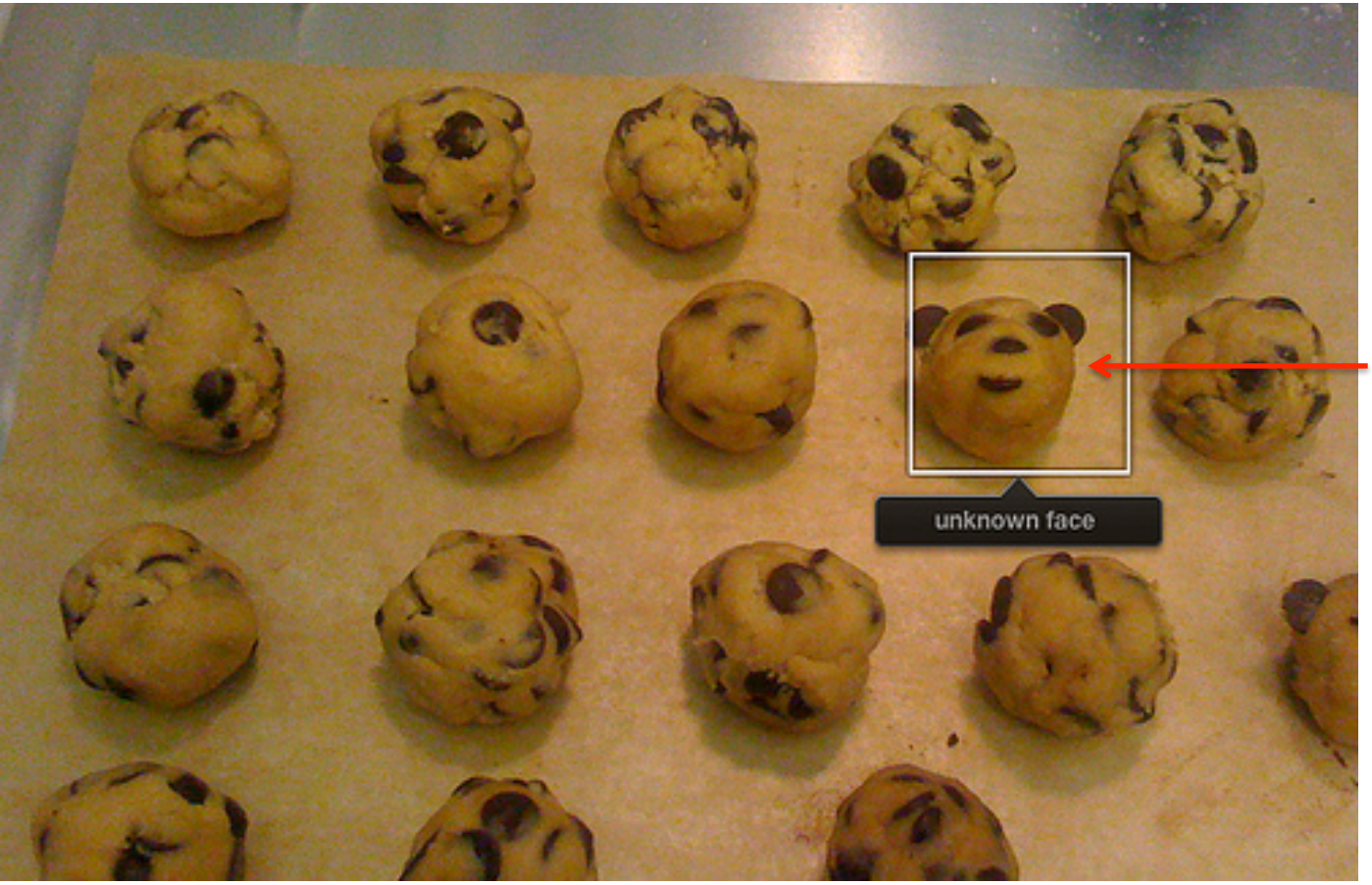
Detection

Recognition

"Maria"

Slide credit S. Lazebnik

Precision



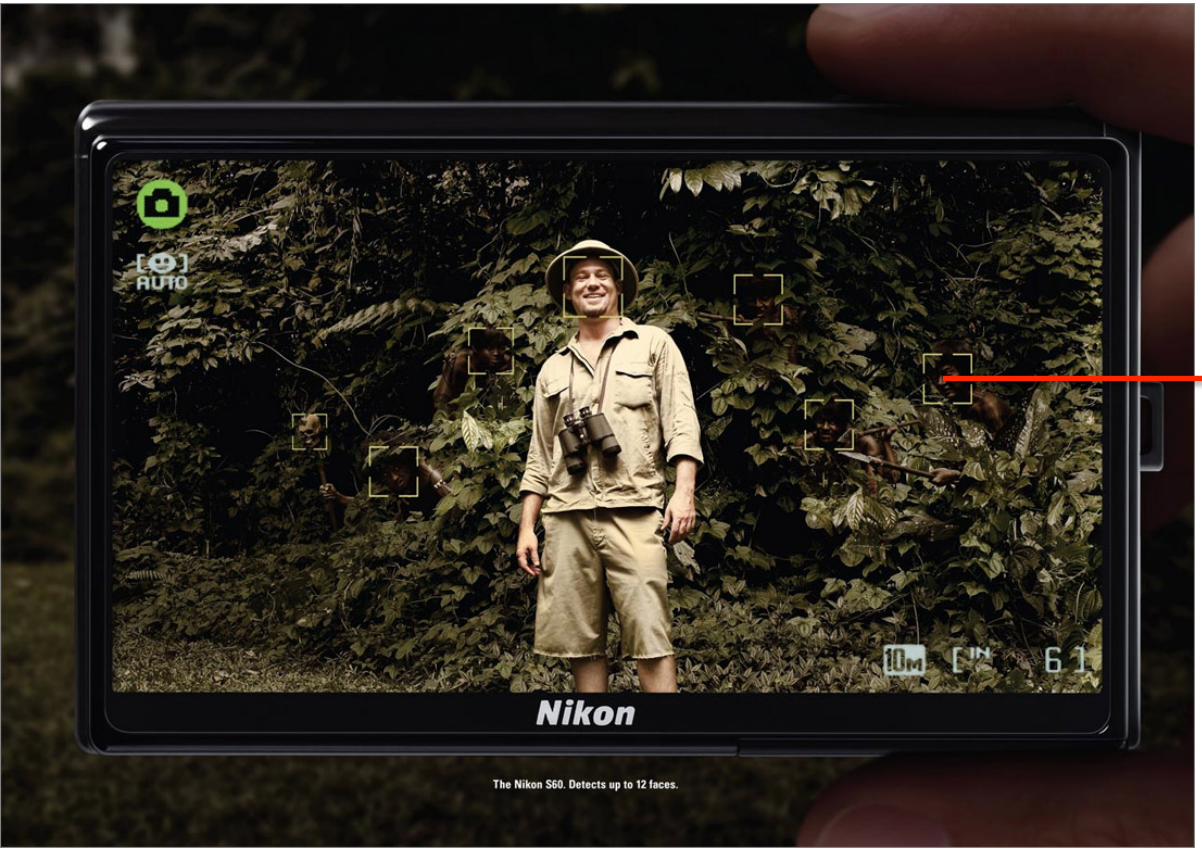
False positive example

$$PRECISION = \frac{TP}{TP + FP}$$

TP – true positives, FP – false positives

Slide credit S. Lazebnik

Recall



ZOOM

$$RECALL = \frac{TP}{TP + FN}$$

TP – true positives, FN – false negatives

Slide credit S. Lazebnik

Viola-Jones face detector

ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001

Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola
viola@merl.com
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones
mjones@crl.dec.com
Compaq CRL
One Cambridge Center
Cambridge, MA 02142

Abstract

This paper describes a machine learning approach for vi-

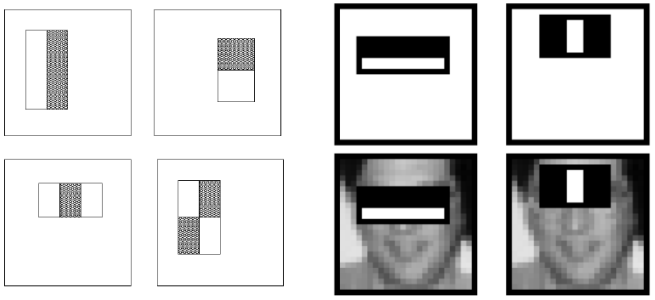
tected at 15 frames per second on a conventional 700 MHz Intel Pentium III. In other face detection systems, auxiliary information, such as image differences in video sequences,

Viola-Jones face detector

Main idea:

- represent local texture with efficiently computable "rectangular" features within window of interest
- select discriminative features to be weak classifiers
- use boosted combination of them as final classifier
- form a cascade of such classifiers, rejecting clear negatives quickly

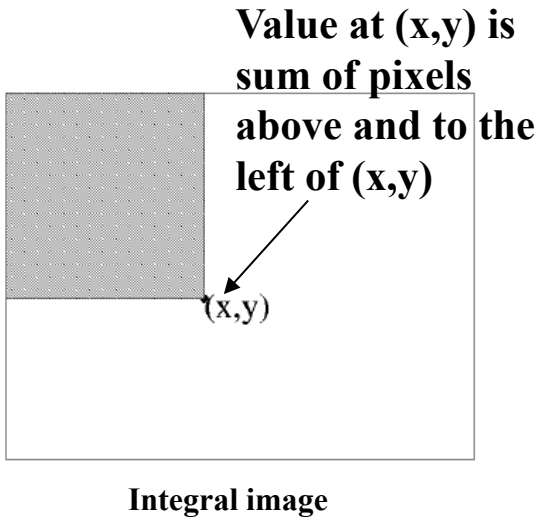
Viola-Jones face detector: features



"**Rectangular" filters**

Feature output is difference between adjacent regions

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



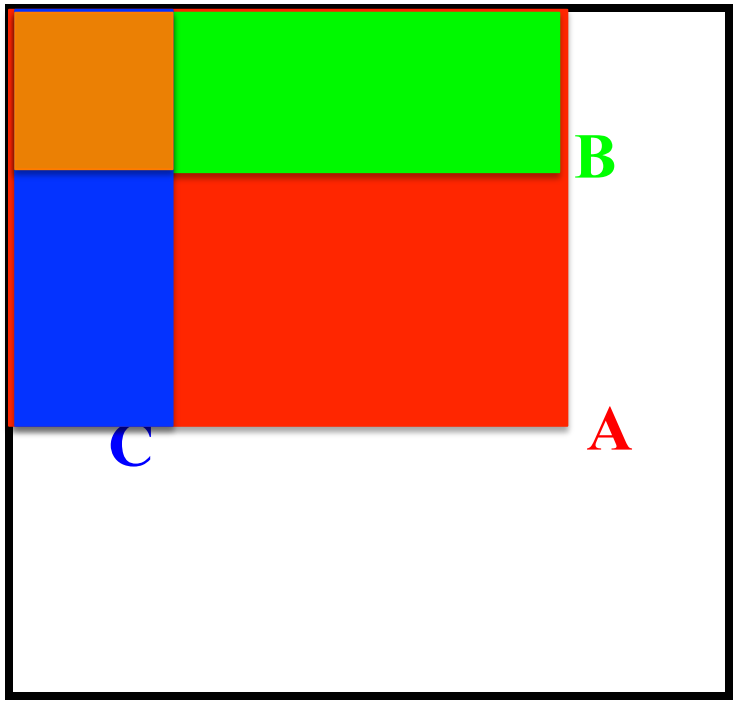
Integral image

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 as:

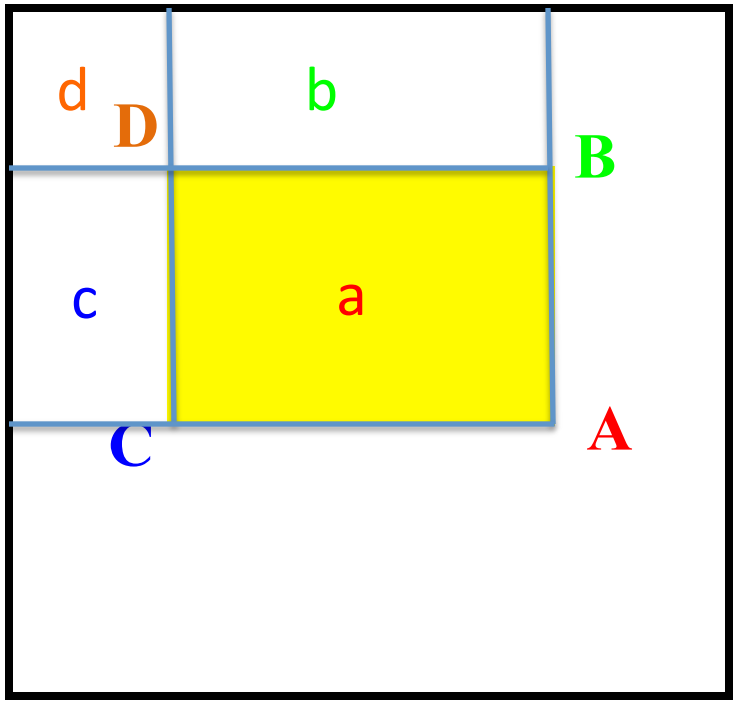
  sum = A – B – C + D



Integral image

Computing sum within a rectangle

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



Integral image

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

sum = A – B – C + D

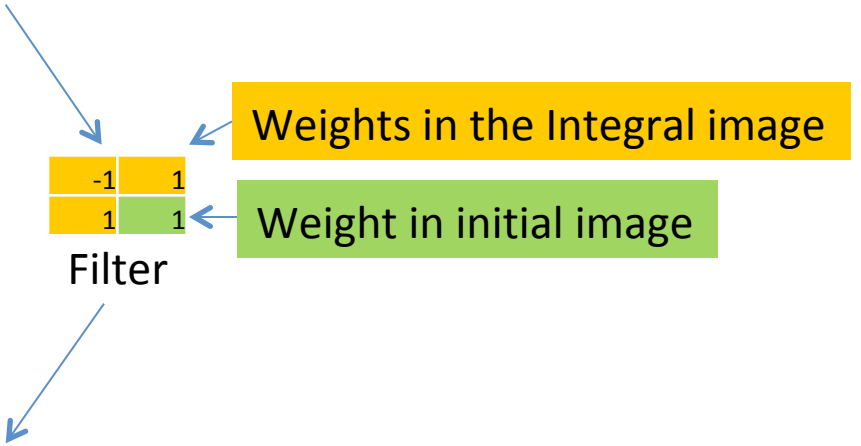
= (a + b + c + d) – (b + d) – (c + d) + d = a

- Only 3 operations (one addition and two subtractions) are required for any size of rectangle!

Building the integral image

34	11	33	3	19	19	18	18	2	24
34	8	36	11	5	11	5	6	15	33
17	22	17	4	6	3	5	7	35	18
8	3	15	22	5	1	20	10	12	22
8	7	1	22	19	29	6	20	9	27
16	7	11	17	15	2	25	19	29	10
34	26	29	31	5	6	30	17	4	10
33	28	30	4	28	21	26	5	32	21
1	18	13	5	27	16	28	19	32	23
12	13	16	23	13	7	21	5	2	15

Initial image



34

Integral image

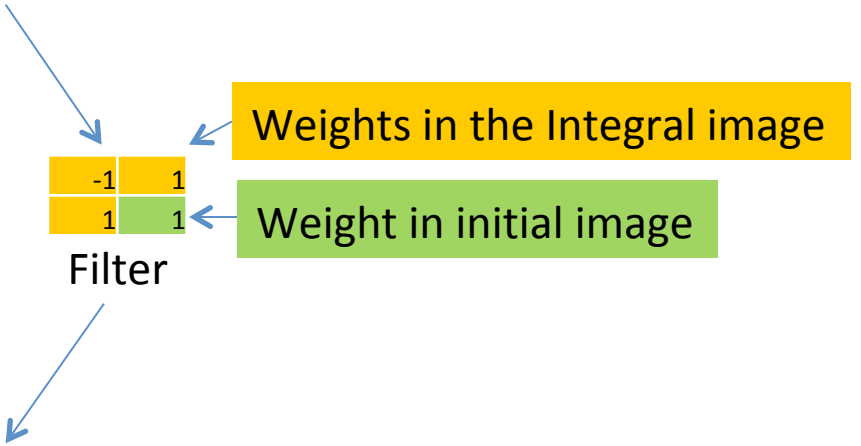
Building the integral image

34	11	33	3	19	19	18	18	2	24
34	8	36	11	5	11	5	6	15	33
17	22	17	4	6	3	5	7	35	18
8	3	15	22	5	1	20	10	12	22
8	7	1	22	19	29	6	20	9	27
16	7	11	17	15	2	25	19	29	10
34	26	29	31	5	6	30	17	4	10
33	28	30	4	28	21	26	5	32	21
1	18	13	5	27	16	28	19	32	23
12	13	16	23	13	7	21	5	2	15

Initial image

34	45

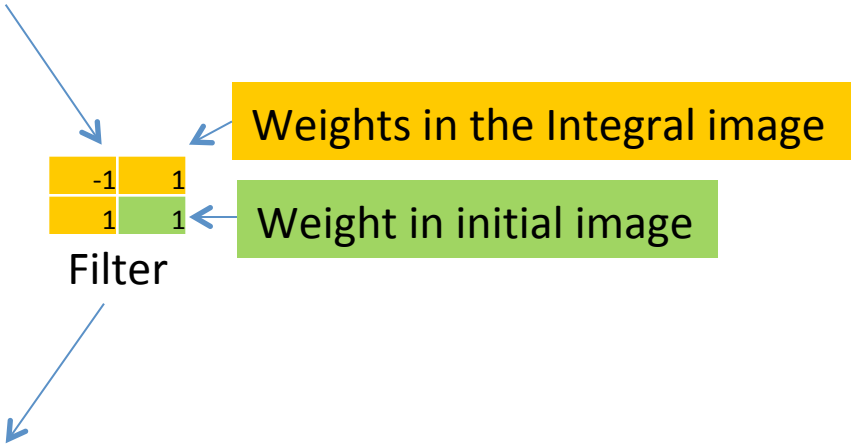
Integral image



Building the integral image

34	11	33	3	19	19	18	18	2	24
34	8	36	11	5	11	5	6	15	33
17	22	17	4	6	3	5	7	35	18
8	3	15	22	5	1	20	10	12	22
8	7	1	22	19	29	6	20	9	27
16	7	11	17	15	2	25	19	29	10
34	26	29	31	5	6	30	17	4	10
33	28	30	4	28	21	26	5	32	21
1	18	13	5	27	16	28	19	32	23
12	13	16	23	13	7	21	5	2	15

Initial image



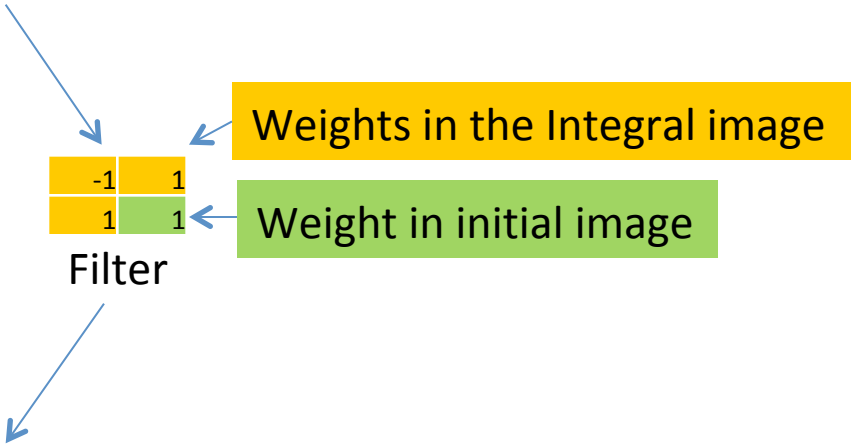
34	45	78

Integral image

Building the integral image

34	11	33	3	19	19	18	18	2	24
34	8	36	11	5	11	5	6	15	33
17	22	17	4	6	3	5	7	35	18
8	3	15	22	5	1	20	10	12	22
8	7	1	22	19	29	6	20	9	27
16	7	11	17	15	2	25	19	29	10
34	26	29	31	5	6	30	17	4	10
33	28	30	4	28	21	26	5	32	21
1	18	13	5	27	16	28	19	32	23
12	13	16	23	13	7	21	5	2	15

Initial image



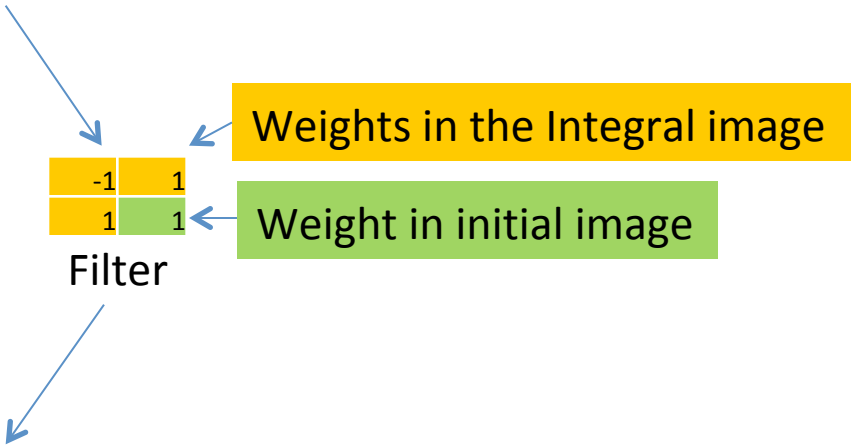
34	45	78	81	100	119	137	155	157	181

Integral image

Building the integral image

34	11	33	3	19	19	18	18	2	24
34	8	36	11	5	11	5	6	15	33
17	22	17	4	6	3	5	7	35	18
8	3	15	22	5	1	20	10	12	22
8	7	1	22	19	29	6	20	9	27
16	7	11	17	15	2	25	19	29	10
34	26	29	31	5	6	30	17	4	10
33	28	30	4	28	21	26	5	32	21
1	18	13	5	27	16	28	19	32	23
12	13	16	23	13	7	21	5	2	15

Initial image

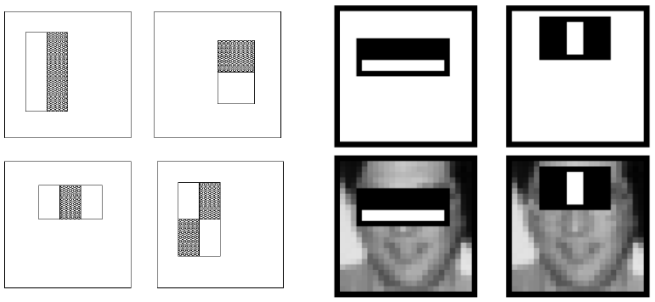


34	45	78	81	100	119	137	155	157	181
68	87	156	170	194	224	247	271	288	345
85	126	212	230	260	293	321	352	404	479
93	137	238	278	313	347	395	436	500	597
101	152	254	316	370	433	487	548	621	745
117	175	288	367	436	501	580	660	762	896
151	235	377	487	561	632	741	838	944	1088
184	296	468	582	684	776	911	1013	1151	1316
185	315	500	619	748	856	1019	1140	1310	1498
197	340	541	683	825	940	1124	1250	1422	1625

Integral image

O(N) operations!
N is # pixels in the image

Viola-Jones face detector: features

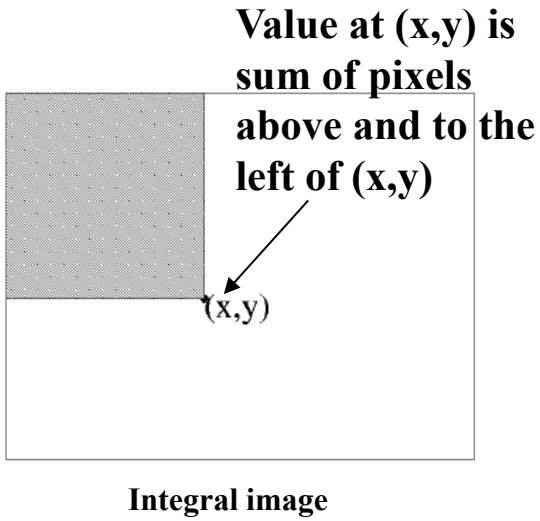


"**Rectangular" filters**

Feature output is difference between adjacent regions

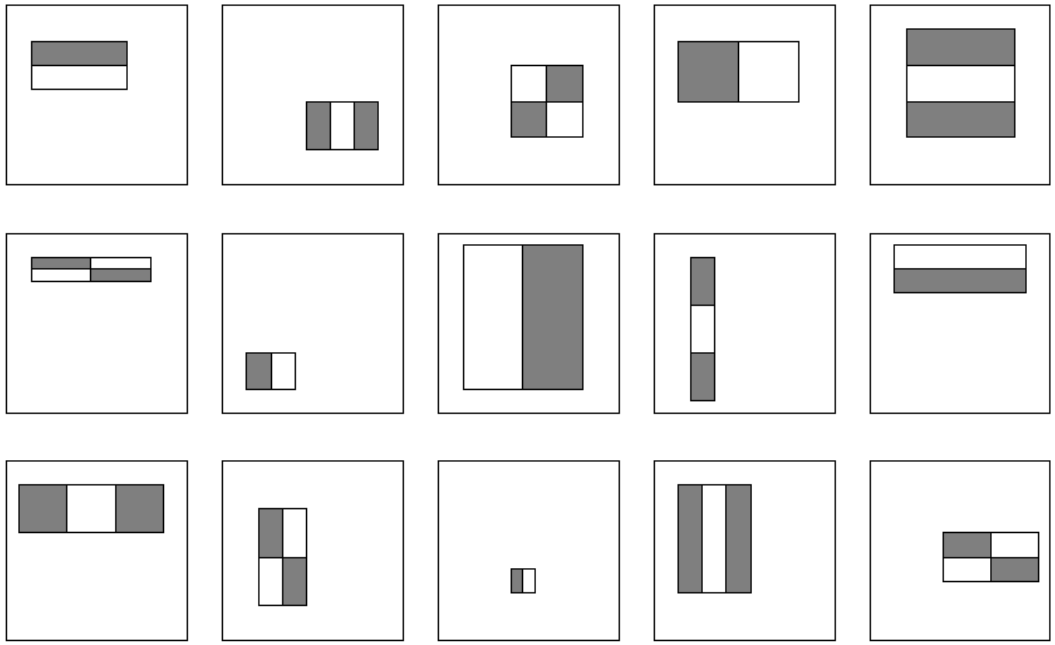
Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost



Integral image

Viola-Jones face detector: features



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

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

Which subset of these features should we use to determine if a window has a face?

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

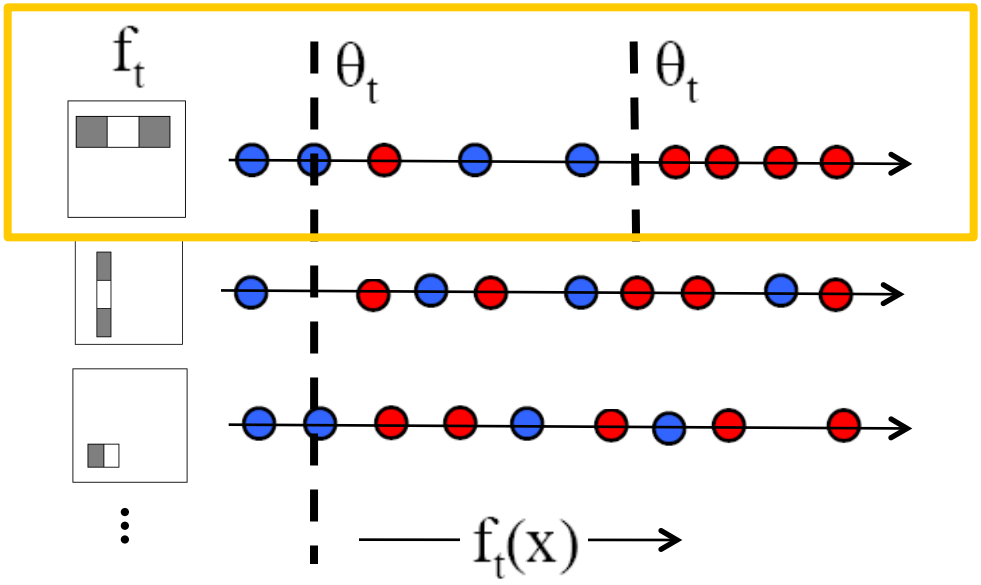
Training data

Faces

non-Faces

Viola-Jones detector: AdaBoost

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



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

imum number of examples are misclassified. A weak classifier $h_j(x)$ thus consists of a feature f_j, a threshold θ_j and a parity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

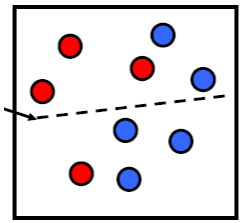
For next round, reweight the examples according to errors, choose another filter/threshold combo.

The resulting weak classifier is in fact from $\mathcal{H}_{DS}^{d} = \{h_{i,\theta,b}: \mathbf{R}^d \rightarrow \{-1,1\}$, $h_{i,\theta,b}(\mathbf{x}) = \text{sign}(\theta - x_i) \times b$, $1 \leq i \leq d$, $\theta \in \mathbf{R}$, $b \in \{-1,+1\}\}$

AdaBoost Algorithm

- 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, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- 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 $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

For T rounds

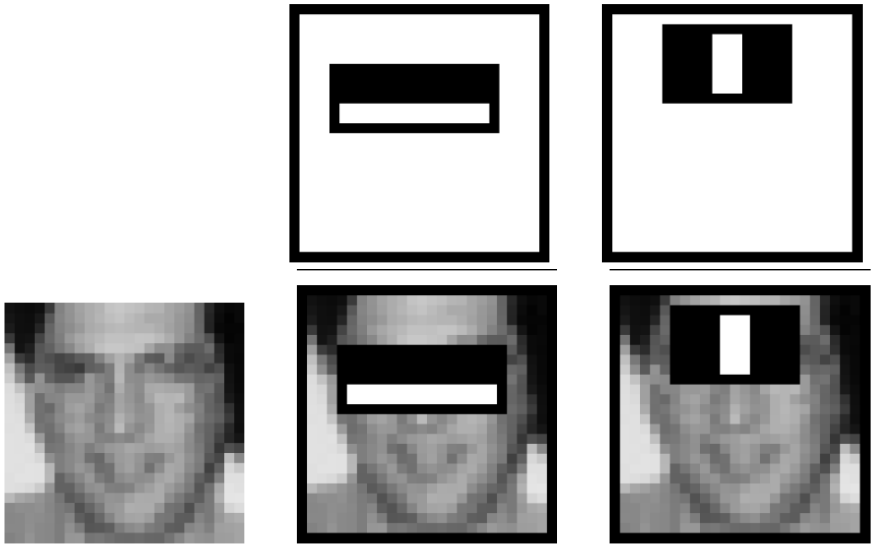
Evaluate *weighted* error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

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

Freund & Schapire 1995

Viola-Jones Face Detector: Results



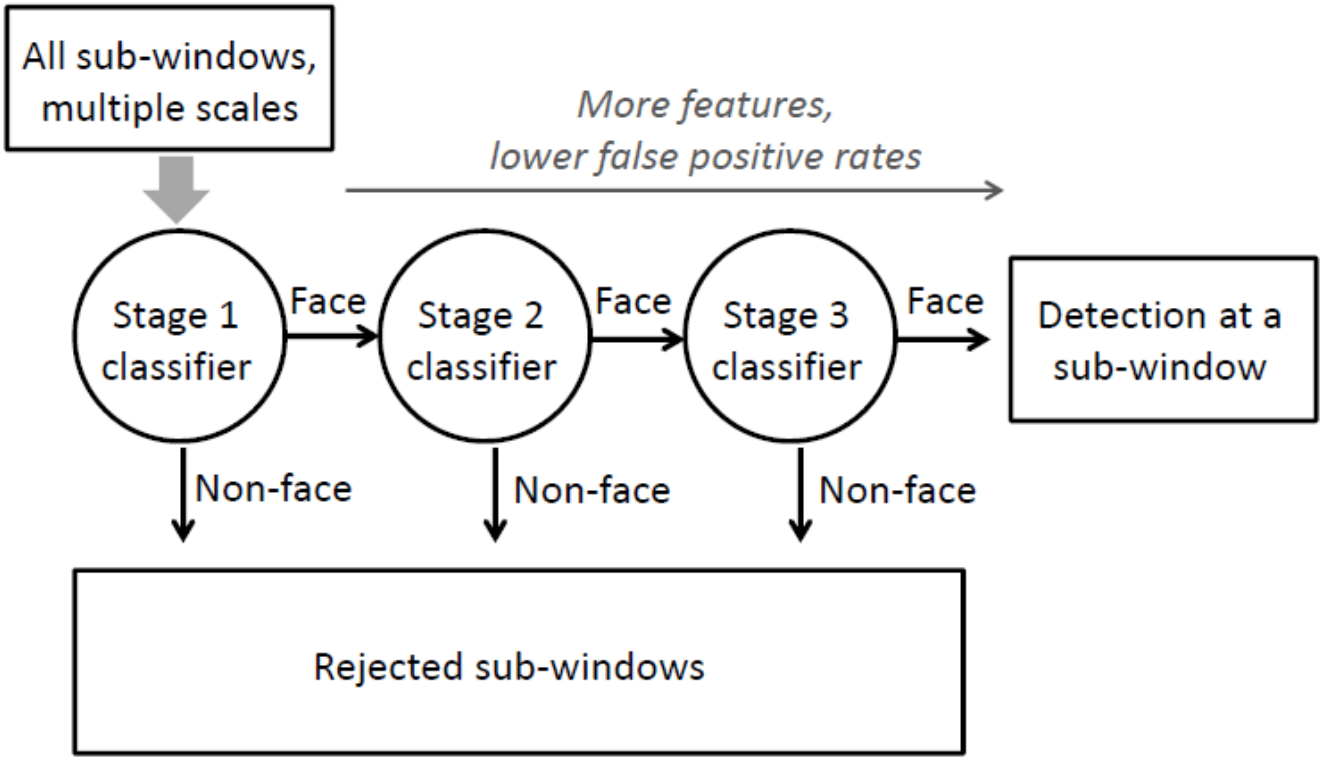
First two features selected

Faster?

- Even if the filters are fast to compute, each new image has a lot of possible windows to search.

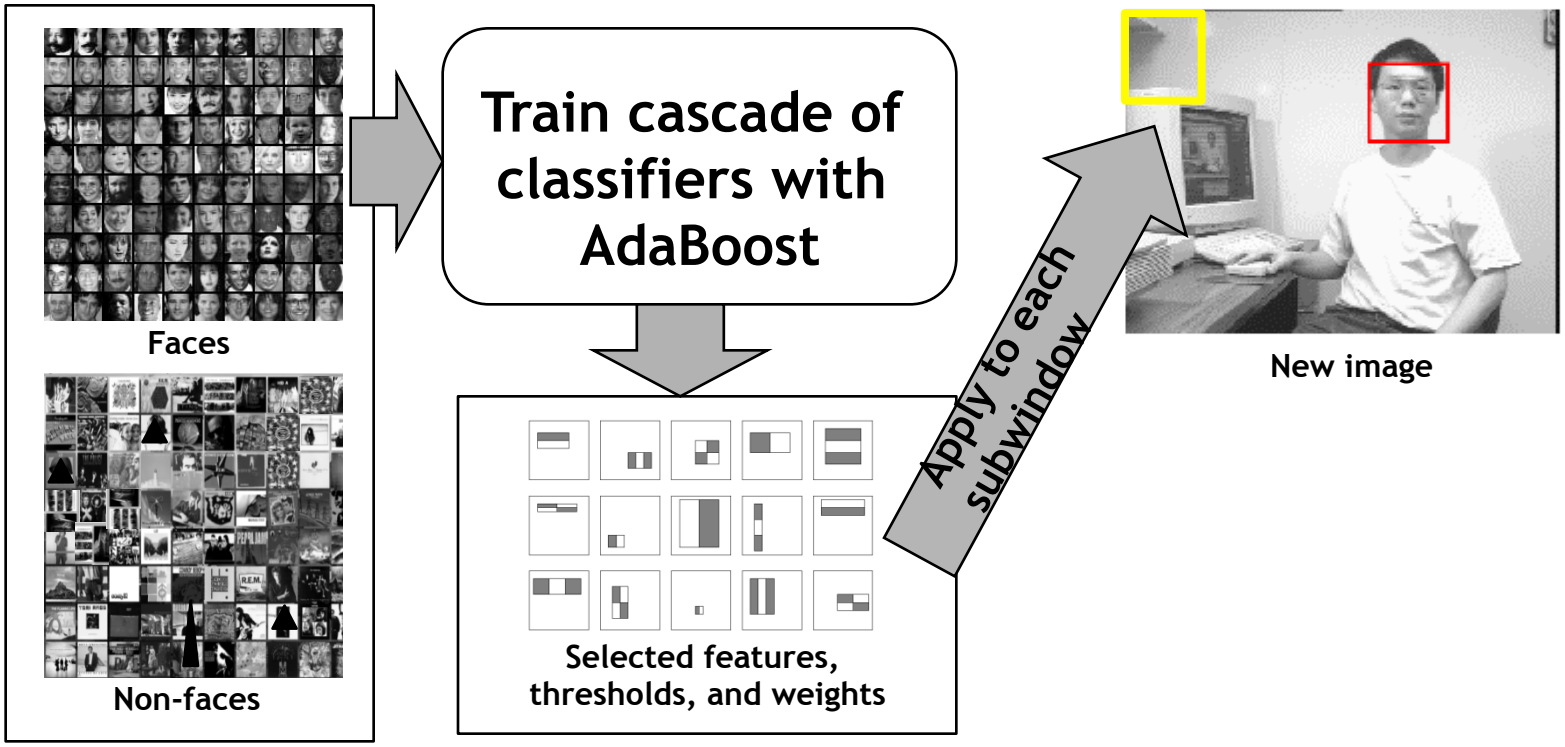
- How to make the detection more efficient?

Cascading classifiers for detection



- Form a *cascade* with low false negative rates early on
- Apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative

Viola-Jones detector: summary



Train with 5K positives, 350M negatives
Real-time detector using 38 layer cascade
6061 features in all layers

Implementation available in OpenCV, Matlab

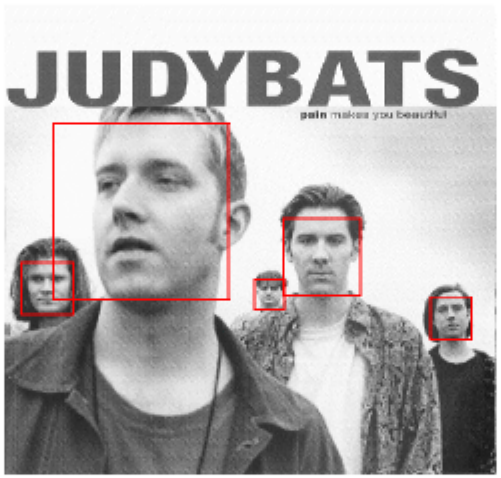
Viola-Jones detector: summary

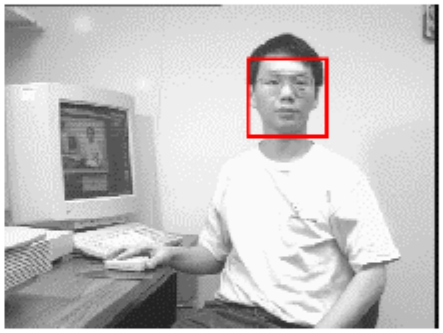
- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* of classifiers for fast rejection of non-face windows

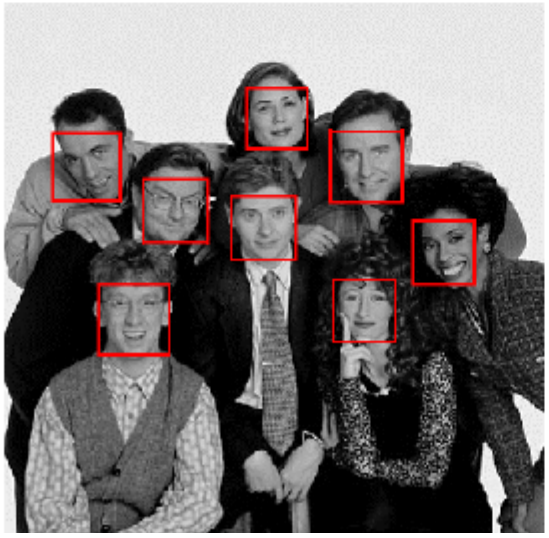
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

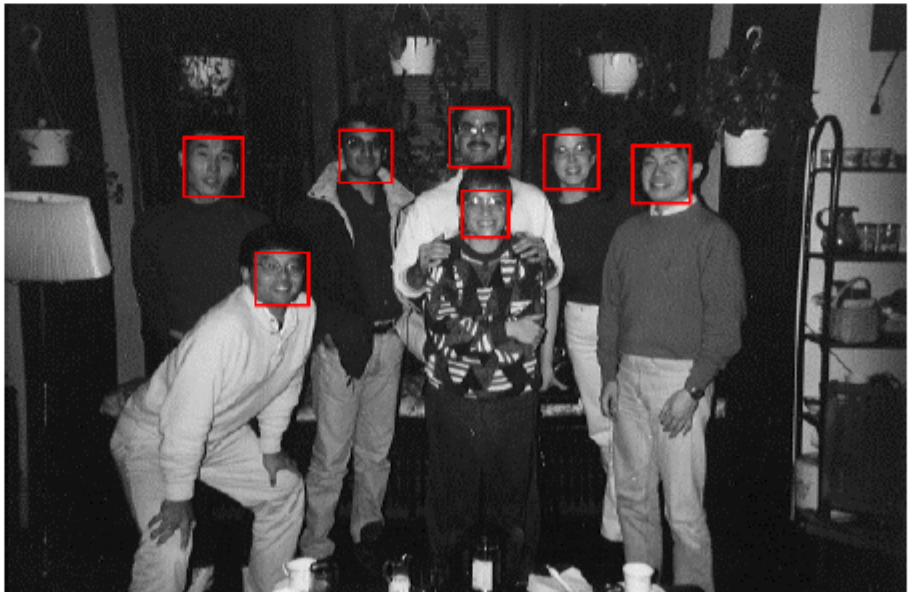
P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Viola-Jones Face Detector: Results

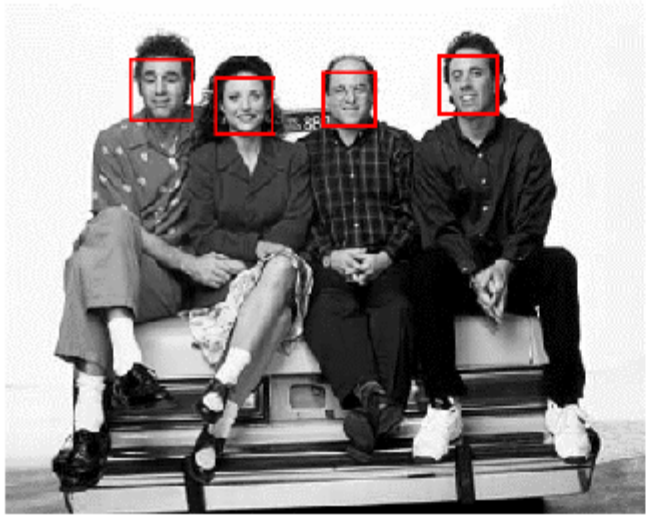


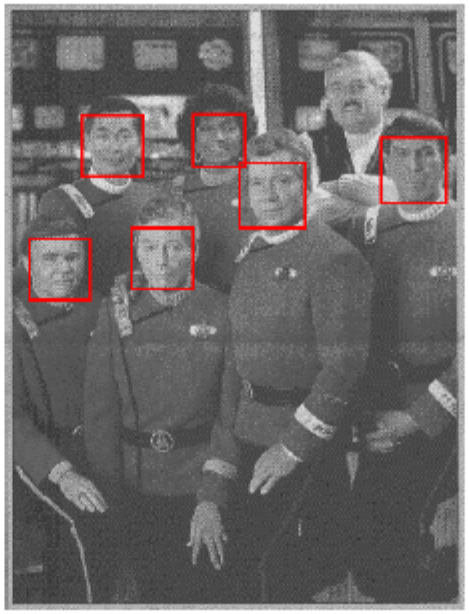


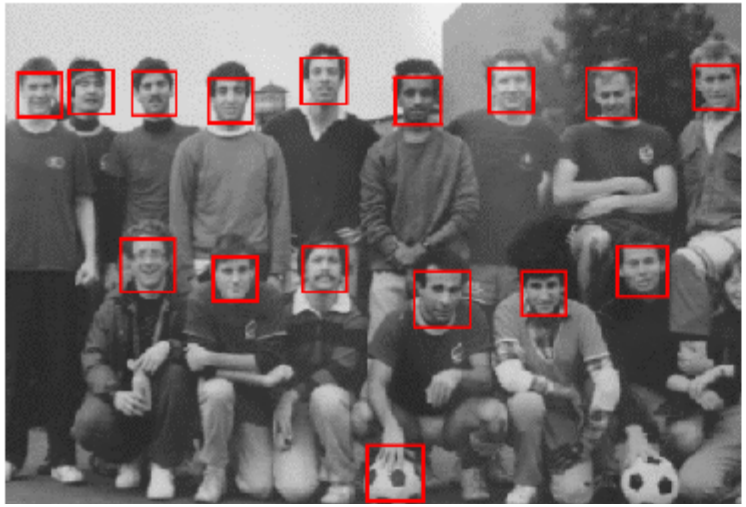


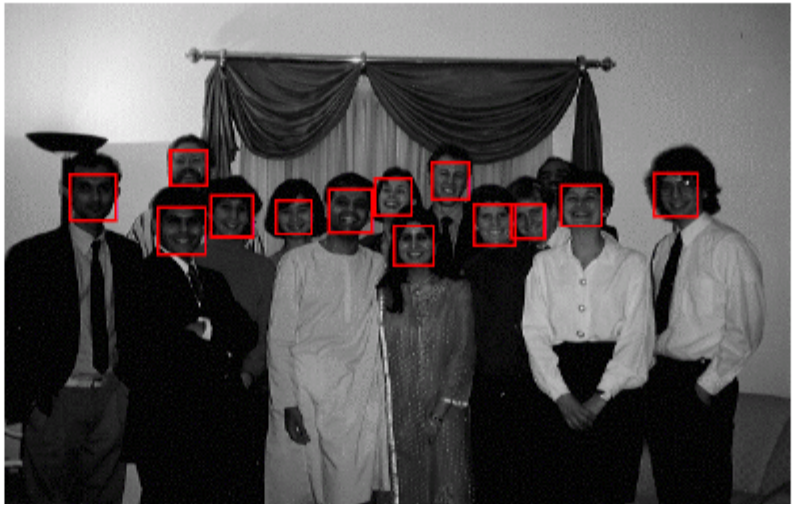


Viola-Jones Face Detector: Results





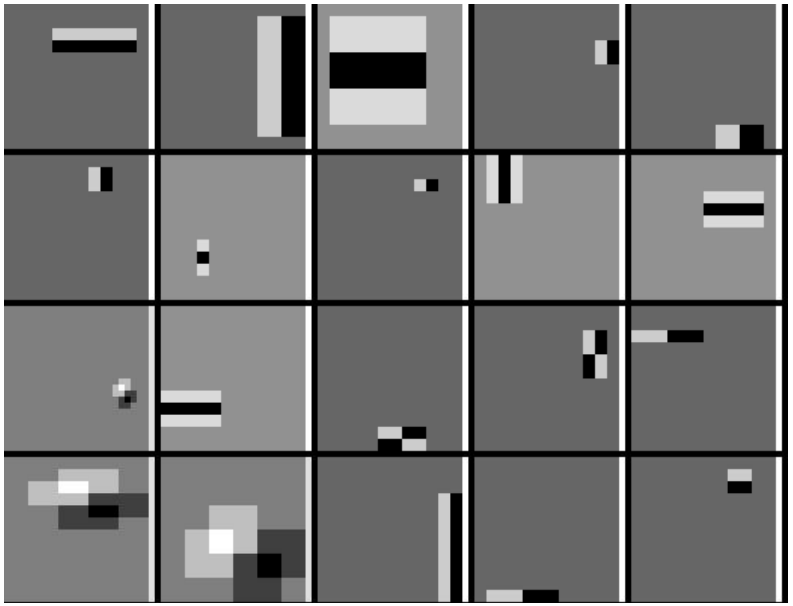




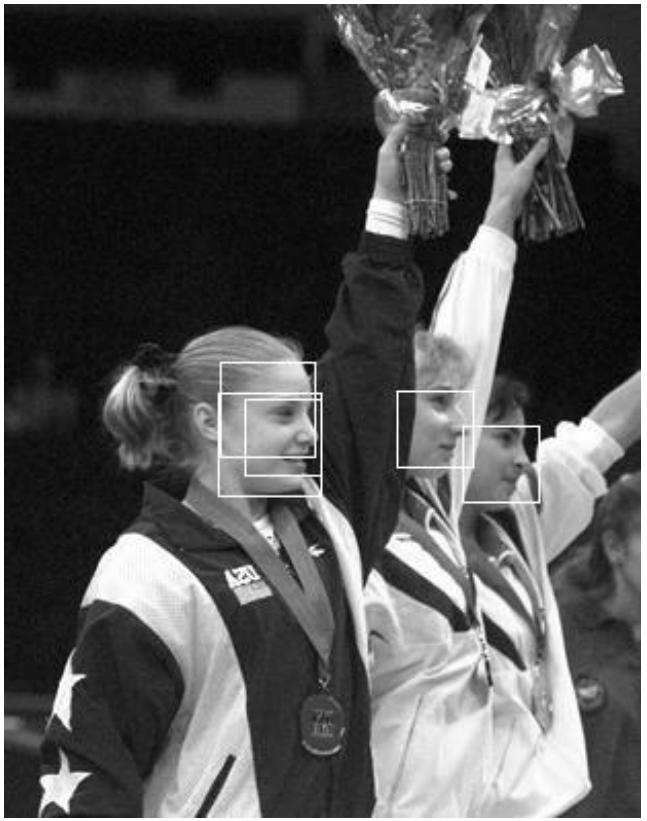
Viola-Jones Face Detector: Results

Detecting profile faces?

Can we use the same detector?



Viola-Jones Face Detector: Results



Detectorul facial Viola-Jones: rezultate