Advanced Machine Learning

Bogdan Alexe,

bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2019-2020

Recap - 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 $D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{w_t}}{Z_{t+1}}$

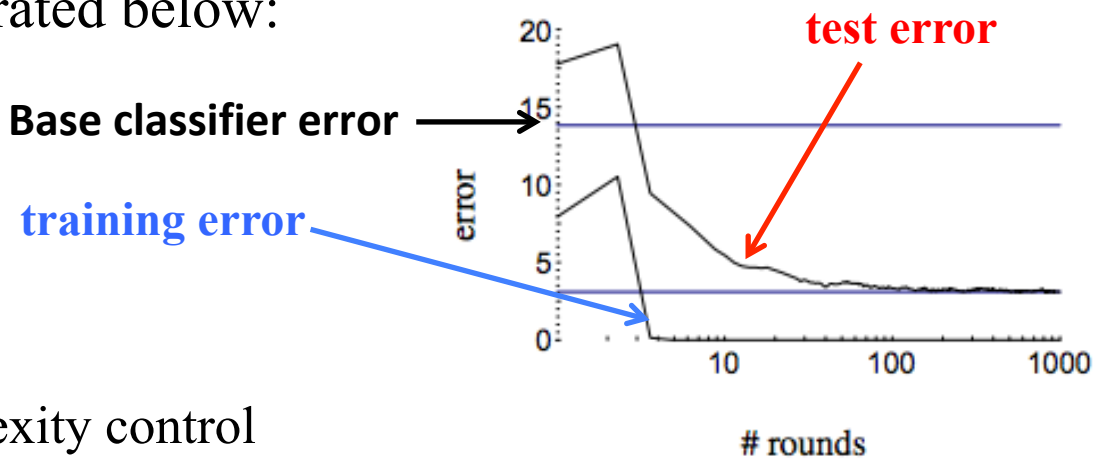
- output final/combined classifier h_{final}: $h_{final}(x) = sign(\sum_{t=1}^{T} w_t h_t(x))$

Recap - 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.

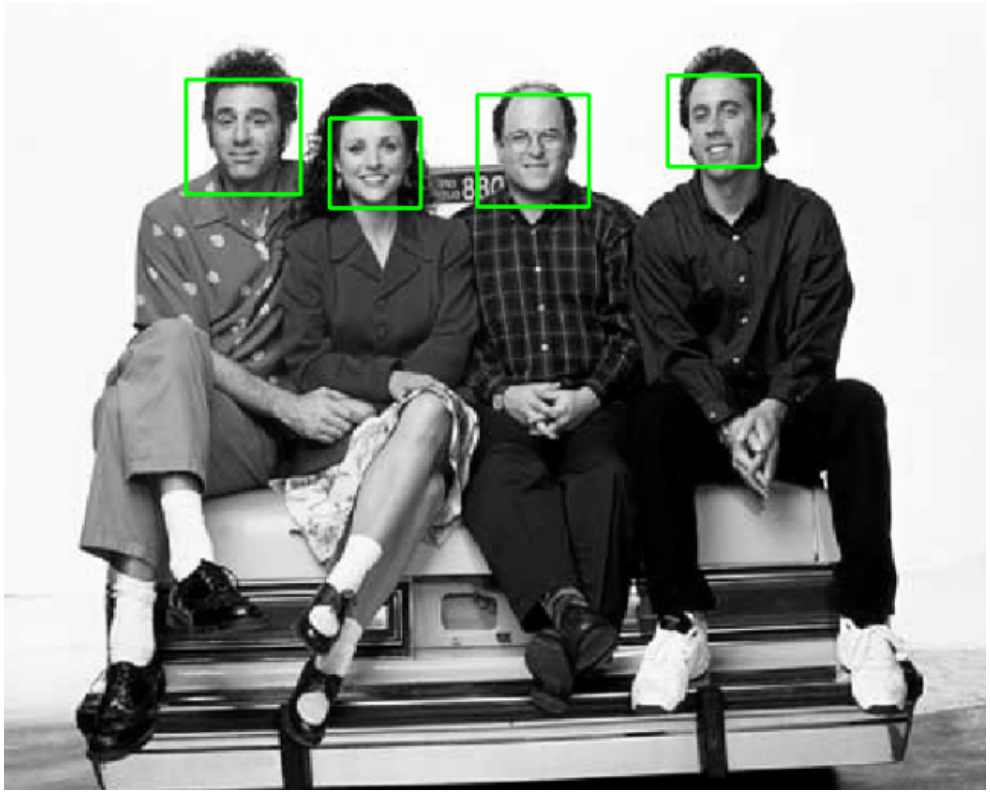
However, in many cases, it has been observed empirically that the generalization error of AdaBoost decreases as a function of the number of rounds of boosting T, as illustrated below:



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

Today's lecture: Overview

- Face detection with AdaBoost
- SVMs

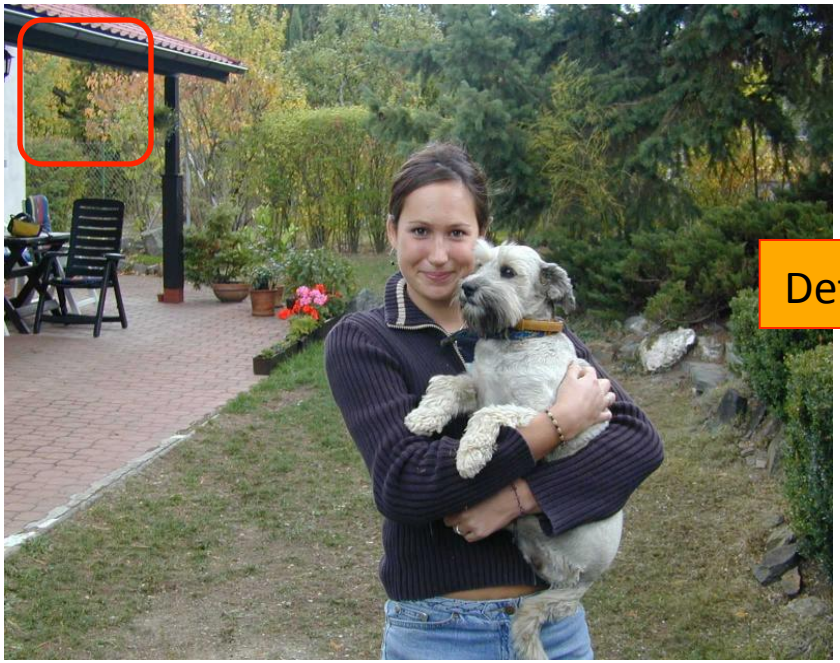


Face detection with AdaBoost

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



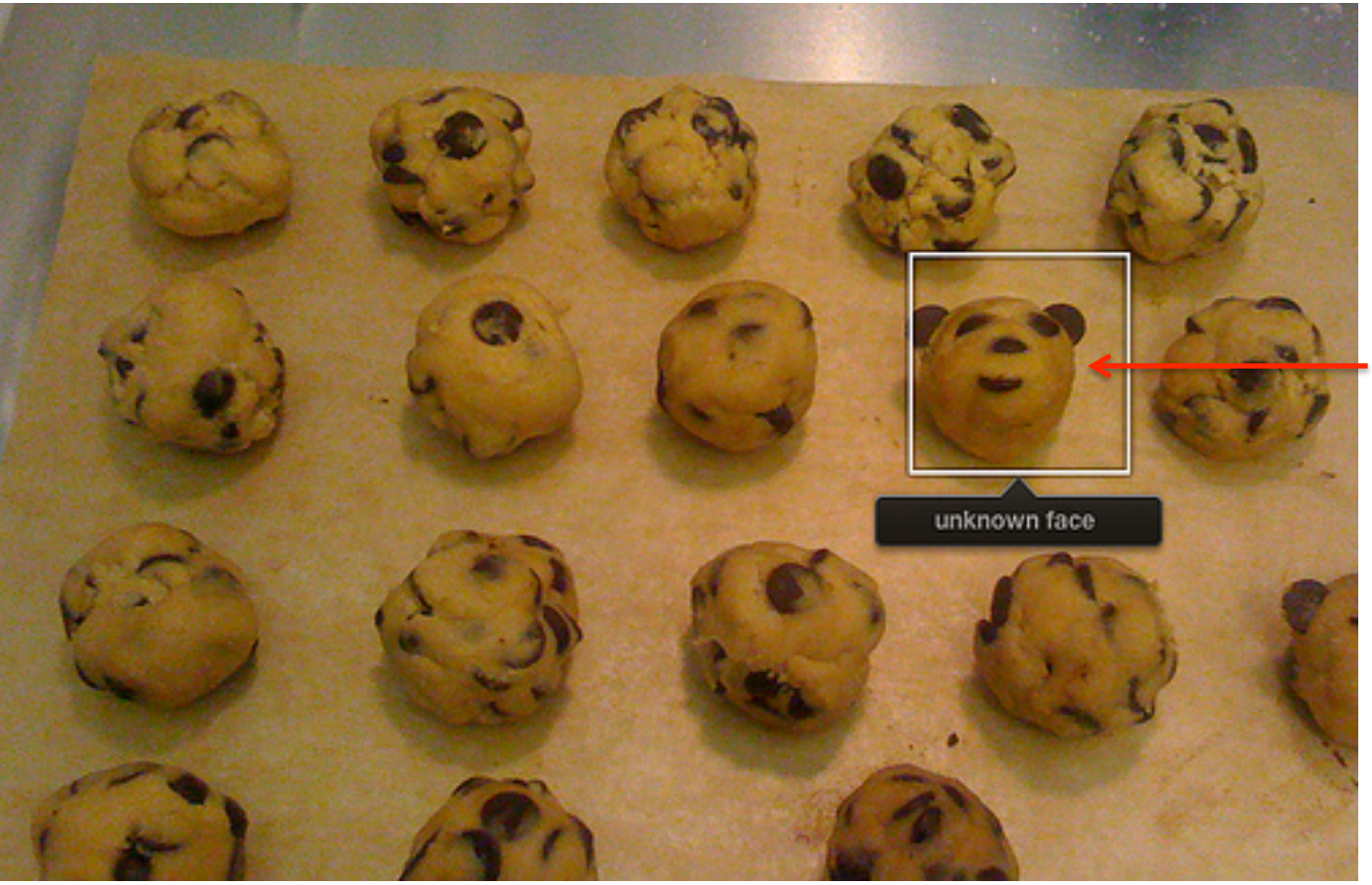
Detection

Recognition

"Maria"

Slide credit S. Lazebnik

Precision



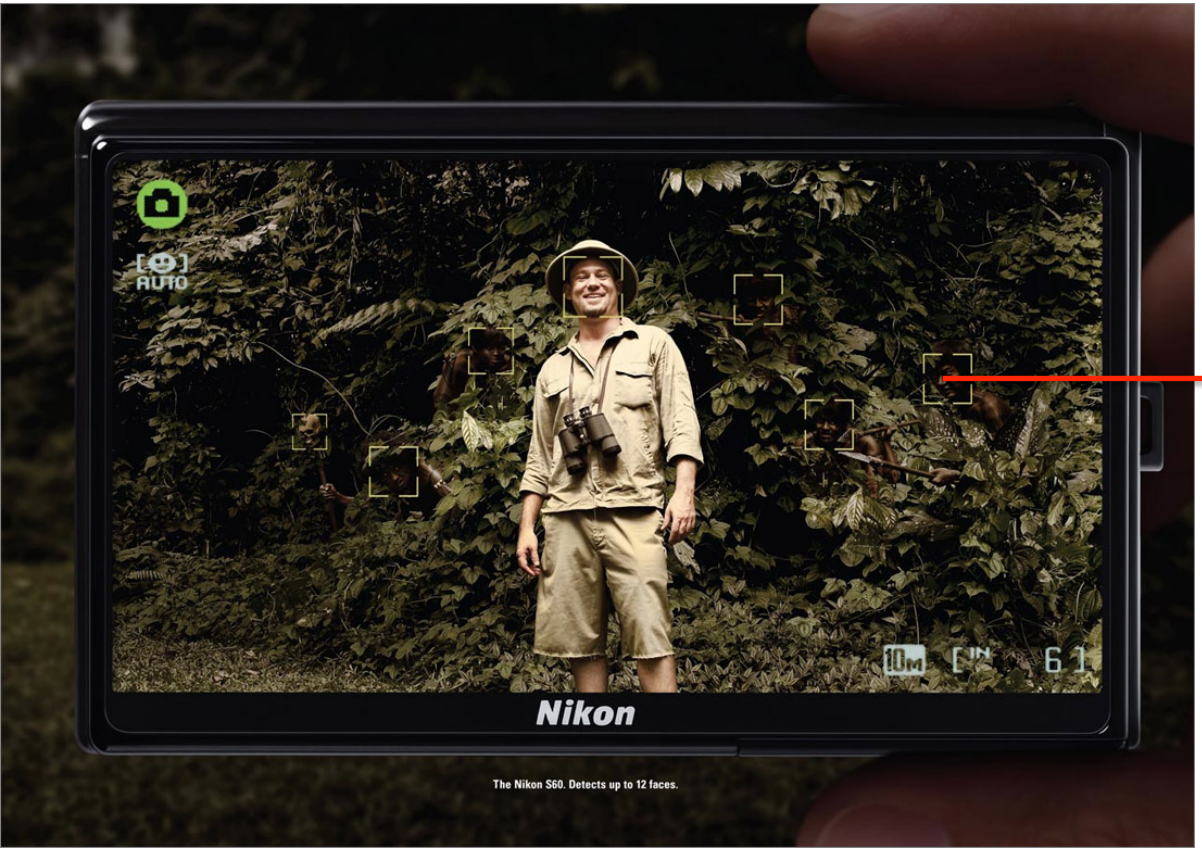
False positive example

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

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-

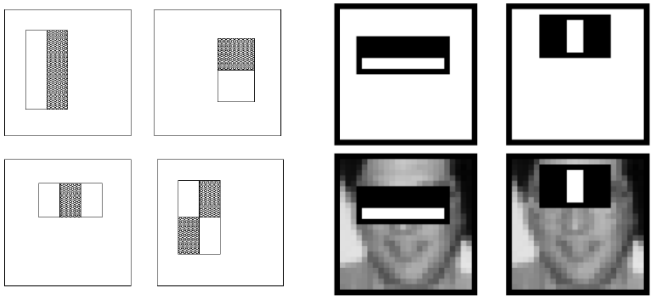
tected at 15 frames per second on a conventional 700 MHz Intel Pentium III. In other face detection systems, auxiliary information, such as image differences in video sequences,

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

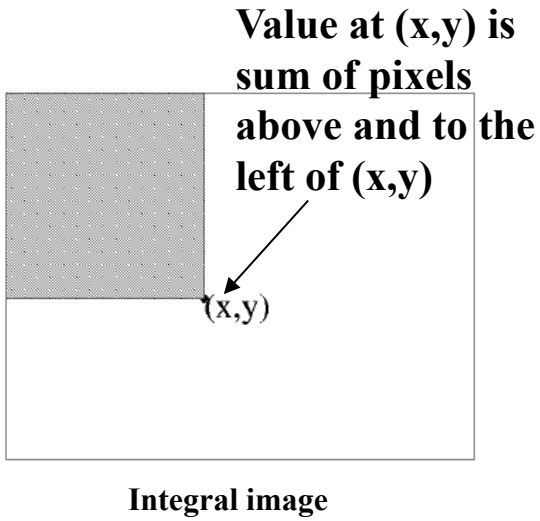
Viola-Jones face detector: features



"**Rectangular" filters**

Feature output is difference between adjacent regions

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



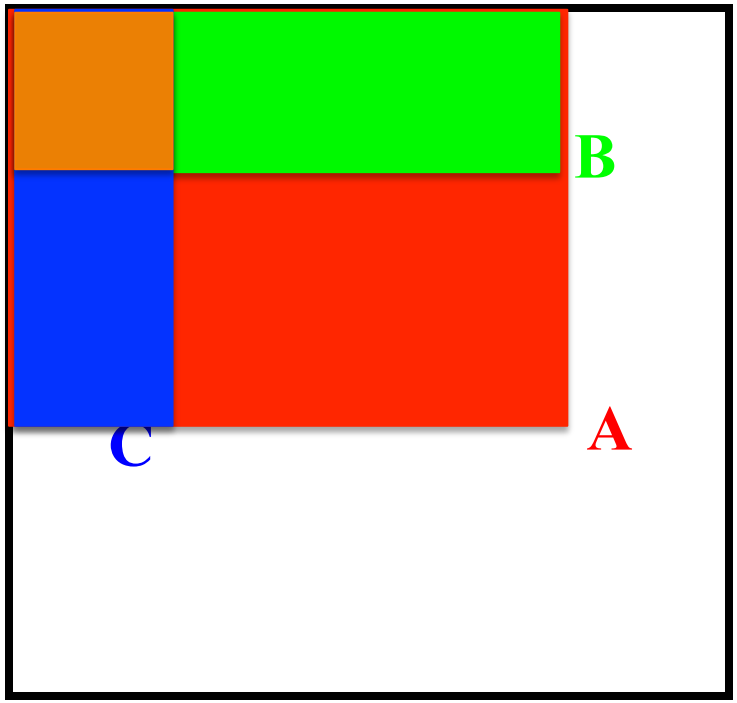
Integral image

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:

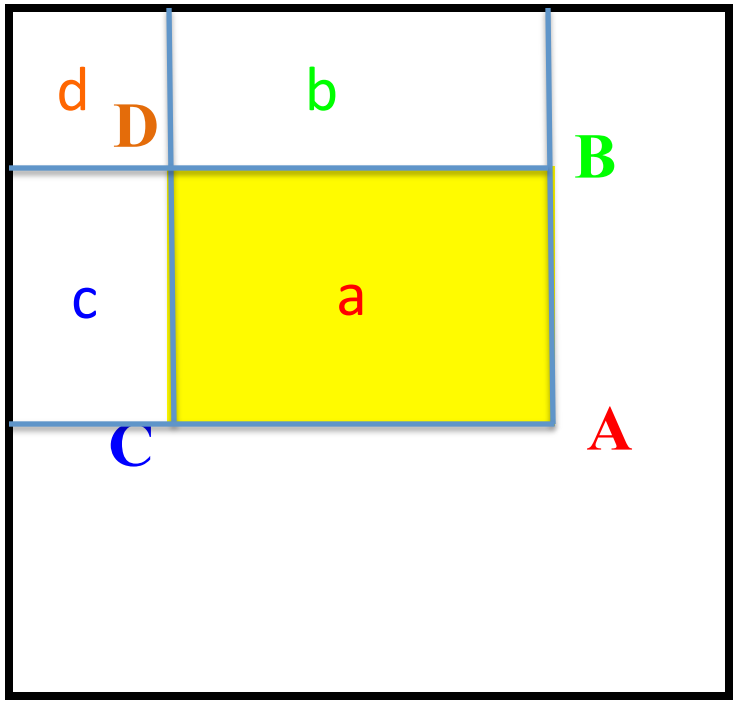
  sum = A – B – C + D



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

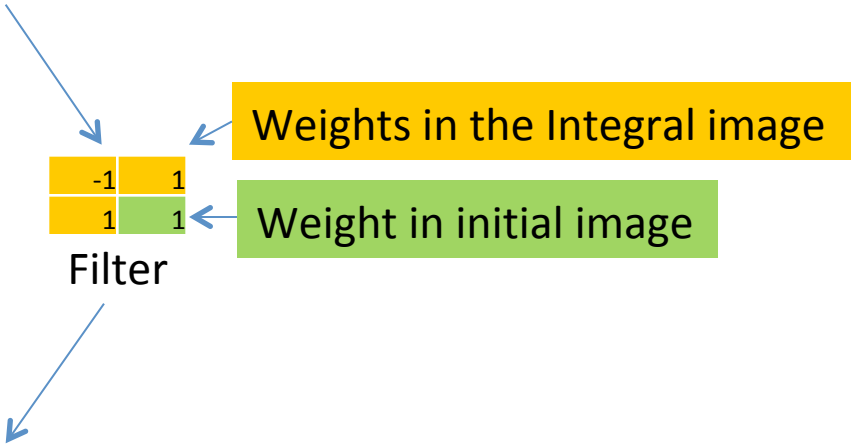
= (a + b + c + d) – (b + d) – (c + d) + d = a

- 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

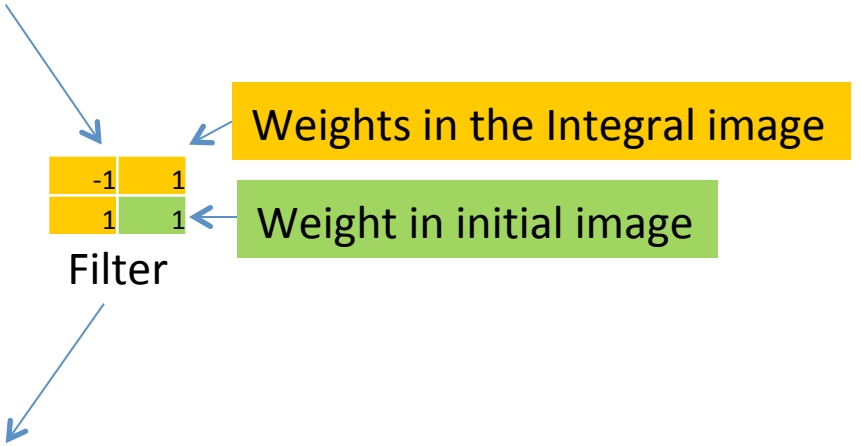
Building the integral image

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

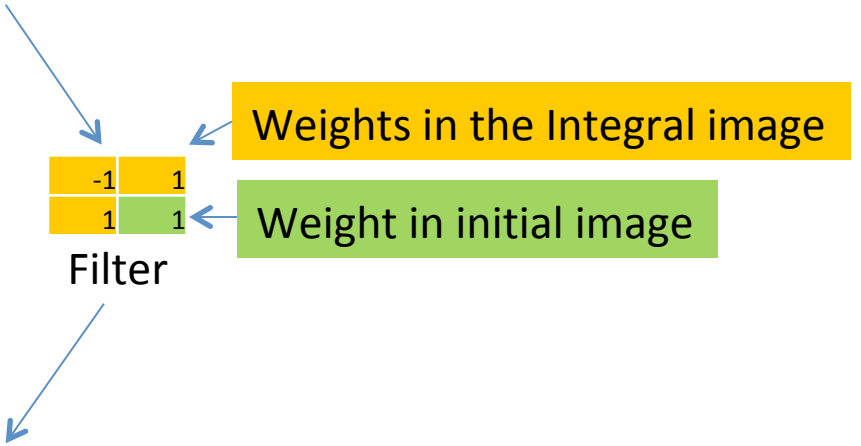
Integral image



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



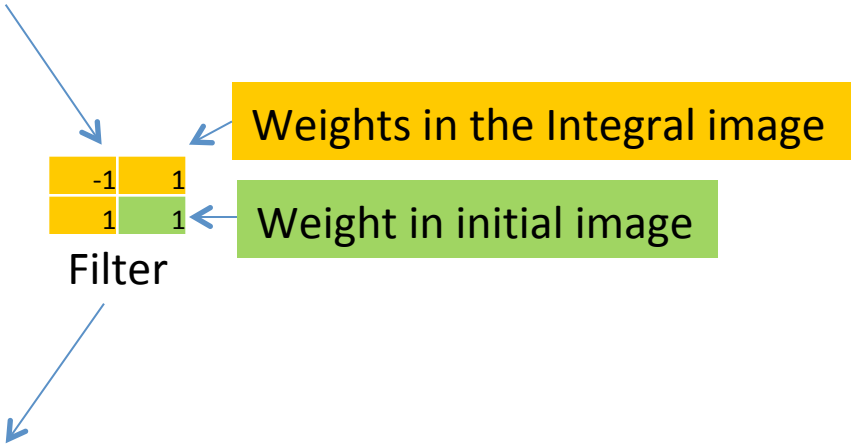
34	45	78

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



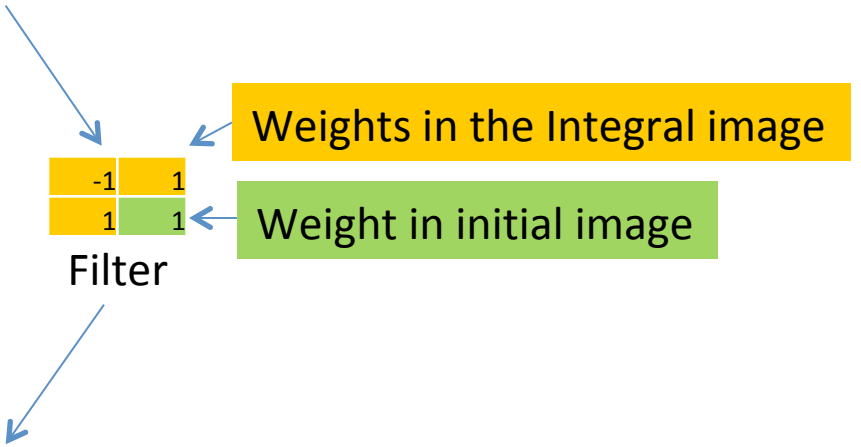
34	45	78	81	100	119	137	155	157	181

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

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

	Sum	Cost	Total Cost
ΣΣ	69	5	5
II	69	3	3

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

ΣΣ: 17+4+6+15+22+5= 69

II: 313+87-194-137= 69

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

	Sum	Cost	Total Cost
ΣΣ	352	23	28
II	352	3	6

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

II: 911+156-247-468= 352

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

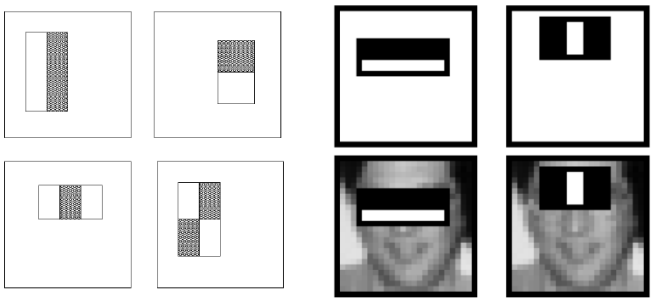
	Sum	Cost	Total Cost
ΣΣ	141	8	36
II	141	1	7

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

II: 762-621= 141

Viola-Jones face detector: features

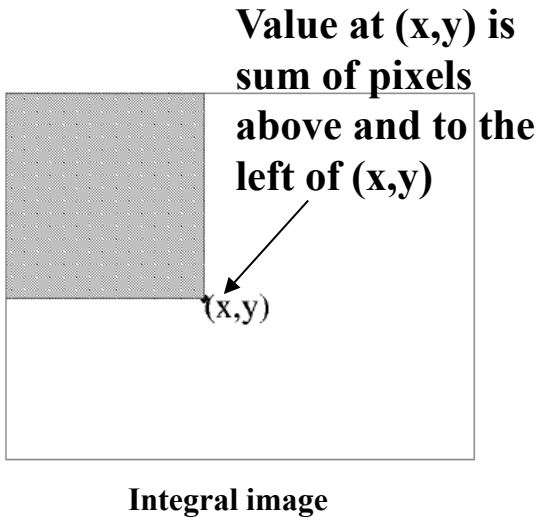


"**Rectangular" filters**

Feature output is difference between adjacent regions

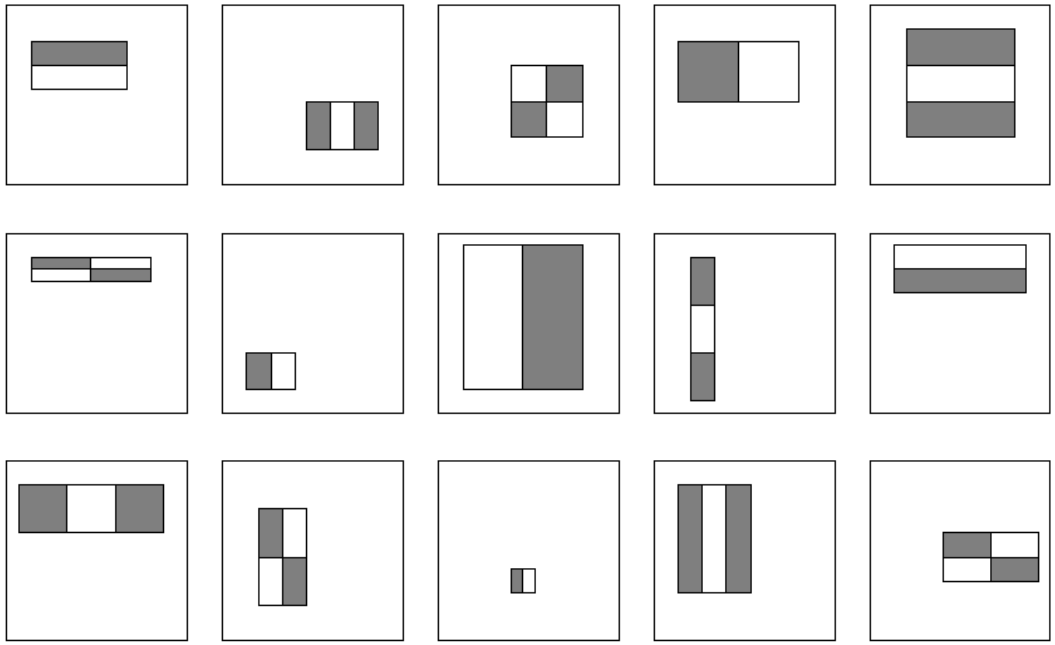
Efficiently computable with integral image: any sum can be computed in constant time

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

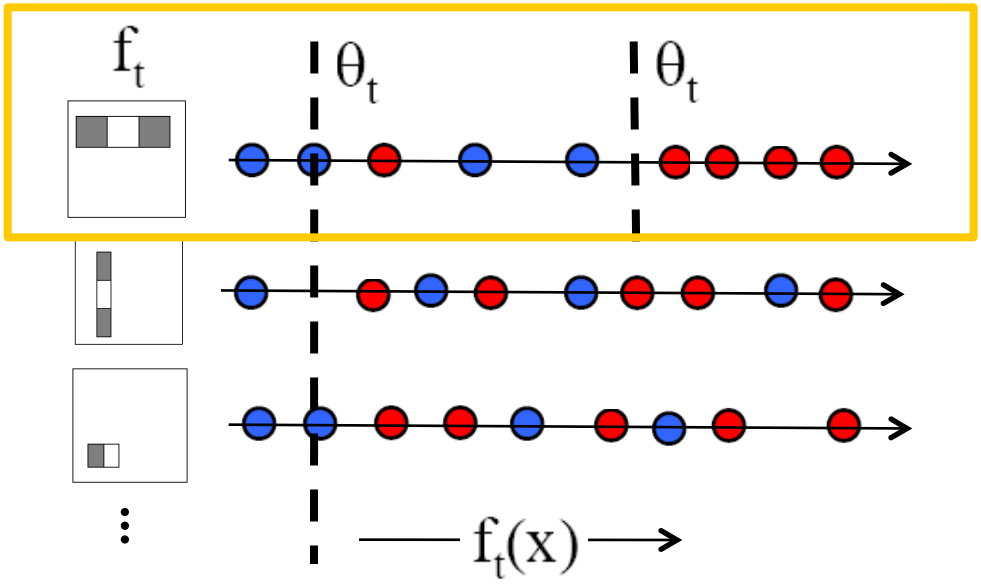
Training data

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:

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

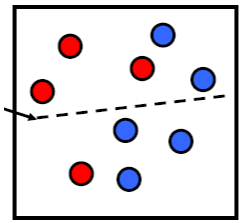
For next round, reweight the examples according to errors, choose another filter/threshold combo.

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

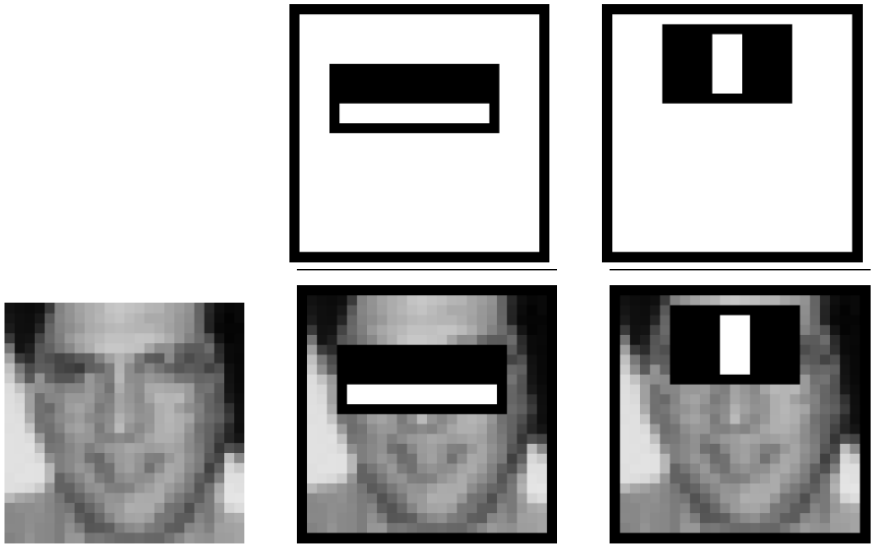
Evaluate *weighted* error for each feature, pick best.

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



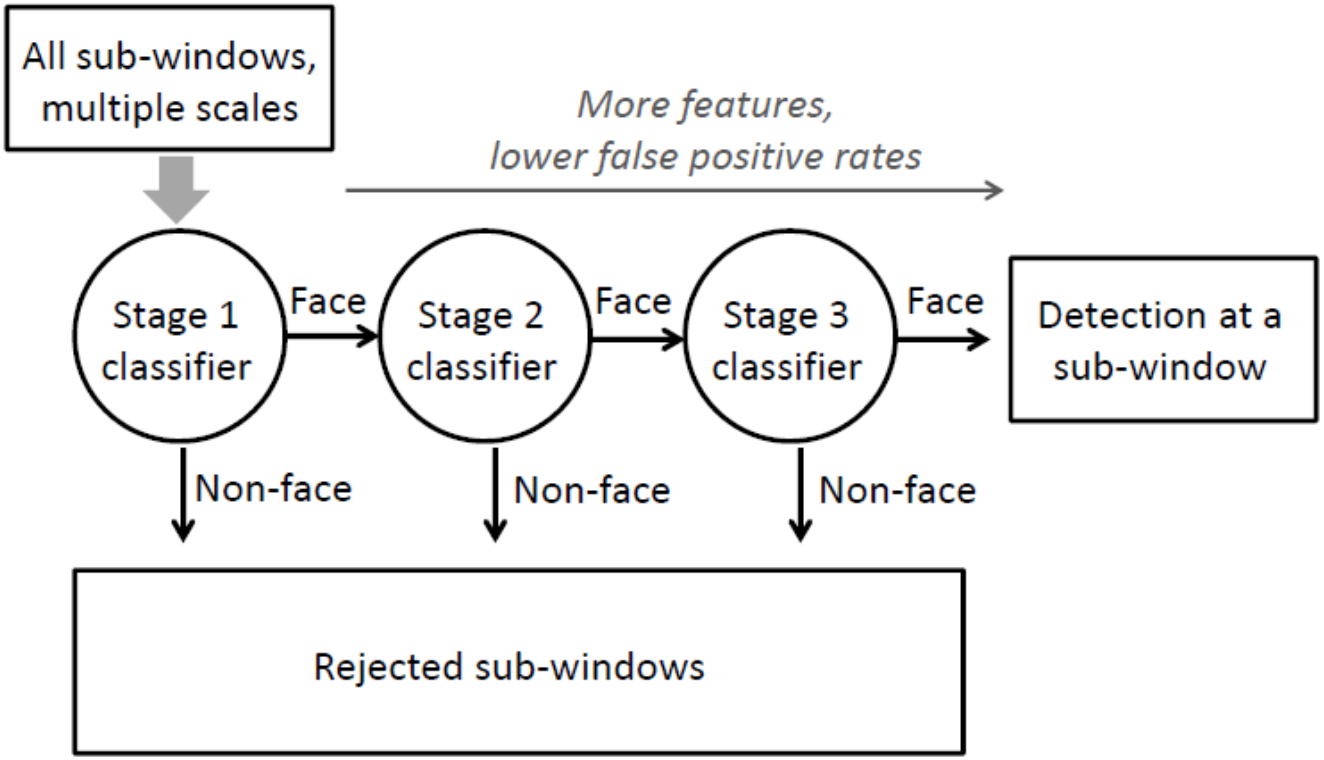
First two features selected

Faster?

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

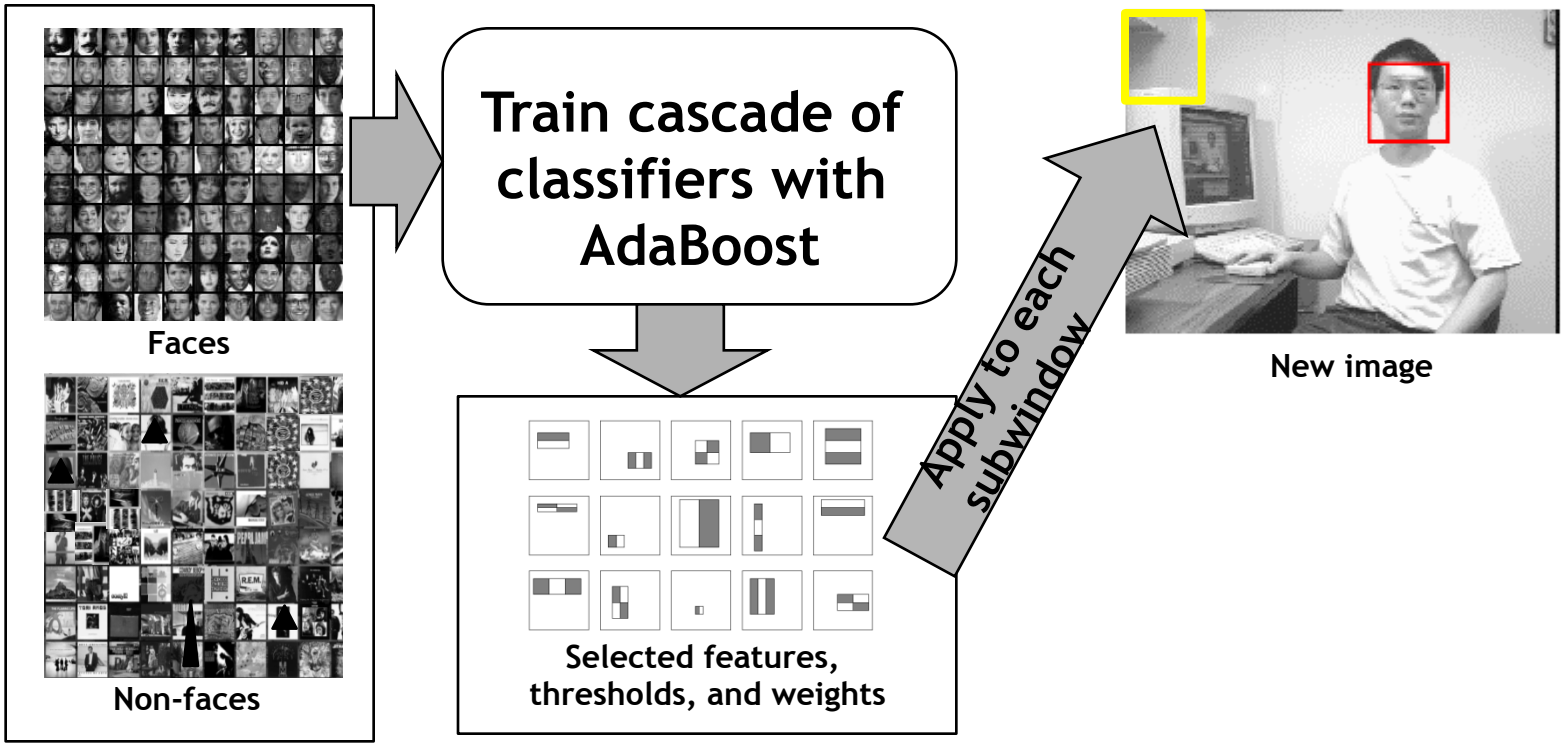
- How to make the detection more efficient?

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

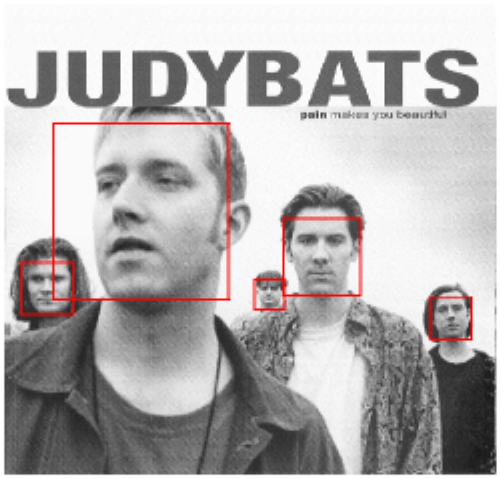
Viola-Jones detector: summary

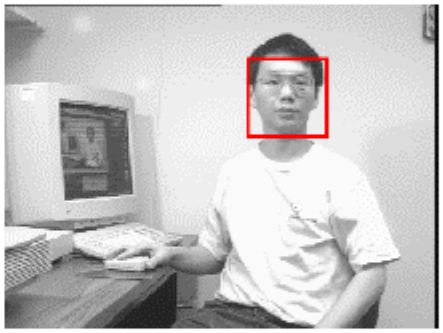
- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* of classifiers for fast rejection of non-face windows

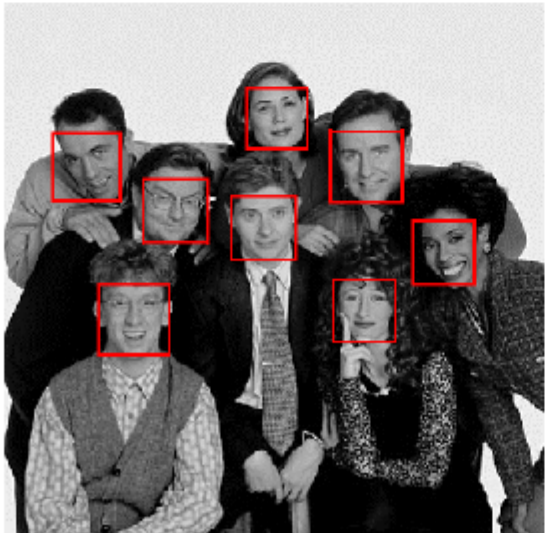
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

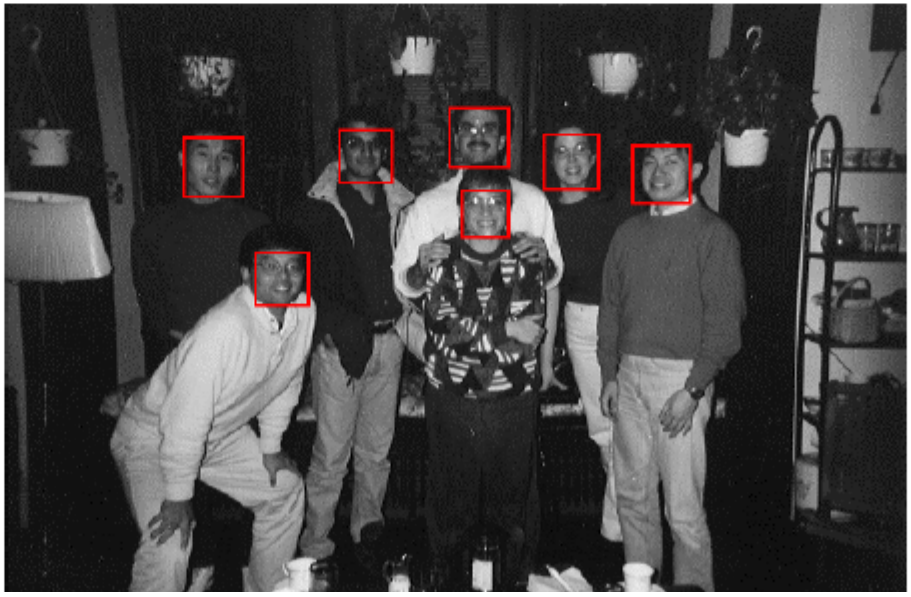
P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Viola-Jones Face Detector: Results

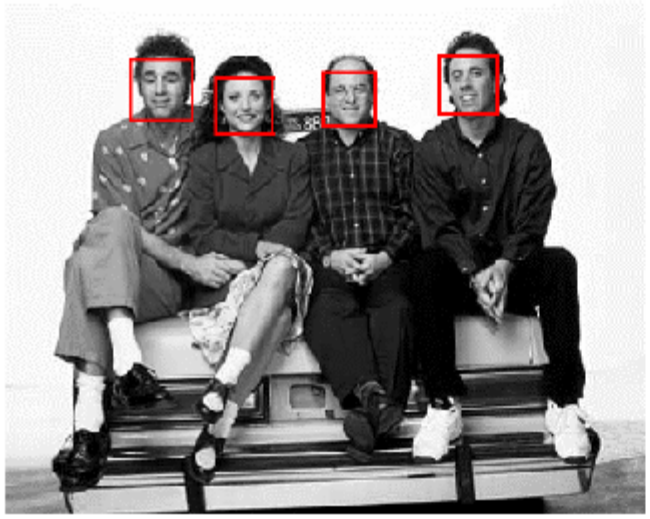


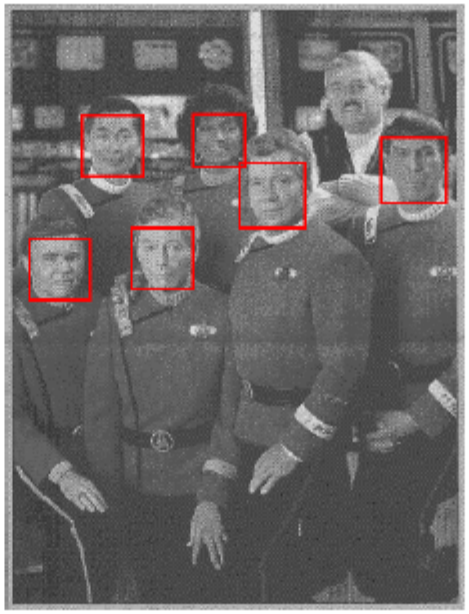


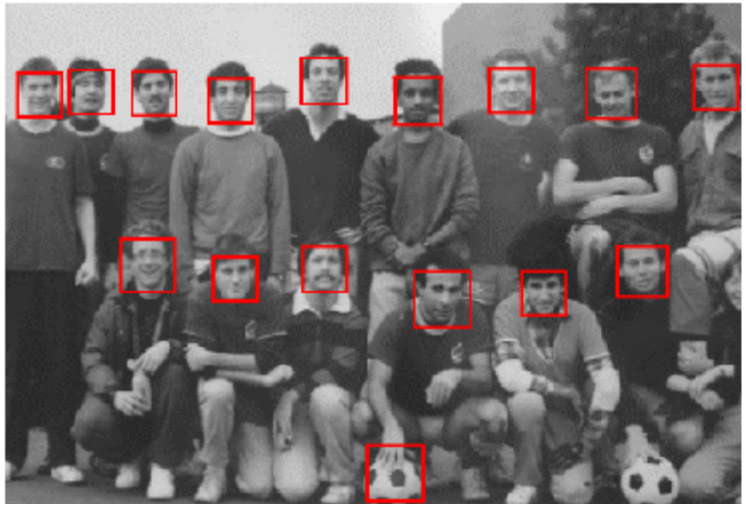


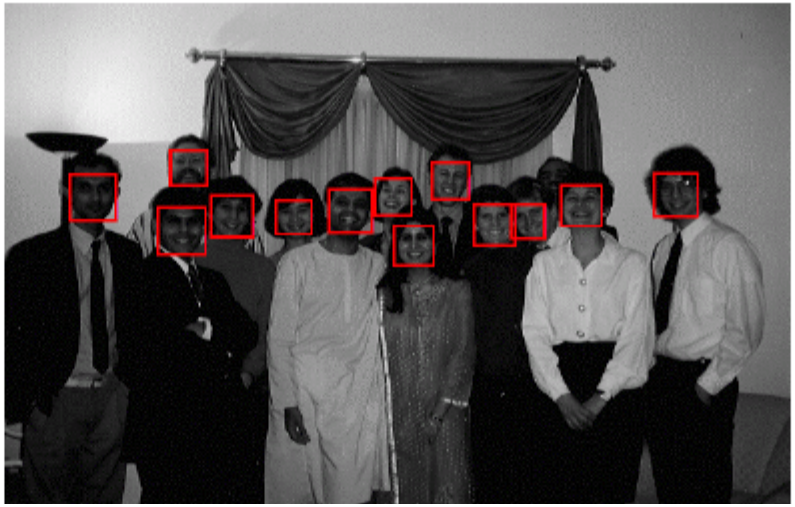


Viola-Jones Face Detector: Results





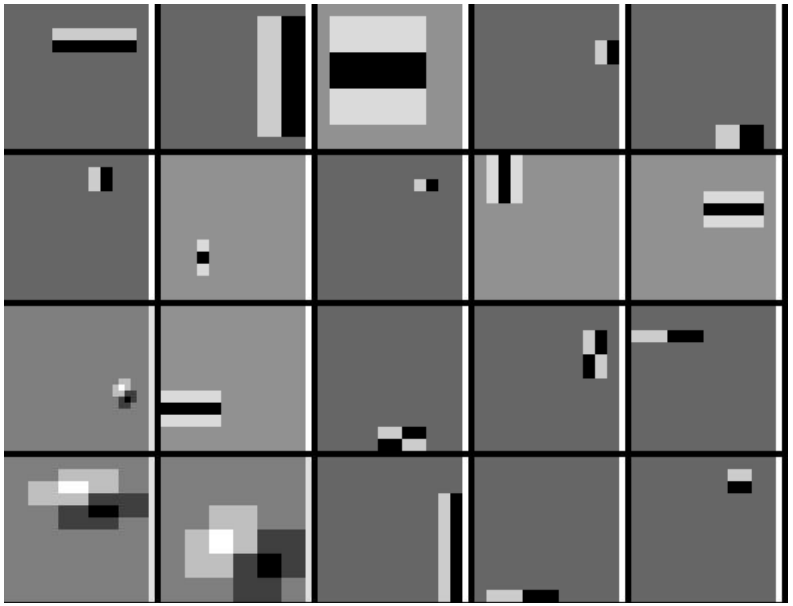




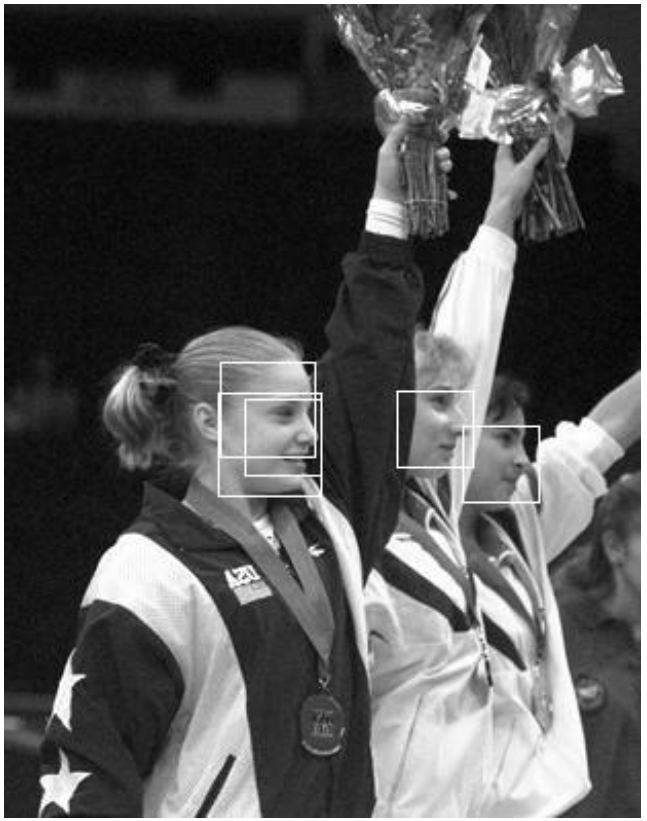
Viola-Jones Face Detector: Results

Detecting profile faces?

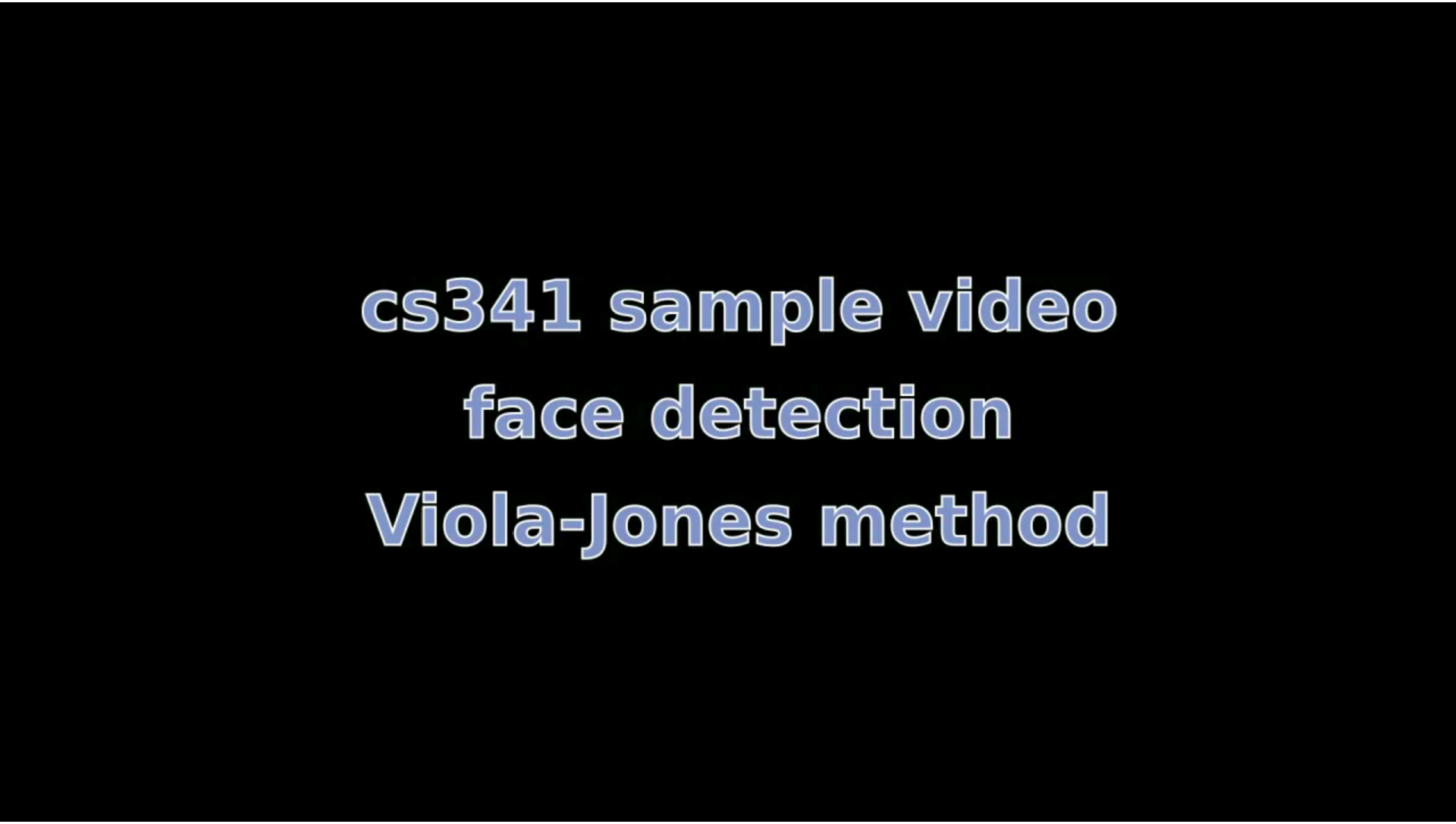
Can we use the same detector?



Viola-Jones Face Detector: Results



Detectorul facial Viola-Jones: rezultate



Support Vector Machines (SVMs)

SVMs

SVM Definition

- A **Support Vector Machine** (**SVM**) is a *non-probabilistic binary linear classifier*.
  - **Classifier** – A supervised learning method which predicts a categorical class.
  - **Linear** – The decision boundary is an n-dimensional hyperplane.
  - **Binary** – It can learn to predict one of two classes (a "+" class and a "-" class).
  - **Non-probabilistic** – The output of its *decision function* is not bounded, so it cannot be interpreted as probability.

There are methods of adapting an SVM to be used for **regression** problems and for making it **non-linear**, **multiclass** and/or **probabilistic**.

- An SVM tries to find a *separating* hyperplane, which is as far away from all training points at the same time (a **maximum-margin hyperplane**)
  - A point is classified as "+" or "-", depending on which part of the hyperplane it lies.

Slide credit Sparktech

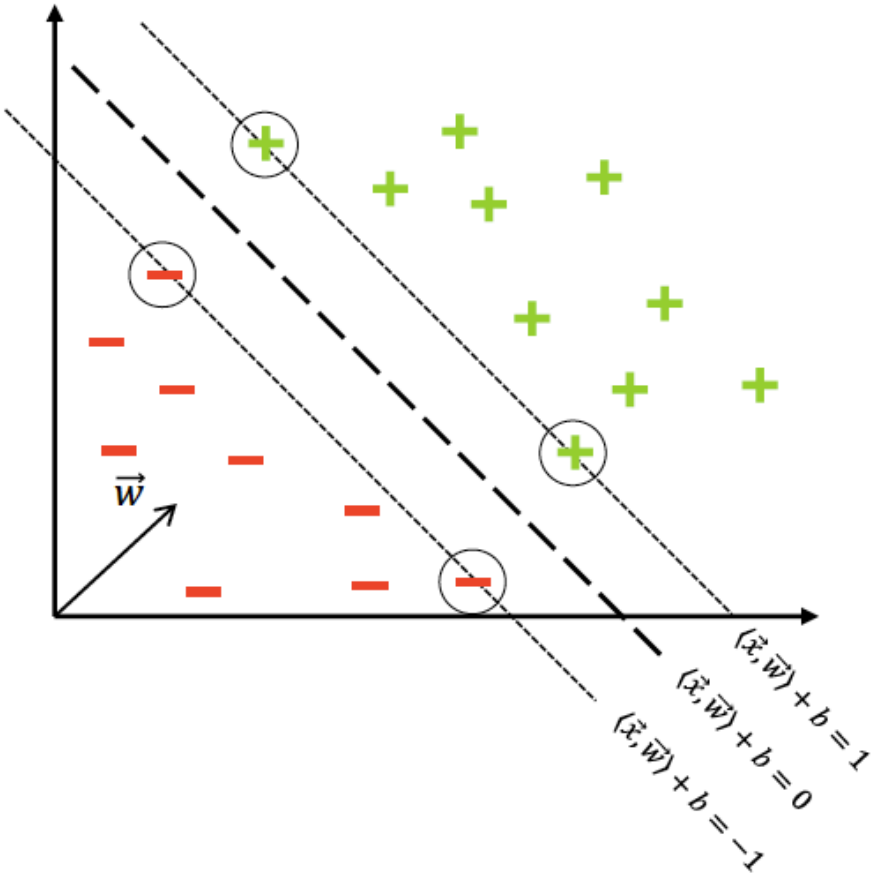
SVMs

Learning a "good" separating hyperplane

- For training examples $(\vec{x}^{(i)}, y^{(i)})$, which lie exactly on the edges of the gap:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 = 0$$

- We call these examples "**Support Vectors**"



Slide credit Sparktech

SVMs

Making the margin "wide"

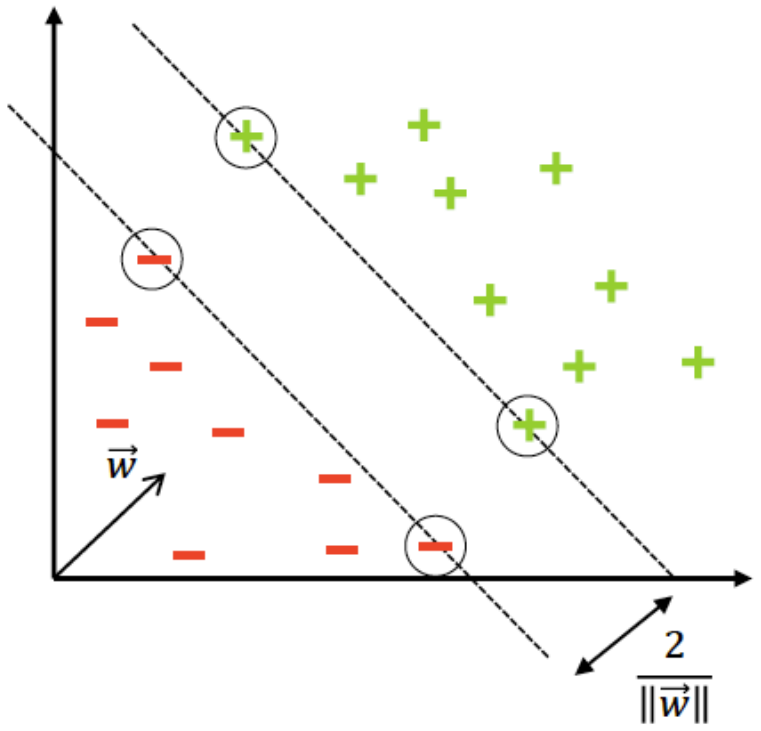
- How do we express the width of the gap?

$$g = \frac{2}{\|\vec{w}\|}$$

- We want to *maximize the gap:*

$$\text{maximize} \frac{2}{\|\vec{w}\|} \Rightarrow \text{minimize } \|\vec{w}\| \Rightarrow$$

$$\textbf{minimize} \frac{\|\vec{w}\|^2}{2}$$



Slide credit Sparktech

SVMs

What we have so far

SVM Primal Form

- The decision rule is:

$$\vec{x} \text{ is a "+" sample if } \langle \vec{x}, \vec{w} \rangle + \mathbf{b} \geq \mathbf{0}$$

- In order to obtain $\vec{w}$ and b we need to:

$$\text{minimize } \frac{\|\vec{w}\|^2}{2}$$

$$\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$$

Slide credit Sparktech

15.1.2 The Sample Complexity of Hard-SVM

Recall that the VC-dimension of halfspaces in $\mathbb{R}^d$ is $d+1$. It follows that the sample complexity of learning halfspaces grows with the dimensionality of the problem. Furthermore, the fundamental theorem of learning tells us that if the number of examples is significantly smaller than d/ϵ then no algorithm can learn an ϵ-accurate halfspace. This is problematic when d is very large.

To overcome this problem, we will make an additional assumption on the underlying data distribution. In particular, we will define a "separability with margin γ" assumption and will show that if the data is separable with margin γ then the sample complexity is bounded from above by a function of $1/\gamma^2$. It follows that even if the dimensionality is very large (or even infinite), as long as the data adheres to the separability with margin assumption we can still have a small sample complexity. There is no contradiction to the lower bound given in the fundamental theorem of learning because we are now making an additional assumption on the underlying data distribution.