

Object Detection by Adaboost Classifier Boosting

Tae-Kyun (T-K) Kim
Senior Lecturer
<https://labicvl.github.io/>

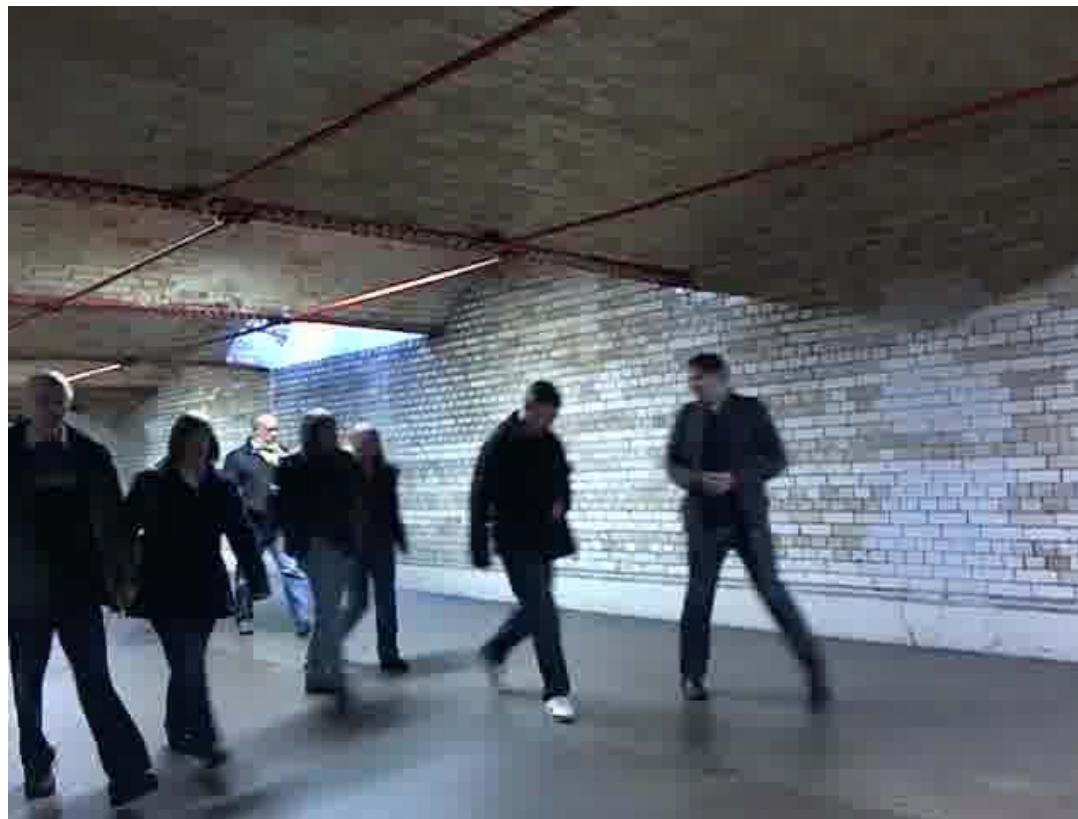
Further reading:

Viola and Jones, Robust real-time object detector, 2004.
<http://www.iis.ee.ic.ac.uk/tkkim/tmp/viola04ijcv.pdf>

Chapter 14, C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

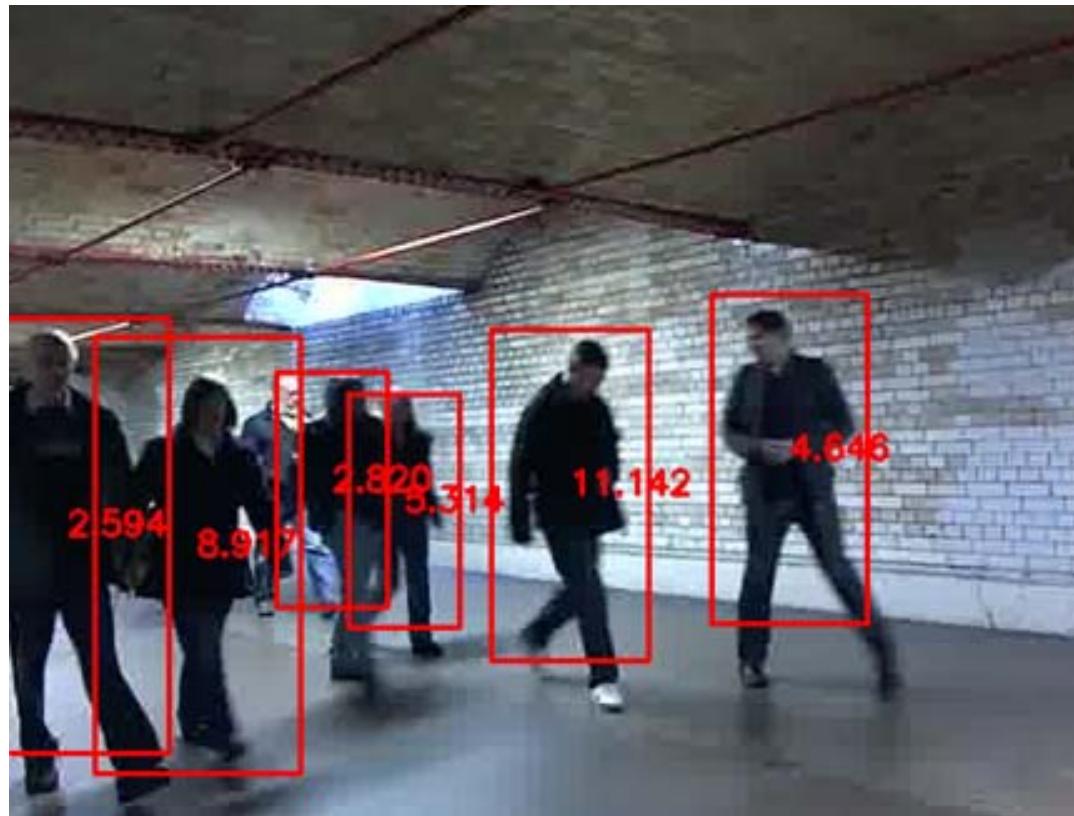
Object detection

- A single image is given as input, without prior knowledge (except a known target object class, e.g. pedestrian).



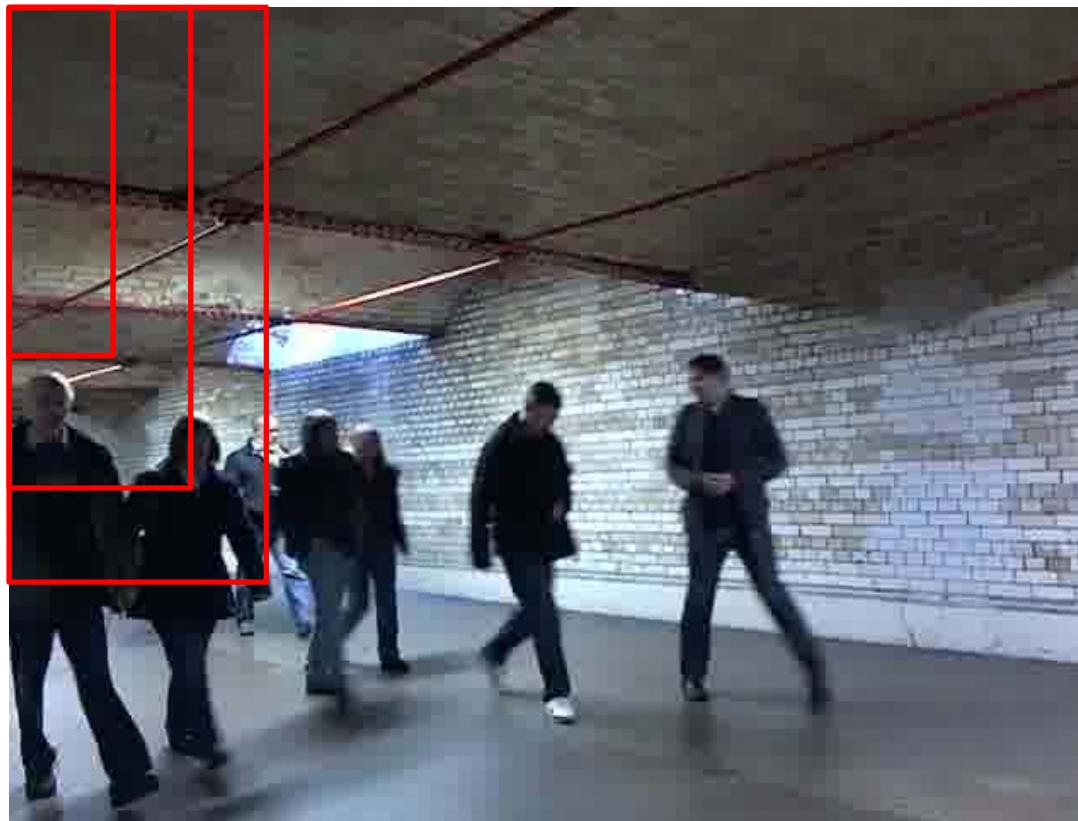
Object detection

- A single image is given as input, without prior knowledge (except a known target object class, e.g. pedestrian).
- Output is a set of tight bounding boxes (positions and scales) of instances of the target object class, optionally with confidence scores.



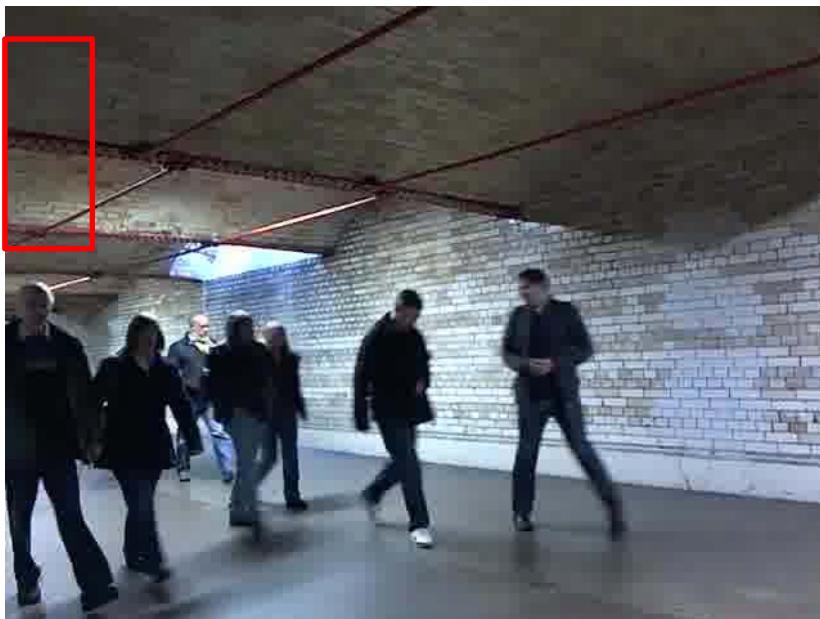
Number of hypotheses (scanning windows)

- We scan every scale and every pixel location in the given image.



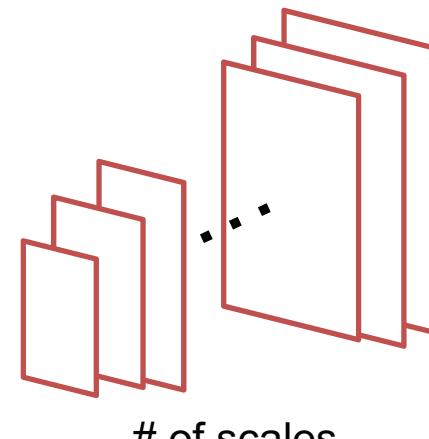
Number of hypotheses (scanning windows)

- We scan every scale and every pixel location in the given image.
- We end up with a huge number of candidate windows i.e. detection hypotheses.



of pixels

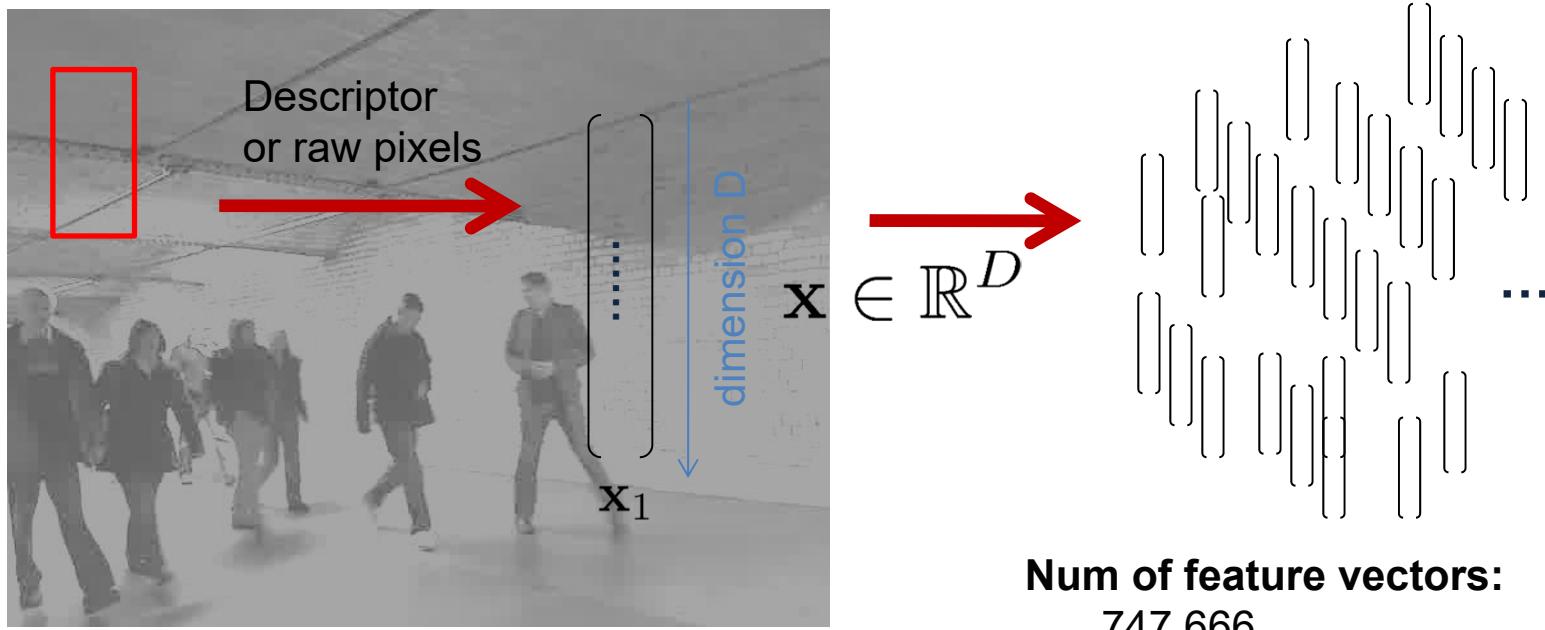
X



→ **Number of windows:** e.g. 747,666

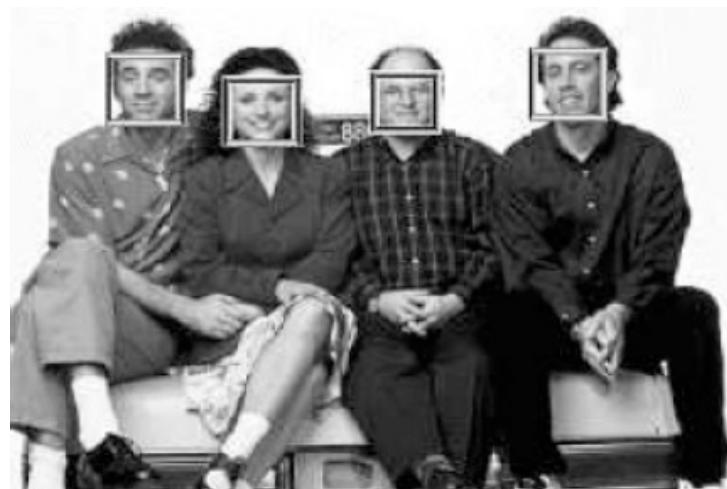
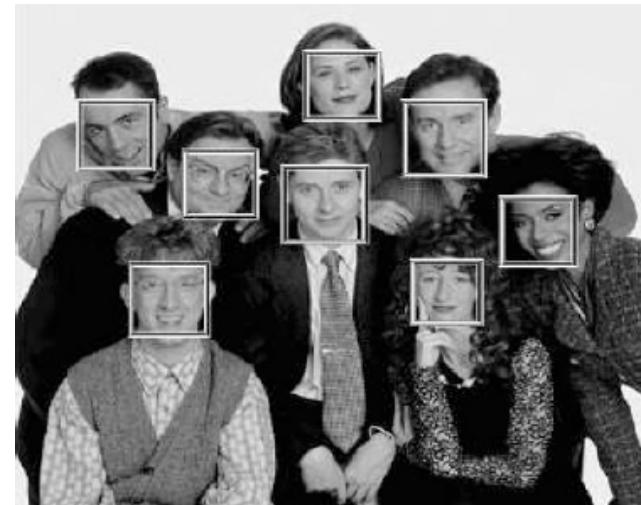
Number of hypotheses (scanning windows)

- Little amount of time is given per a scanning window.
- It requires an extremely fast yet reliable feature/classifier.



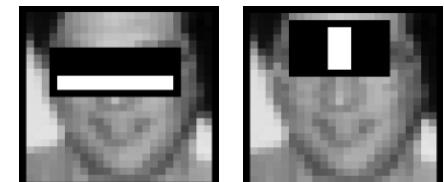
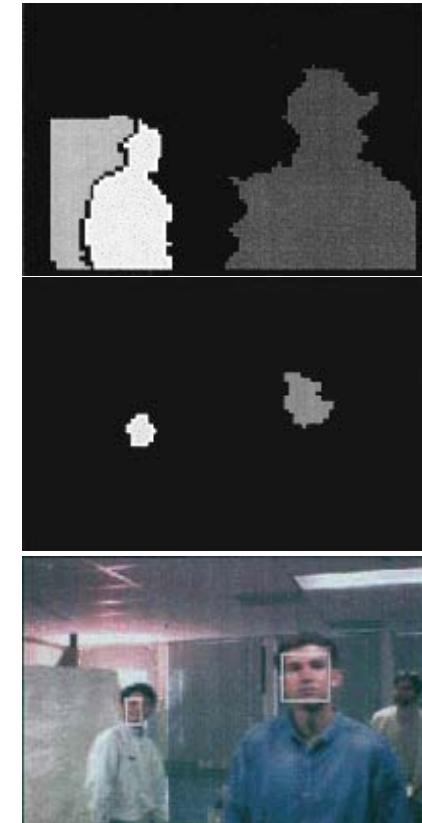
$$\xrightarrow{f: \mathbf{x} \rightarrow t} \xrightarrow{\text{Time per window (or vector): } 0.00000134 \text{ sec}} \begin{array}{l} \text{Binary classification: } t \in \{-1, 1\} \\ 6 \end{array}$$

Examples of face detection



A brief history for face detection

- Earlier approaches are knowledge-based, feature-based, or template-based.
- More contemporary methods are appearance-based i.e. learning a classifier from examples: e.g.
 - PCA
 - Distribution-based [Sung and Poggio, 1994]
 - Multi-Layer Perceptron [Rowley, Kanade, 1998]
 - Support Vector Machine [Osuna et al., 1997]
- To accelerate speed, the search space is narrowed down by integrating visual cues [Darrell et al, 2000]:
 - Connected pixels from **stereo** depth data (top).
 - **Skin colour (hue)** regions (middle).
 - Face pattern detection output (bottom).
- Since Viola & Jones 2001, **boosting** simple features has been a dominating art:
 - Adaboost classification
 - Weak classifiers: rectangle filter responses



Introduction to boosting classifiers

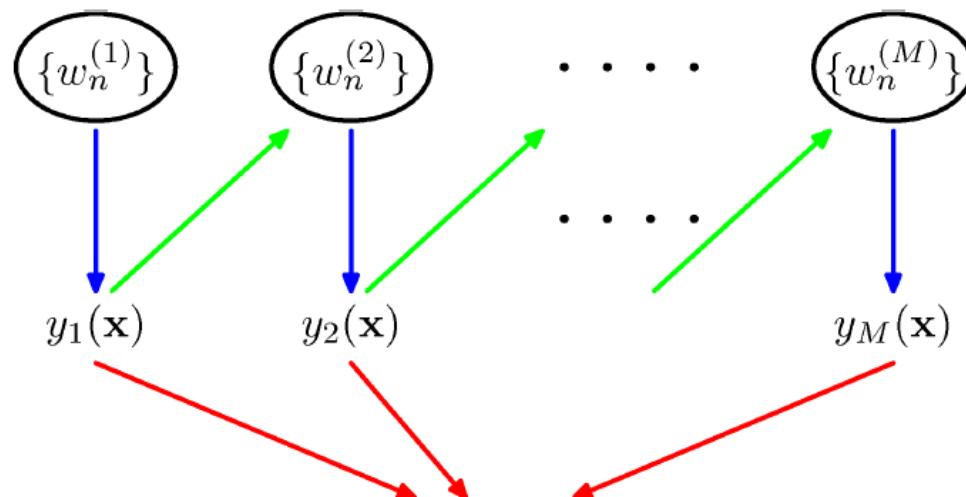
AdaBoost (adaptive boosting)

- Boosting gives good results even if the base classifiers have a performance slightly better than *random* guessing.
- Hence, the base classifiers are called *weak classifiers* or *weak learners*.

Boosting

For a two (binary)-class classification problem, we train with

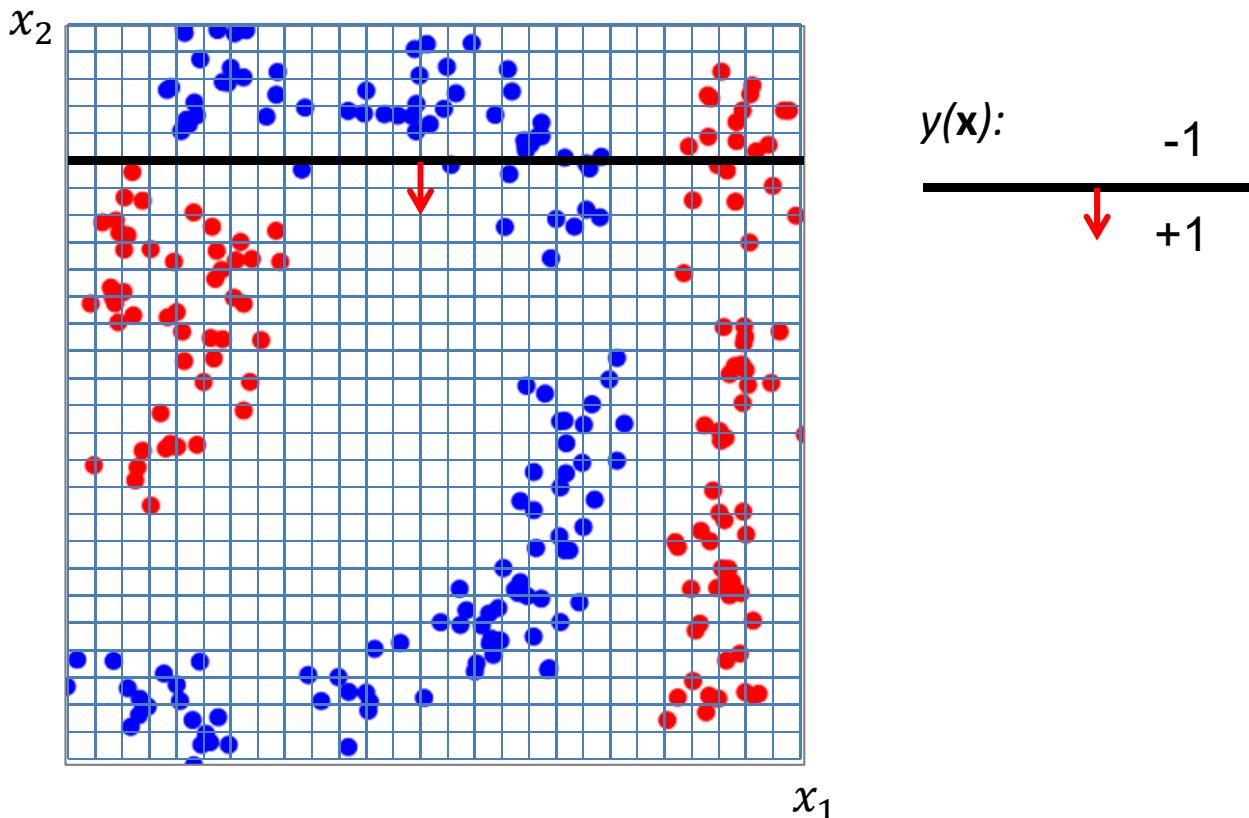
- training data $\mathbf{x}_1, \dots, \mathbf{x}_N$
- target variables t_1, \dots, t_N , where $t_N \in \{-1, 1\}$,
- data weight w_1, \dots, w_N
- weak (base) classifier candidates $y(\mathbf{x}) \in \{-1, 1\}$.



$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_m^M \alpha_m y_m(\mathbf{x}) \right)$$

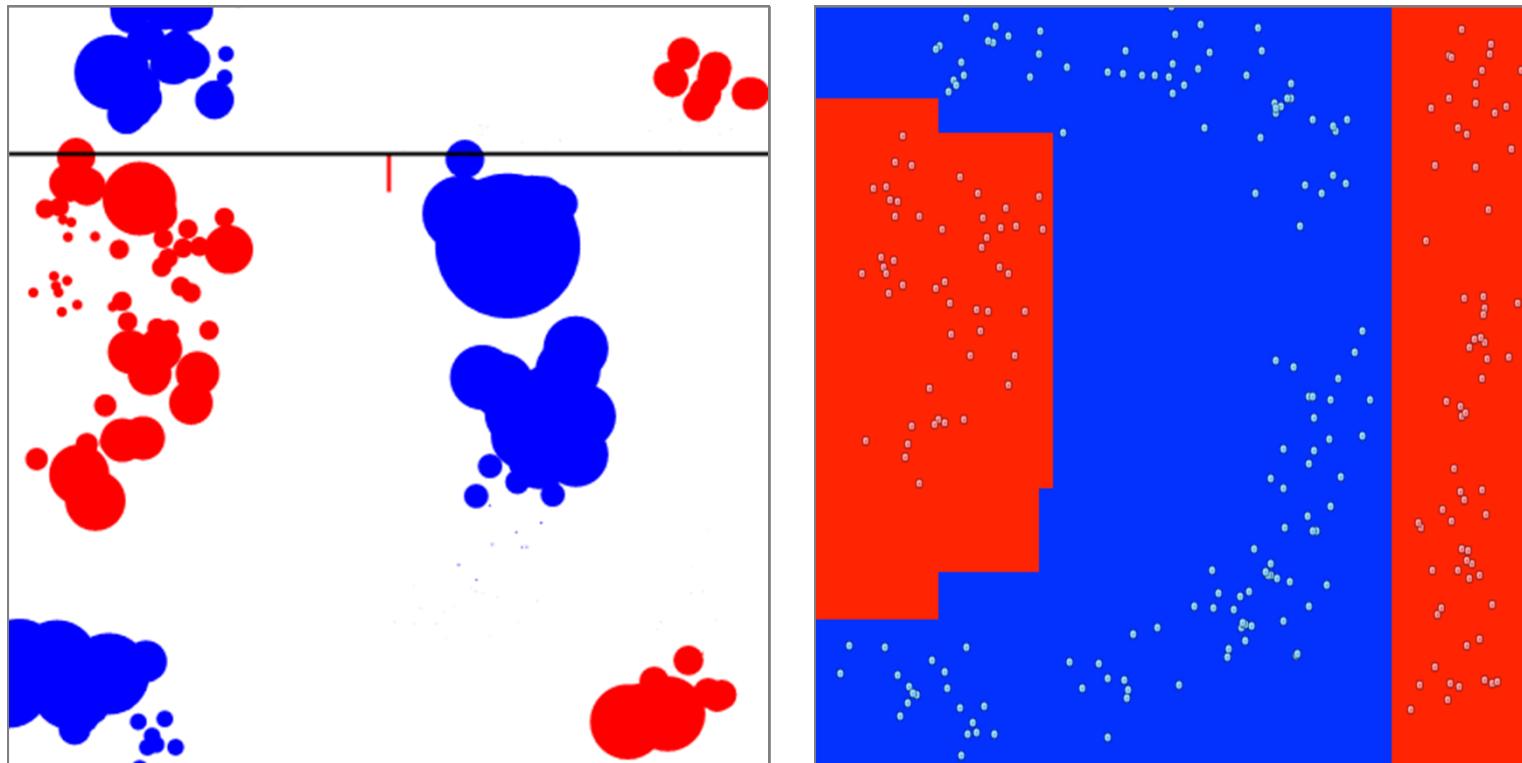
Boosting

- In the example below, the weak learner $y(\mathbf{x}) \in \{-1, 1\}$ is defined as axis-aligned e.g. $[x_i < \tau]$, for any $i \in \{1, \dots, d\}$, $\mathbf{x} = [x_1, x_2, \dots, x_d]$.



Boosting

- The algorithm iteratively does
 - 1) reweighting training samples, by assigning higher weights to previously misclassified samples,
 - 2) finding the best weak classifier for the weighted samples.
- Complex nonlinear classification problems are solved by a set of simple weak learners.



AdaBoost (adaptive boosting)

Initialise the data weights $\{w_n\}$ by $w_n^{(1)} = 1/N$ for $n = 1, \dots, N$.

For $m = 1, \dots, M$: the number of weak classifiers to choose

(a) Learn a classifier $y_m(\mathbf{x})$ that minimises the weighted error, among all weak classifier candidates

$$J_m = \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

where I is the impulse function.

(b) Evaluate

$$\epsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^N w_n^{(m)}}$$

AdaBoost (adaptive boosting)

and set

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}$$

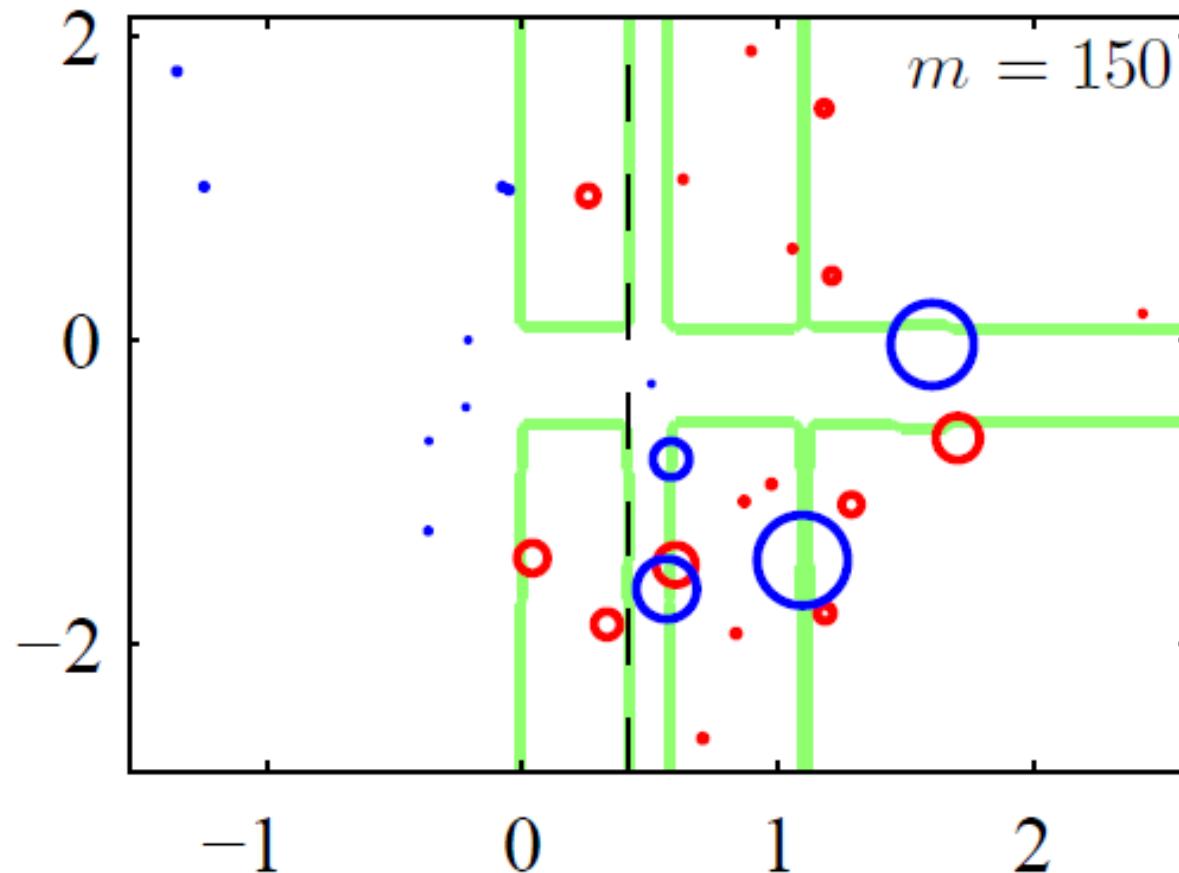
(c) Update the data weights

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\}$$

Make predictions using the final model by

$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \right)$$

AdaBoost (adaptive boosting)



Boosting as an optimisation framework

- AdaBoost is the sequential minimisation of the exponential error function

$$E = \sum_{n=1}^N \exp\{-t_n f_m(\mathbf{x}_n)\}$$

where $t_n \in \{-1, 1\}$ and $f_m(\mathbf{x})$ is a classifier as a linear combination of base classifiers $y_l(\mathbf{x})$

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^m \alpha_l y_l(\mathbf{x})$$

- We minimise E with respect to the weight α_l and the parameters of the base classifiers $y_l(\mathbf{x})$.

Boosting as an optimisation framework

- Sequential Minimisation: suppose that the base classifiers $y_1(\mathbf{x}), \dots, y_{m-1}(\mathbf{x})$ and their coefficients $\alpha_1, \dots, \alpha_{m-1}$ are fixed, and we minimise only w.r.t. α_m and $y_m(\mathbf{x})$.
- The error function is rewritten by

$$\begin{aligned} E &= \sum_{n=1}^N \exp\{-t_n f_m(\mathbf{x}_n)\} \\ &= \sum_{n=1}^N \exp \left\{ -t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\} \\ &= \sum_{n=1}^N w_n^{(m)} \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\} \end{aligned}$$

where $w_n^{(m)} = \exp\{-t_n f_{m-1}(\mathbf{x}_n)\}$ are constants.

Boosting as an optimisation framework

- Denote the set of data points correctly classified by $y_m(\mathbf{x}_n)$ by T_m , and those misclassified M_m , then

$$E = e^{-\alpha_m/2} \sum_{n \in T_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in M_m} w_n^{(m)}$$

$$= \left(e^{\alpha_m/2} - e^{-\alpha_m/2} \right) \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^N w_n^{(m)}$$

- When we minimise w.r.t. $y_m(\mathbf{x}_n)$, the second term is constant and minimising E is equivalent to

$$J_m = \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

Boosting as an optimisation framework

- By setting the derivative w.r.t. α_m to 0, we obtain $\alpha_m = \ln \left\{ \frac{1-\epsilon_m}{\epsilon_m} \right\}$

where $\epsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^N w_n^{(m)}}$

- From $E = \sum_{n=1}^N w_n^{(m)} \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\}$

→ $w_n^{(m+1)} = w_n^{(m)} \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\}$

- As $t_n y_m(\mathbf{x}_n) = 1 - 2I(y_m(\mathbf{x}_n) \neq t_n)$

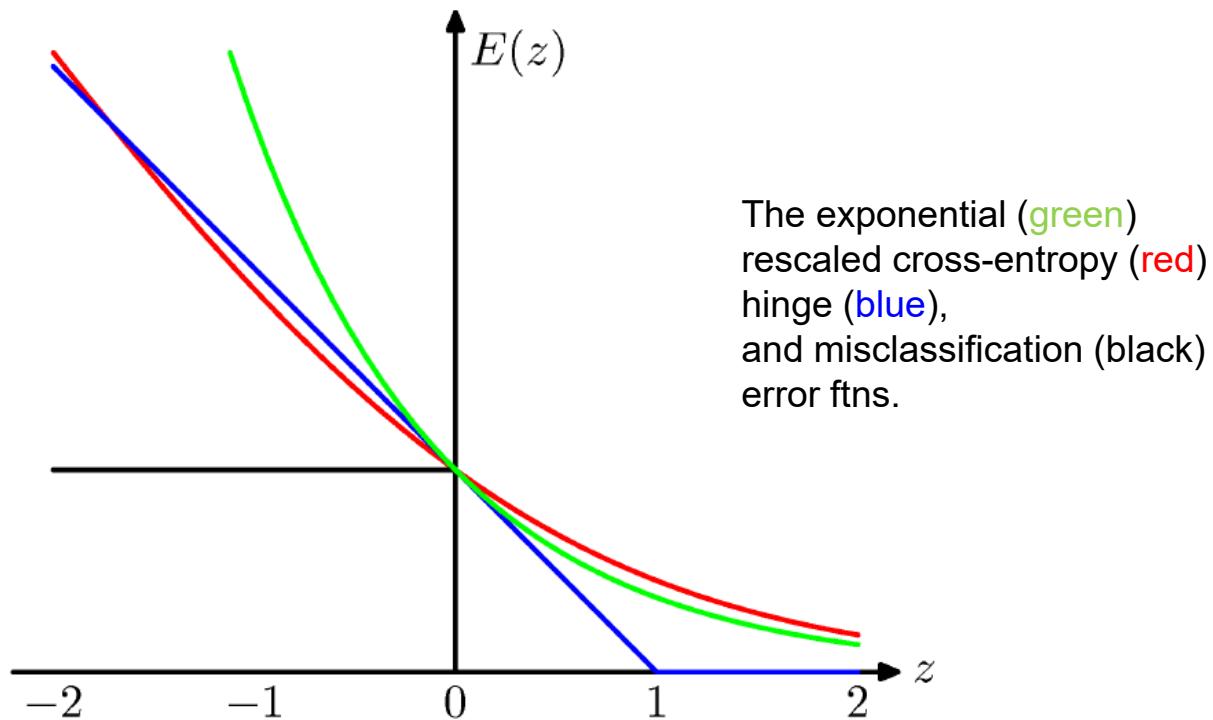
$$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m/2) \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\}$$

- The term $\exp(-\alpha_m/2)$ is independent of n , thus we obtain

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\}$$

Exponential error function

- Pros: it leads to simple derivations of Adaboost algorithms.
- Cons: it penalises large negative values. It is prone to outliers.



Robust real-time object detector

[Viola and Jones 2001]

- A face detection framework that is capable of processing images extremely rapidly while achieving high detection rates.
- Three key components:
 - The first is the introduction of a new image representation called the “**Integral Image**” which allows the features used by the detector to be computed very quickly.
 - The second is a simple and efficient classifier which is built using the **AdaBoost** learning algorithm (Freund and Schapire, 1995) to select a small number of critical visual features from a very large set of potential features.
 - The third contribution is a method for combining classifiers in a “**cascade**” which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions.

Robust real-time object detector

[Viola and Jones 2001]

- Adaboost classification

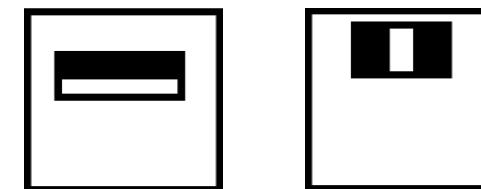
Strong classifier

$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \right)$$

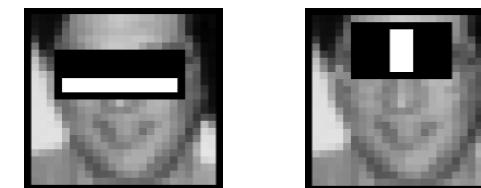
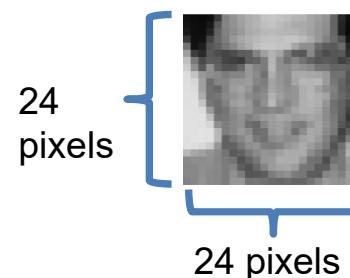
Weak classifier

- Weak classifiers: rectangle filter responses (160,000 **in total feature pool**)

$$y_m(\mathbf{x}) = \begin{cases} +1 & \text{if } h_m(\mathbf{x}) \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

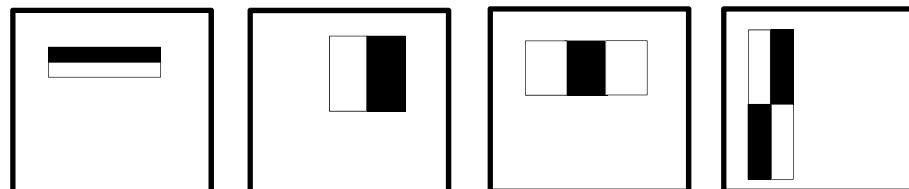


.....



Rectangle features

- Our face detection procedure classifies images based on the value of simple features.
- The simple features used are reminiscent of Haar basis functions which have been used by Papageorgiou et al. 1998.
- More specifically, we use following example of features.
 - The value of a *two-rectangle feature* is the difference between the sum of the pixels within two rectangular regions. The regions have the same size and shape and are horizontally or vertically adjacent.
 - A *three-rectangle feature* computes the sum within two outside rectangles subtracted from the sum in a center rectangle.
 - Finally a *four-rectangle feature* computes the difference between diagonal pairs of rectangles.
- Given that the base resolution of the detector is 24×24 , **the exhaustive set of rectangle features is large**, 160,000.

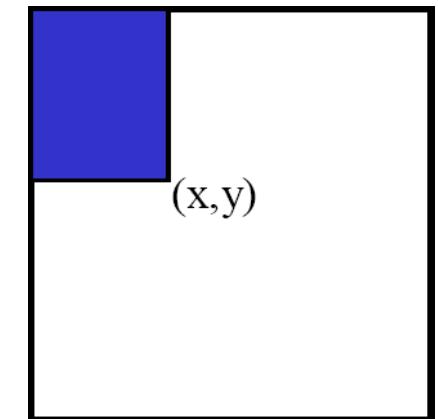


Integral image

- Rectangle features can be computed very rapidly using an intermediate representation for the image which we call the integral image.
- The integral image at location (x, y) contains the sum of the pixels above and to the left of (x, y) , inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image.



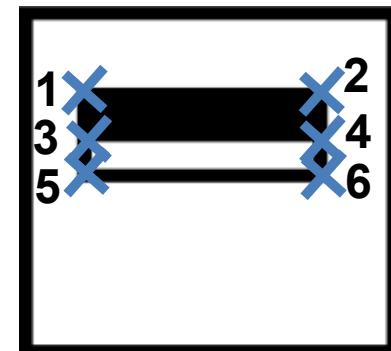
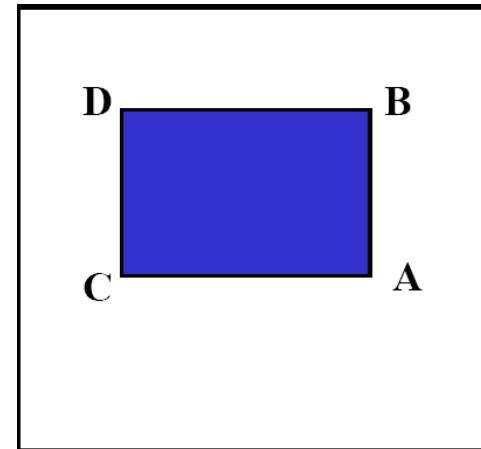
- Using the following pair of recurrences:

$$\begin{aligned}s(x, y) &= s(x, y - 1) + i(x, y) \\ ii(x, y) &= ii(x - 1, y) + s(x, y)\end{aligned}$$

where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$, and $ii(-1, y) = 0$, the integral image can be computed in one pass over the original image.

Rectangle features

- Using the integral image any rectangular sum can be computed in four array references:
 - The sum of original image values within the rectangle can be computed: $\text{Sum} = \text{ii}(A) - \text{ii}(B) - \text{ii}(C) + \text{ii}(D)$.
 - This provides the fast evaluation of rectangle-filter responses.
- Clearly the difference between two rectangular sums can be computed in eight references. Since the two-rectangle features defined above involve adjacent rectangular sums they can be computed in six array references.



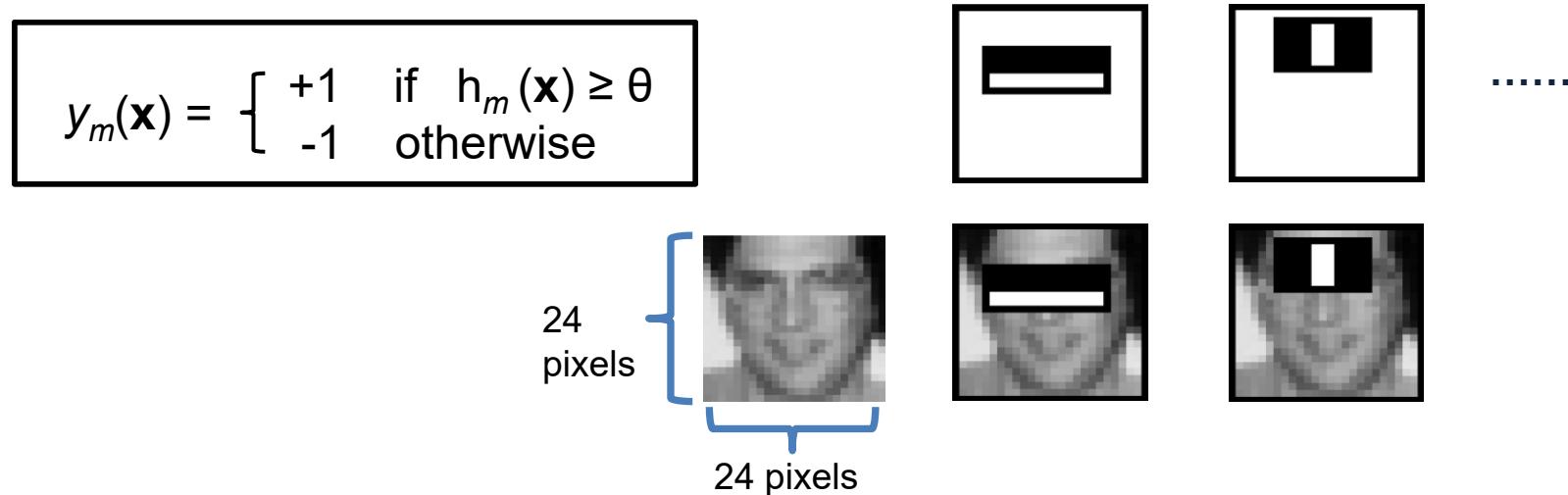
$$h_m(x) = (\text{ii}(6) - \text{ii}(4) - \text{ii}(5) + \text{ii}(3)) \\ - (\text{ii}(4) - \text{ii}(2) - \text{ii}(3) + \text{ii}(1))$$

AdaBoost Learning

- Given **a feature set and a training set of positive and negative images**, any machine learning approach could be used to learn a classification function.
- Recall that there are 160,000 rectangle features associated with each image sub-window.
- Even though each feature can be computed very efficiently, computing the complete set is prohibitively expensive.
- **The main challenge is to find a very small number of features** can be combined to form an effective classifier.
- Here AdaBoost is used both to select the features and to train the classifier (Freund and Schapire, 1995).
- AdaBoost combines a collection of weak classification functions to form a stronger classifier.

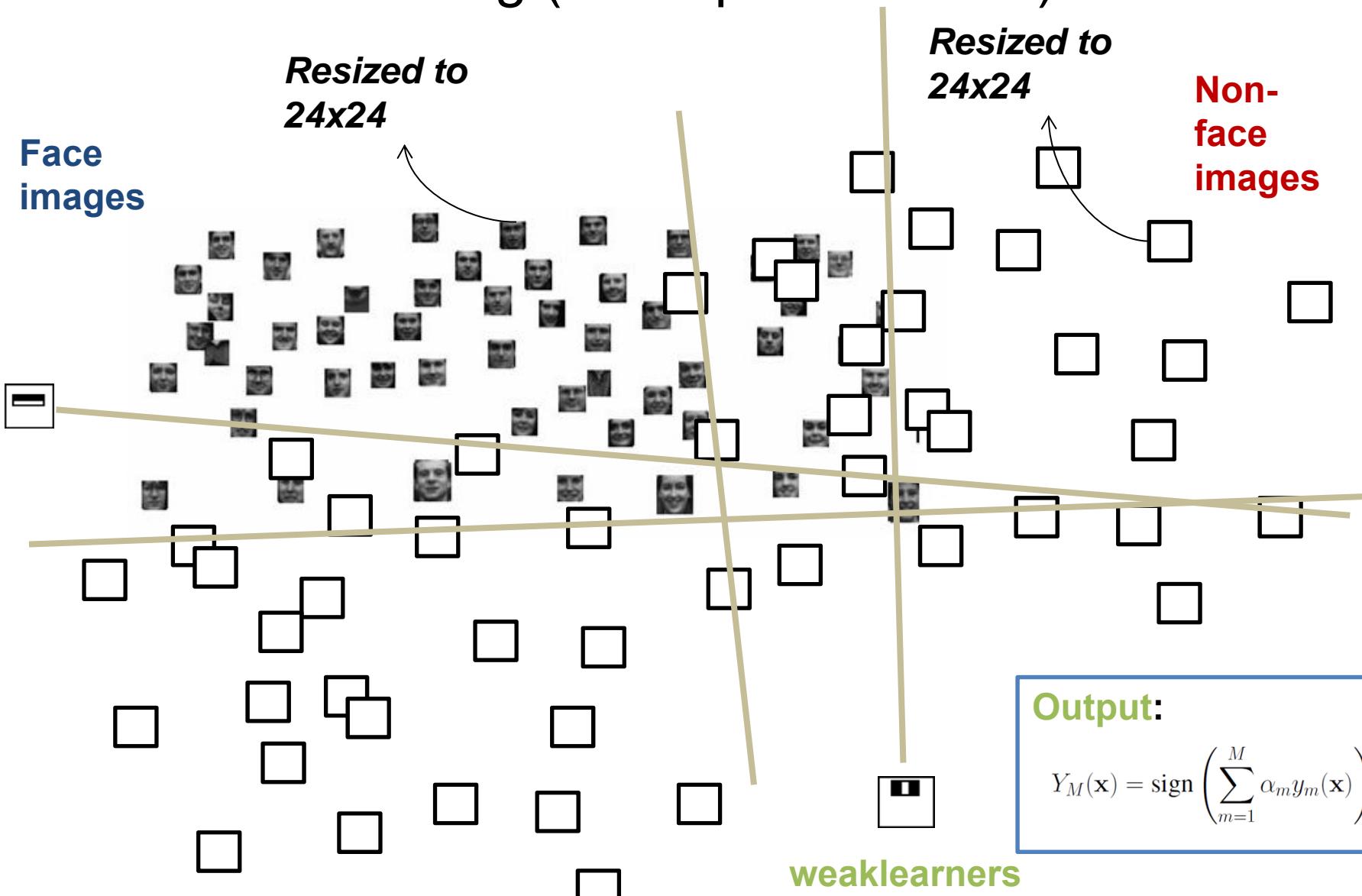
AdaBoost Learning

- One practical method is to restrict the weak learner to the classification function depends on a single feature.
- The weak learner $y_m(\mathbf{x})$ is designed to select the single rectangle feature which best separates the positive and negative examples.



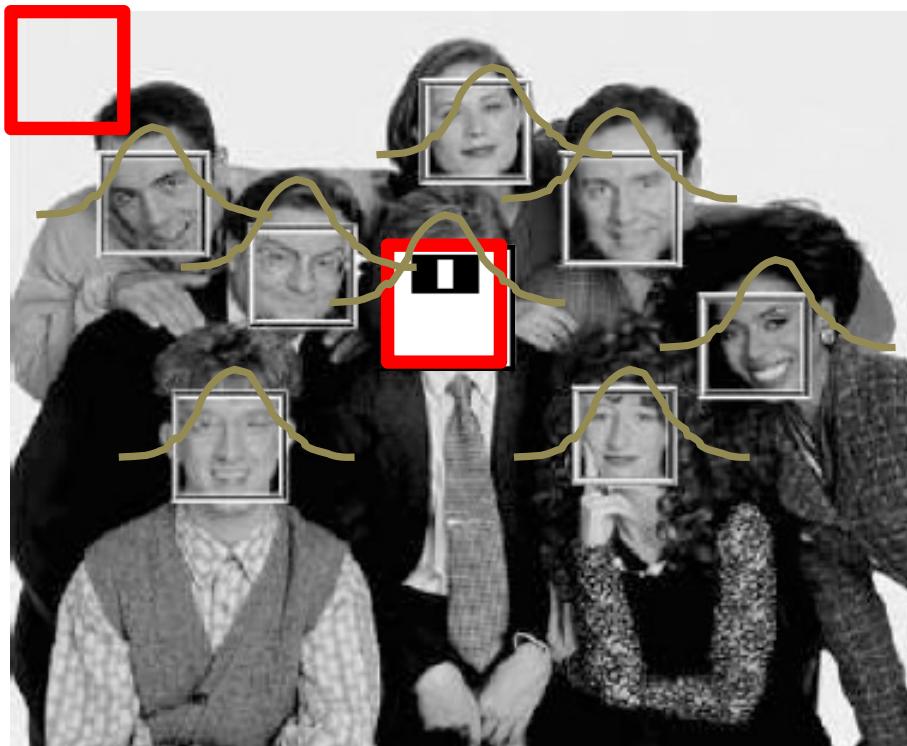
- Here h_m is a rectangle feature, and x is a 24×24 pixel sub-window of an image.
- The weak classifiers that we use (thresholded single features) can be viewed as single node decision trees. Such structures are called decision stumps.

Learning (concept illustration)



Evaluation (testing)

- The learnt boosting classifier i.e. $\sum_{m=1}^M \alpha_m y_m(\mathbf{x})$ is applied to every scan-window.
- The response map is obtained, then the non-local maxima suppression is performed to detect local maximas.



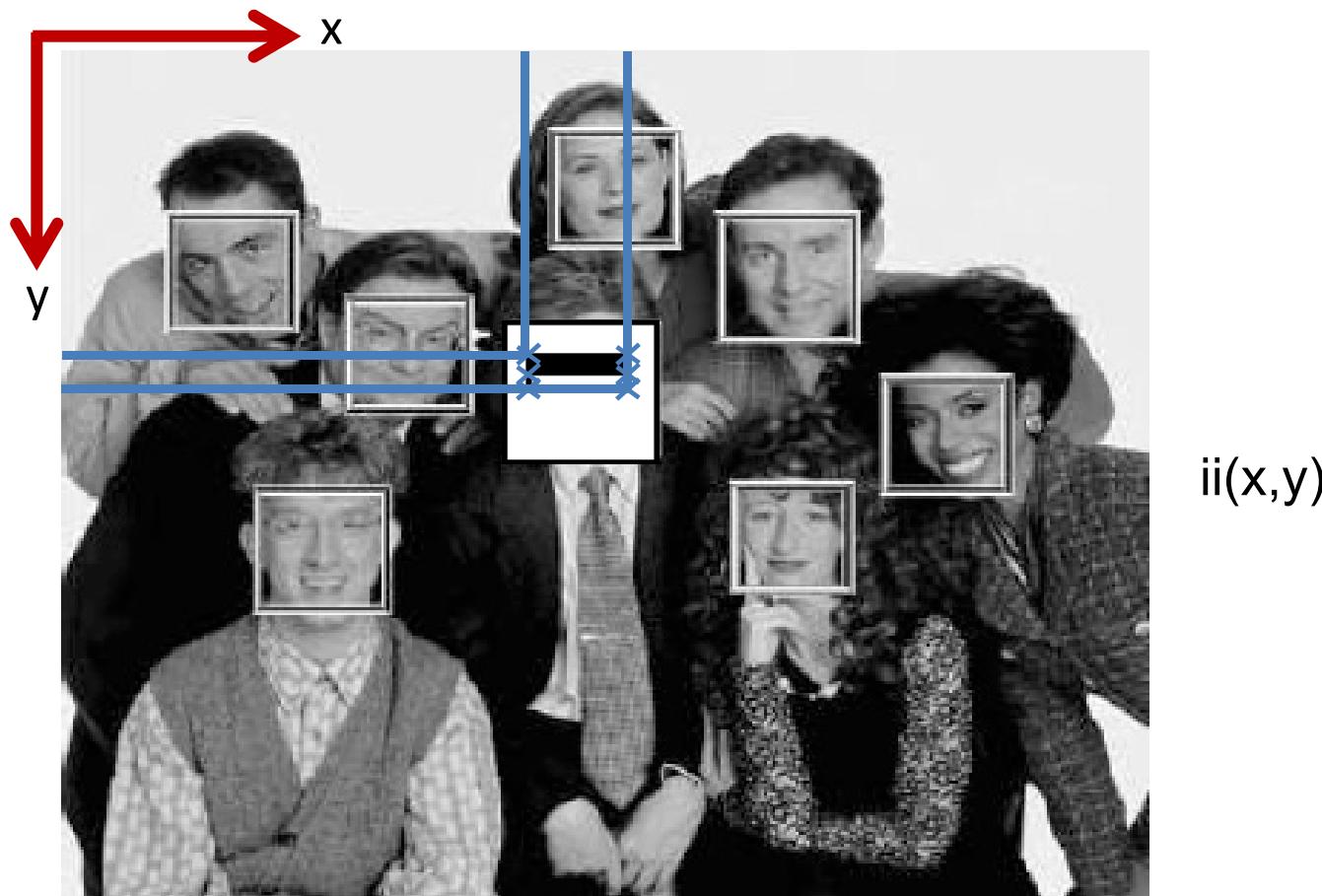
Non-local
maxima
suppression is
carried out to
detect faces.



response map

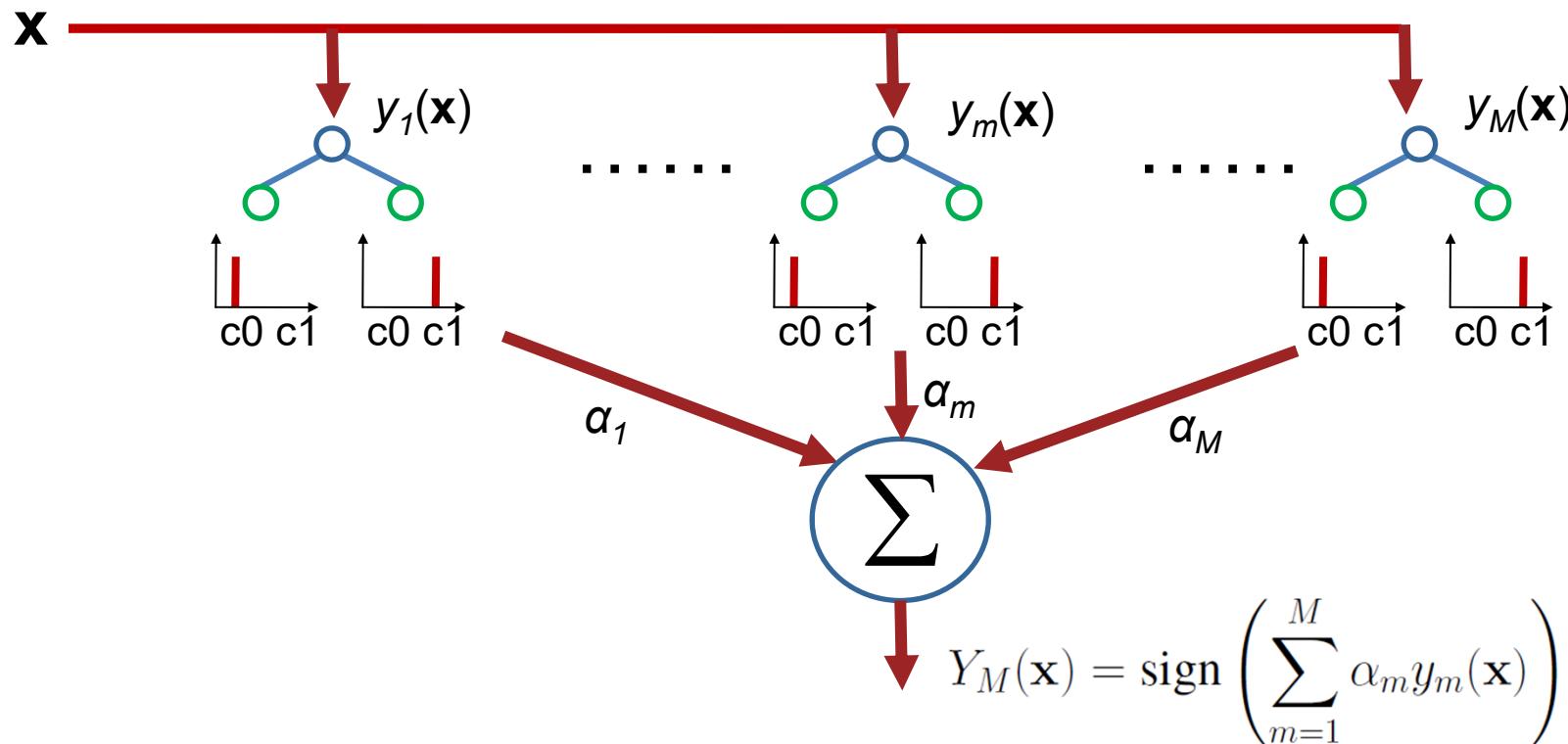
Evaluation (testing)

- The boosting evaluation time is greatly accelerated using the Integral image.
- Each weak classifier is computed using a small number of array references on the integral image.



Boosting (very shallow network)

- The strong classifier $Y_M(\mathbf{x})$ as boosted decision stumps has a flat structure.



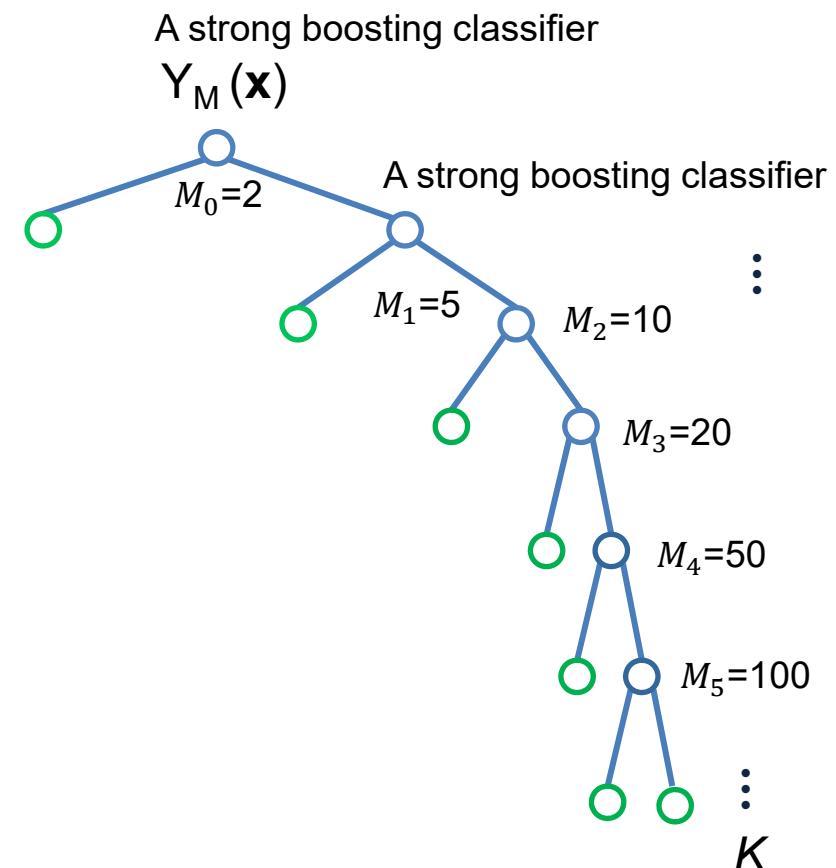
- Cf. Decision “ferns” has been shown to outperform “trees” [Zisserman et al, 07] [Fua et al, 07].

Boosting - continued

- Good generalisation is achieved by a flat structure.
- It does sequential optimisation.
- It provides fast evaluation on top of the Integral image.
- However, its run-time speed is sub-optimal.

A cascade of classifiers

- A cascade of classifiers achieves increased detection performance while radically reducing computation time.
- The key insight is that smaller, and therefore more efficient, boosted classifiers reject many of the negative sub-windows while detecting almost all positive instances.
- Simpler classifiers are used to reject the majority of sub-windows before more complex classifiers are to achieve low false positive rates.
- Stages in the cascade are constructed by AdaBoost.
- It is very imbalanced tree structured.



A cascade of classifiers

- The detection system requires good detection rate and extremely low false positive rates.
- False positive rate and detection rate are

$$F = \prod_{i=0}^K f_i \quad D = \prod_{i=0}^K d_i$$

where f_i is the false positive rate and d_i is the detection rate of i -th classifier on the examples that get through to it.

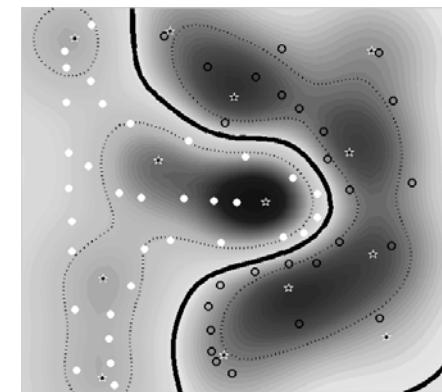
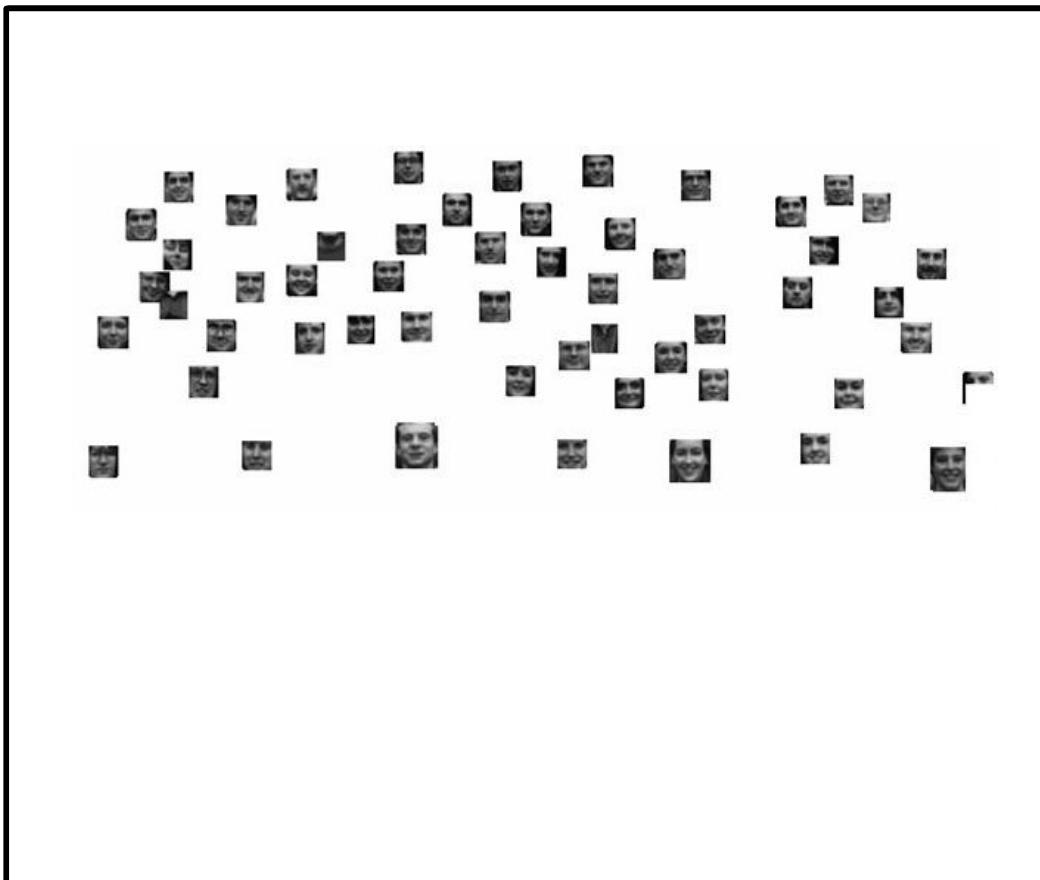
- The expected number of features (or weaklearners) evaluated is

$$M = M_0 + \sum_{i=1}^K \left(M_i \prod_{j < i} p_j \right)$$

where p_j is the proportion of windows input to j -th classifier, and M_i is the number of weaklearners of i -th classifier.

Object detection by a cascade of classifiers

- It speeds up object detection by coarse-to-fine search.



Romdhani et al. ICCV01

Receiver operating characteristic (ROC)

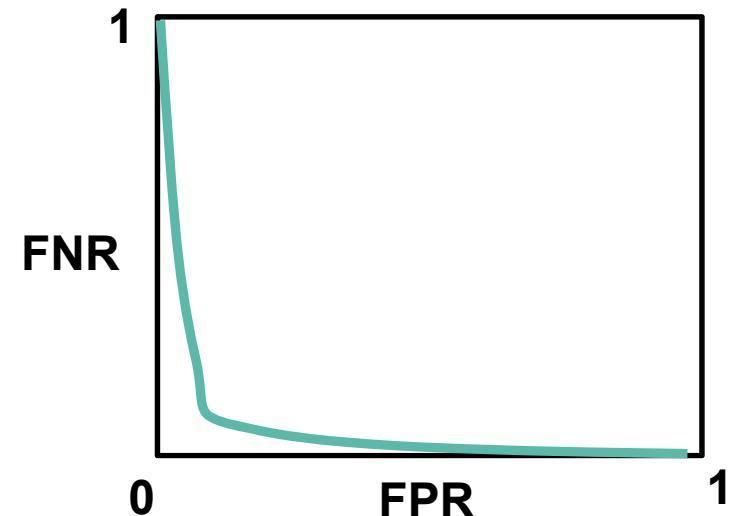
- Boosting classifier score (prior to the binarisation) is compared with a threshold.

$$\sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \begin{cases} \geq \text{Threshold} & \rightarrow \text{Class 1 (face)} \\ < \text{Threshold} & \rightarrow \text{Class -1 (no face)} \end{cases}$$

- The ROC curve is drawn by the false negative rate against the false positive rate at various threshold values:
 - False positive rate (FPR) = FP/N
 - False negative rate (FNR) = FN/P

where

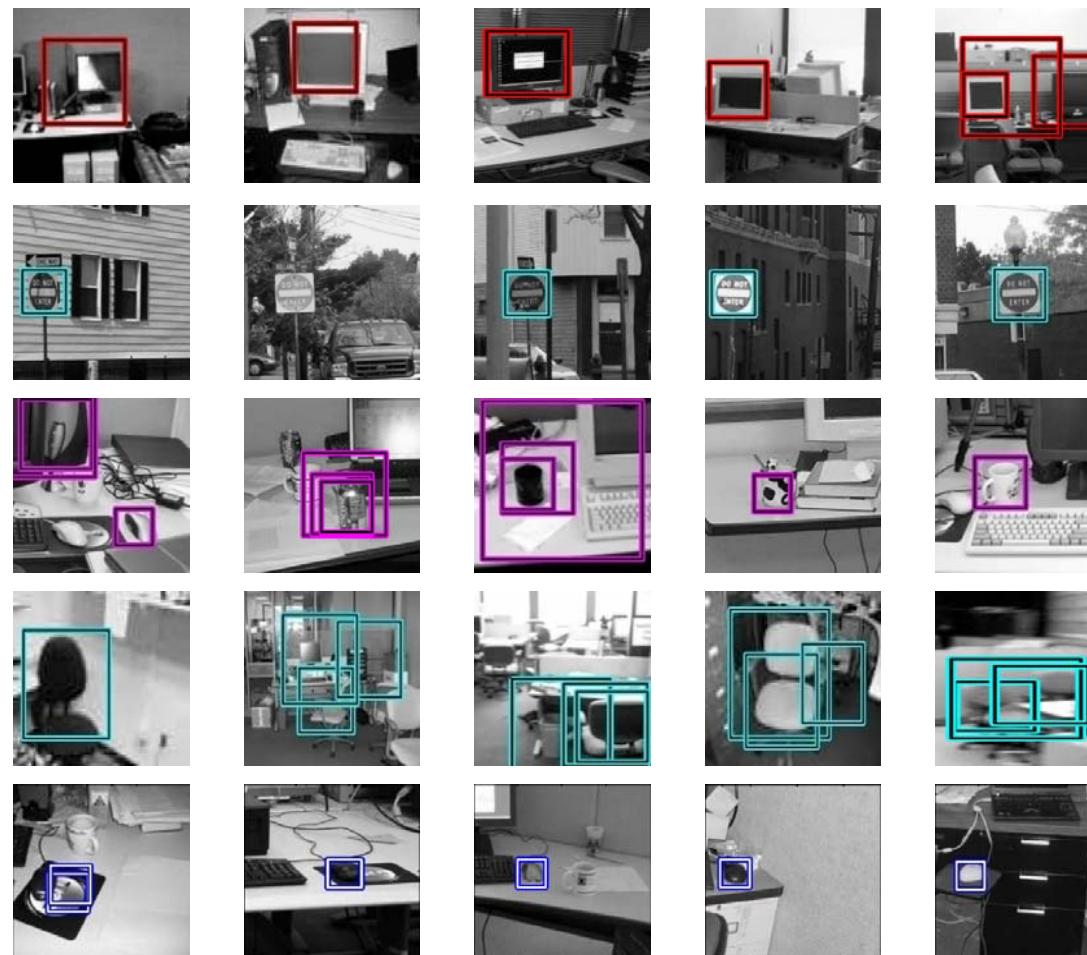
P positive instances,
 N negative instances,
 FP false positive cases, and
 FN false negative cases.



Multiclass object detection

[Torralba et al PAMI 07]

- A boosting algorithm, originally designed for binary class problems, has been extended to multi-class problems.



Multiclass object detection

[Torralba et al PAMI 07]

- Multi-view or multi-category object detection
- Images exhibit multi-modality.
- A single boosting classifier is not sufficient.
- Manual labeling sub-categories.

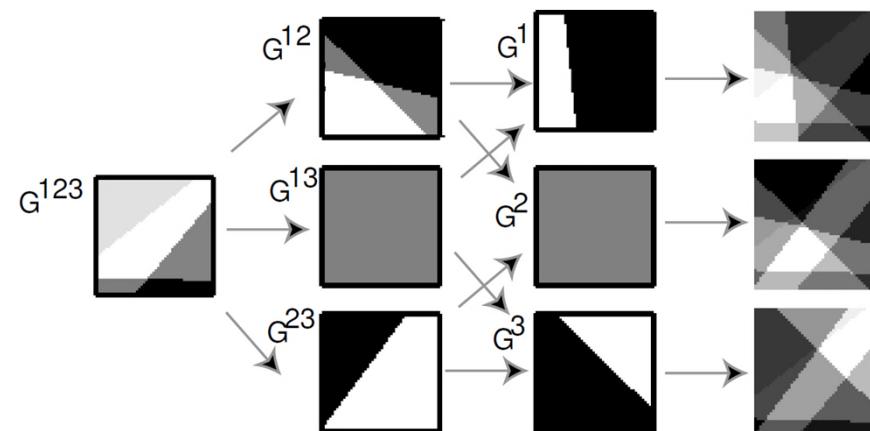
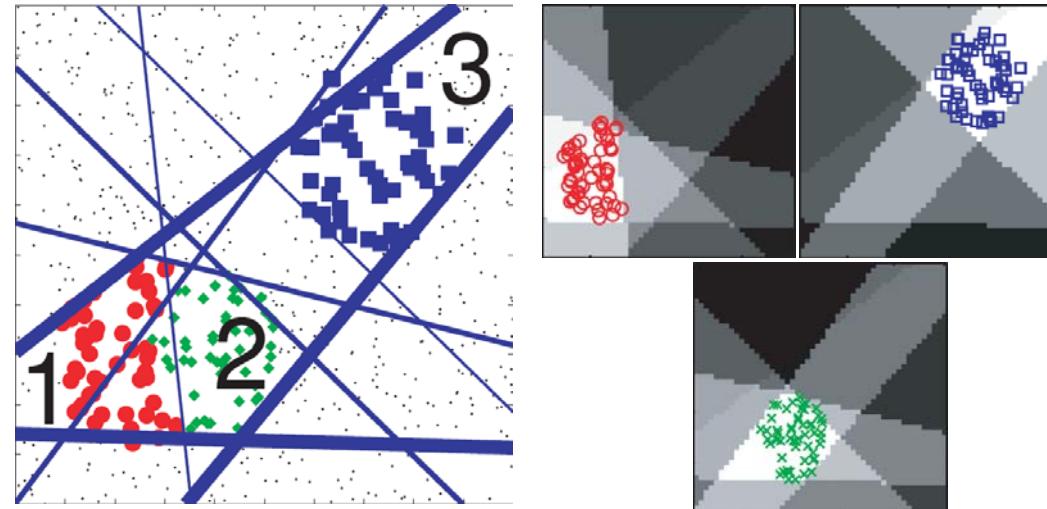


From Torralba et al 07

Multiclass object detection

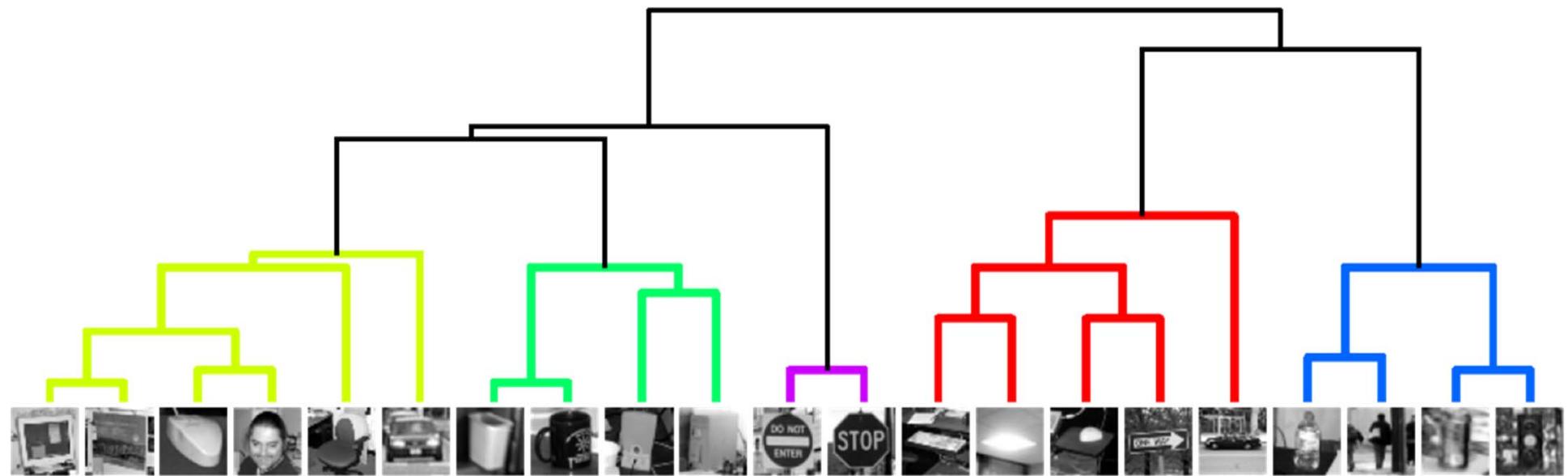
[Torralba et al PAMI 07]

- (automatic) Learning multiple boosting classifiers by sharing features.
- Tree structure speeds up the evaluation (testing) time.



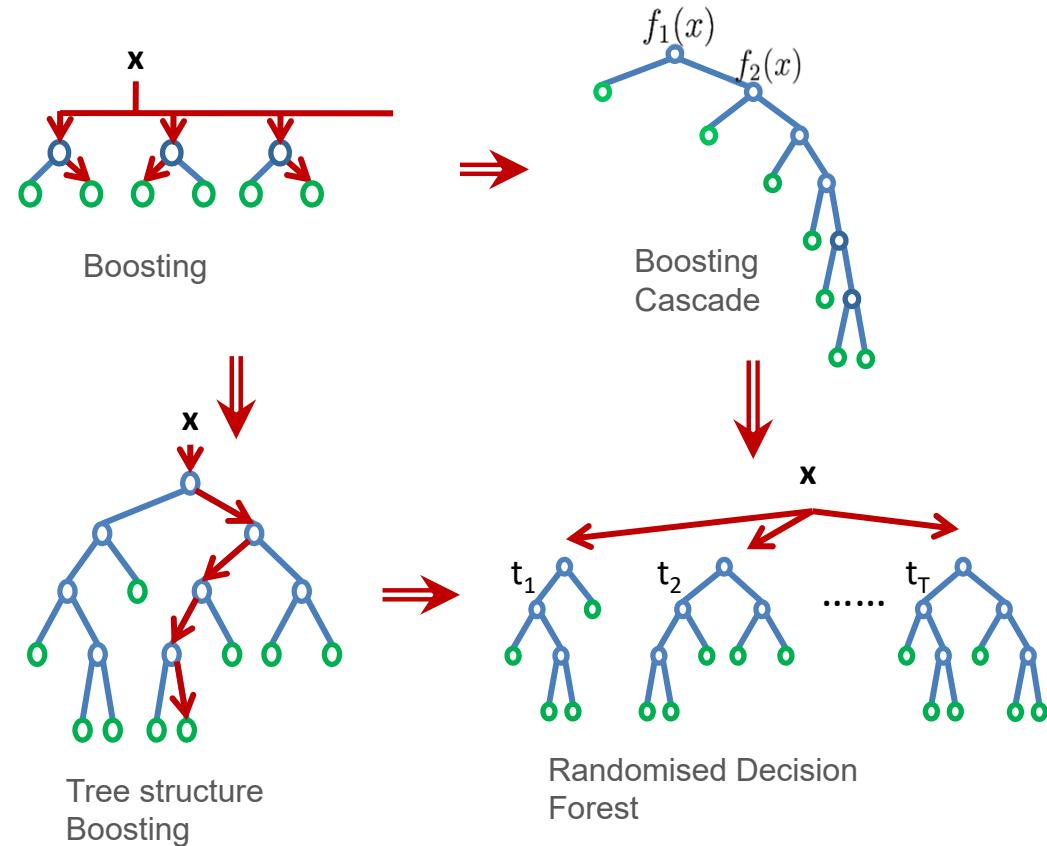
Multiclass object detection

[Torralba et al PAMI 07]

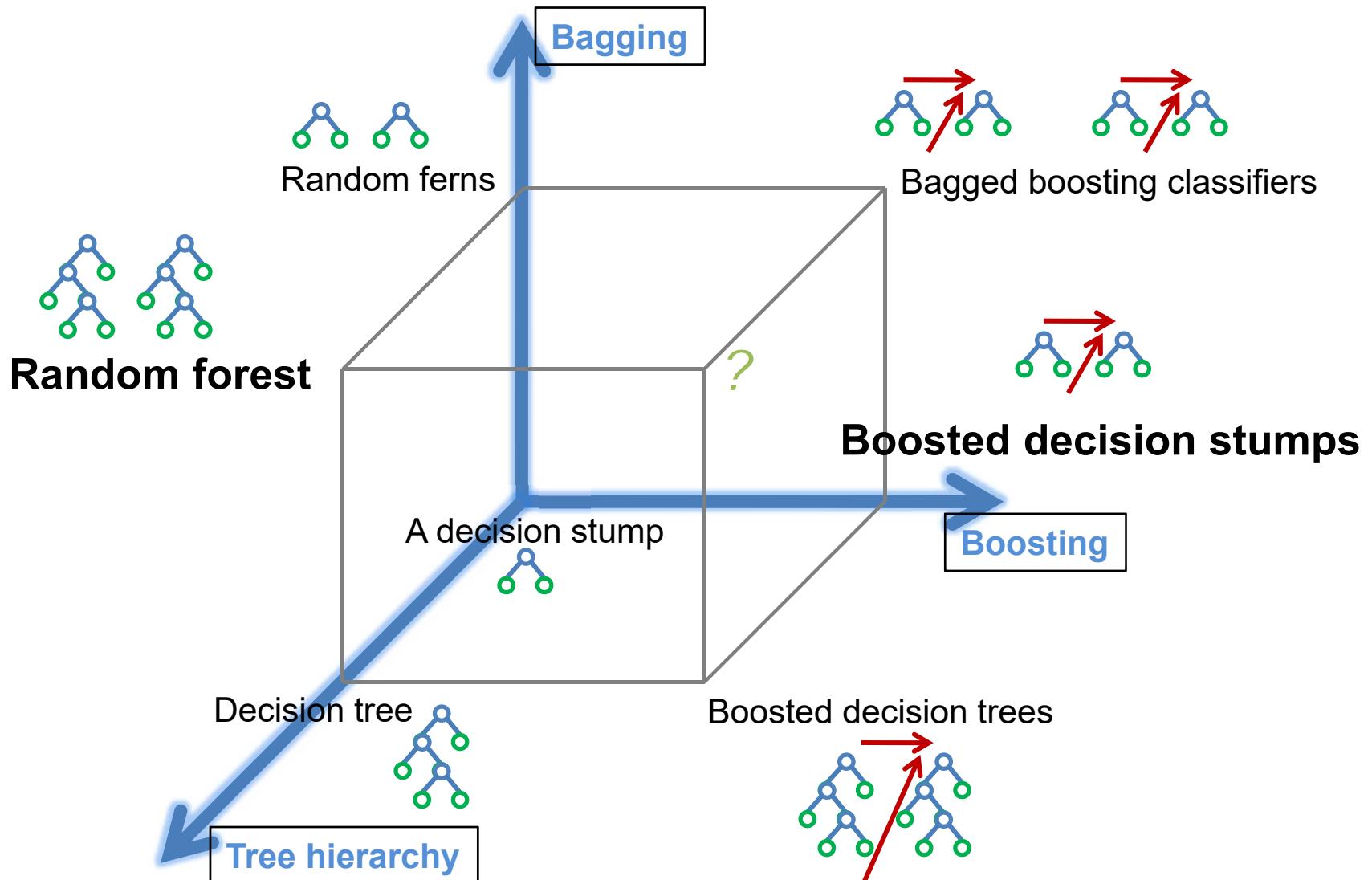


Boosting classifiers and decision forests

- Boosting can be seen in a flat structure.
- Boosting cascade designed to accelerate run-time is a highly imbalanced tree.
- Tree-structured boosting classifiers have been studied to tackle multi-class problems.



Boosting classifiers and decision forests



Summary

Random Forest

- Pros
 - Generalization through random samples/features
 - Extremely fast classification
 - Highly scalable in training
 - Inherently multi-classes
- Cons
 - Inconsistency
 - Difficulty for adaptation

Boosting Decision Stumps

- Pros
 - Generalisation by a flat structure
 - Fast classification
 - Optimisation framework
- Cons
 - Slower than RF
 - Slow training

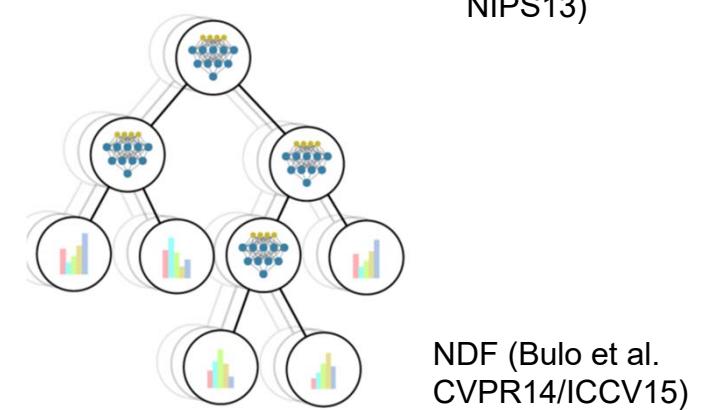
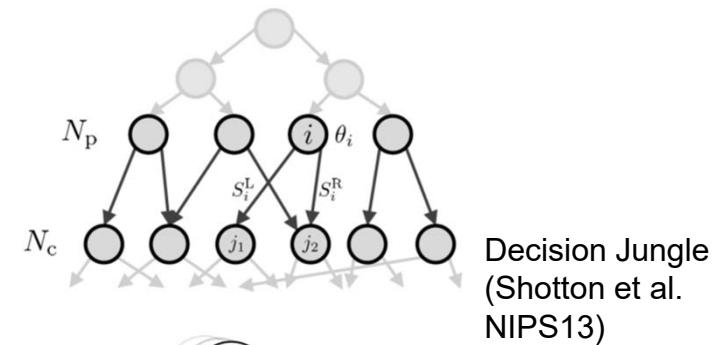
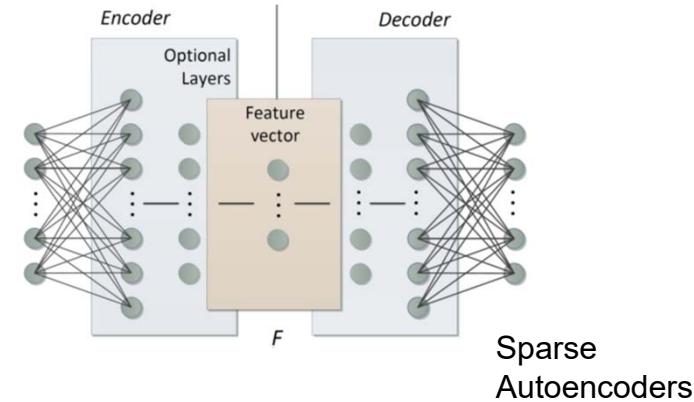
See more in Yin, Criminisi, CVPR07, and Belle et al, ICPR08

Connectivity (memory)

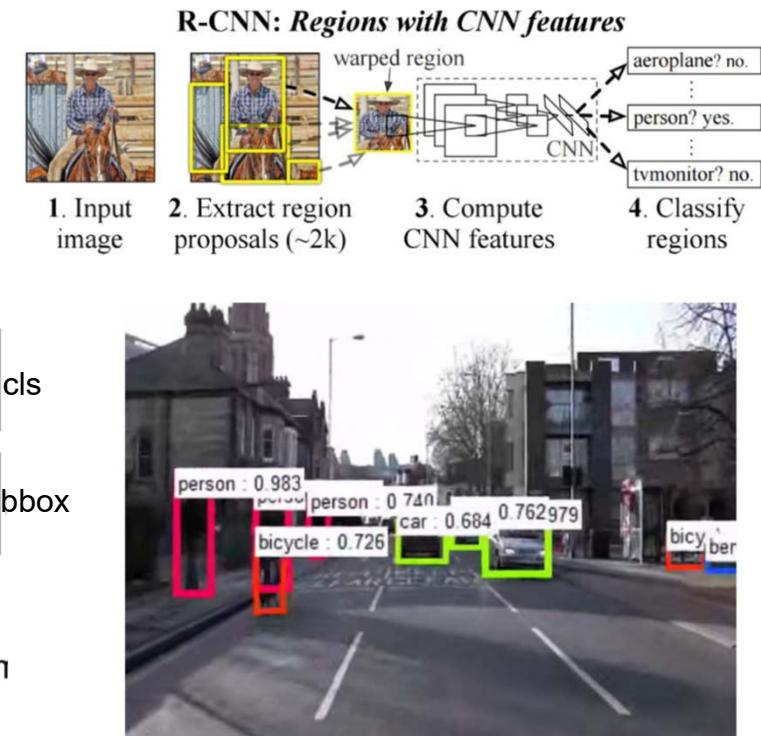
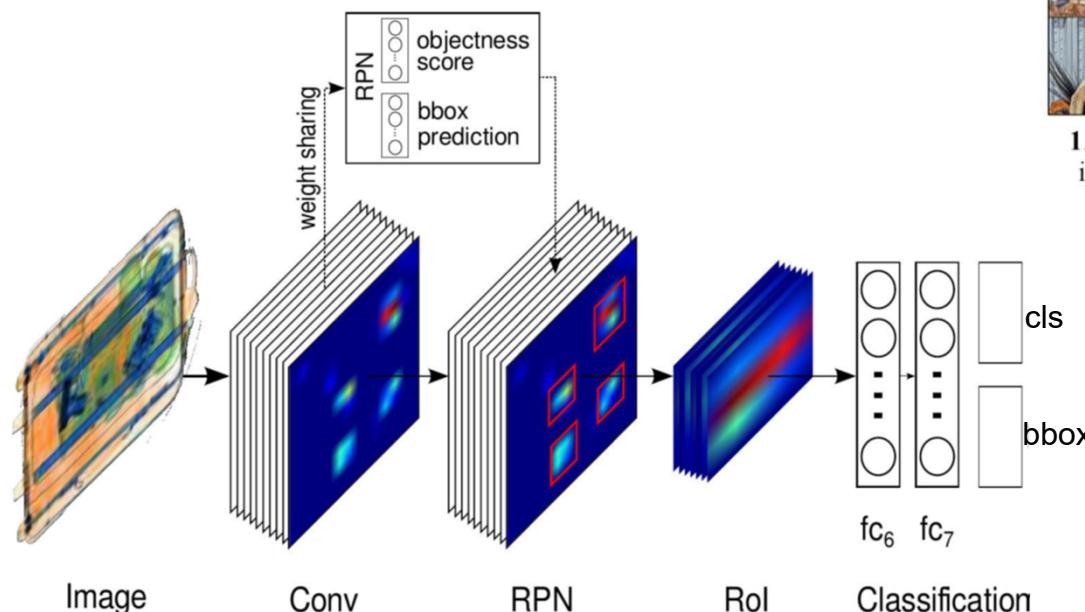
- CNN is fully connected.
- Memory used in decision trees grows exponentially with depth.
- Decision Jungle (ensembles of directed acyclic graphs) allows multiple paths.

Data representation

- DF requires a set of features manually defined.
- Neural Decision Forests jointly tackles data representation and discriminative learning, using randomized Multi-Layer Perceptrons as split nodes.
- Hough Networks (cf. Hough Forest)
(Riegler et al. BMVC14) jointly perform classification and regression.



Region-based CNN (R-CNN)



- Learn both a **Region Proposal Network (RPN)**
 - *likely object locations, given the image*
- ... and then **classify those regions** via existing CNN architecture (jointly trained)

<https://www.youtube.com/watch?v=WZmSMkK9VuA>
<https://www.youtube.com/watch?v=uKU2pzpGUIM>