《机器学习》课件

4.2 Adaboost and Object Detection

北京航空航天大學
BEIHANG UNIVERSITY

集成算法

- 集成方法
  - 并行方法
    - bagging
    - 随机森林
  - 串行方法
    - GBDT
    - Adaboost
    - XGBoost

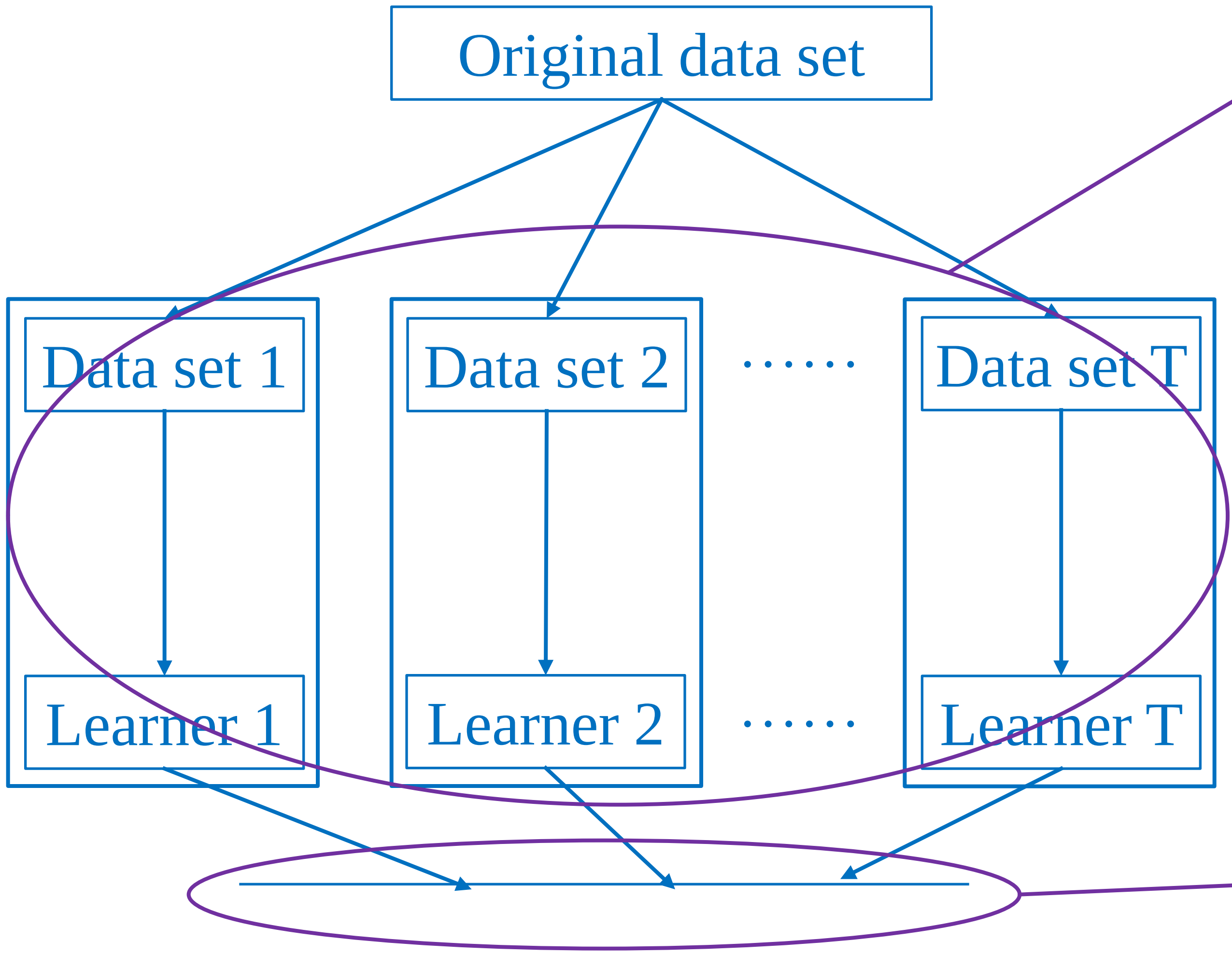
Bagging集成算法

- 集成学习是机器学习算法中非常耀眼的一类方法，它通过训练多个基本的分类器（如支持向量机、神经网络、决策树等），再通过基本分类器的决策融合，构成一个完整的具有更强学习分辨能力的学习器。
- Bagging（Bootstrap aggregation，引导聚集算法）：是集成学习的框架，用于减小预测误差，可并行拟合。

Bagging集成算法

- Bagging集成方法的框架：
  - 每轮从原始样本集中使用Bootstrap（有放回）的方法抽取n个训练样本。共进行T轮抽取，得到T个训练集。
  - 每次使用一个训练集得到一个模型，T个训练集共得到T个模型。
  - 分类任务使用投票的方式集成，而回归任务通过平均的方式集成。

Bagging集成算法



Bootstrap a set of learners
Generate many data sets from the original data set through bootstrap sampling(random sampling with replacement), then train an individual learner per data set

Voting for classification
The output is the class label receiving the most number of votes
Averaging for regression
The output is the average output of the individual learners

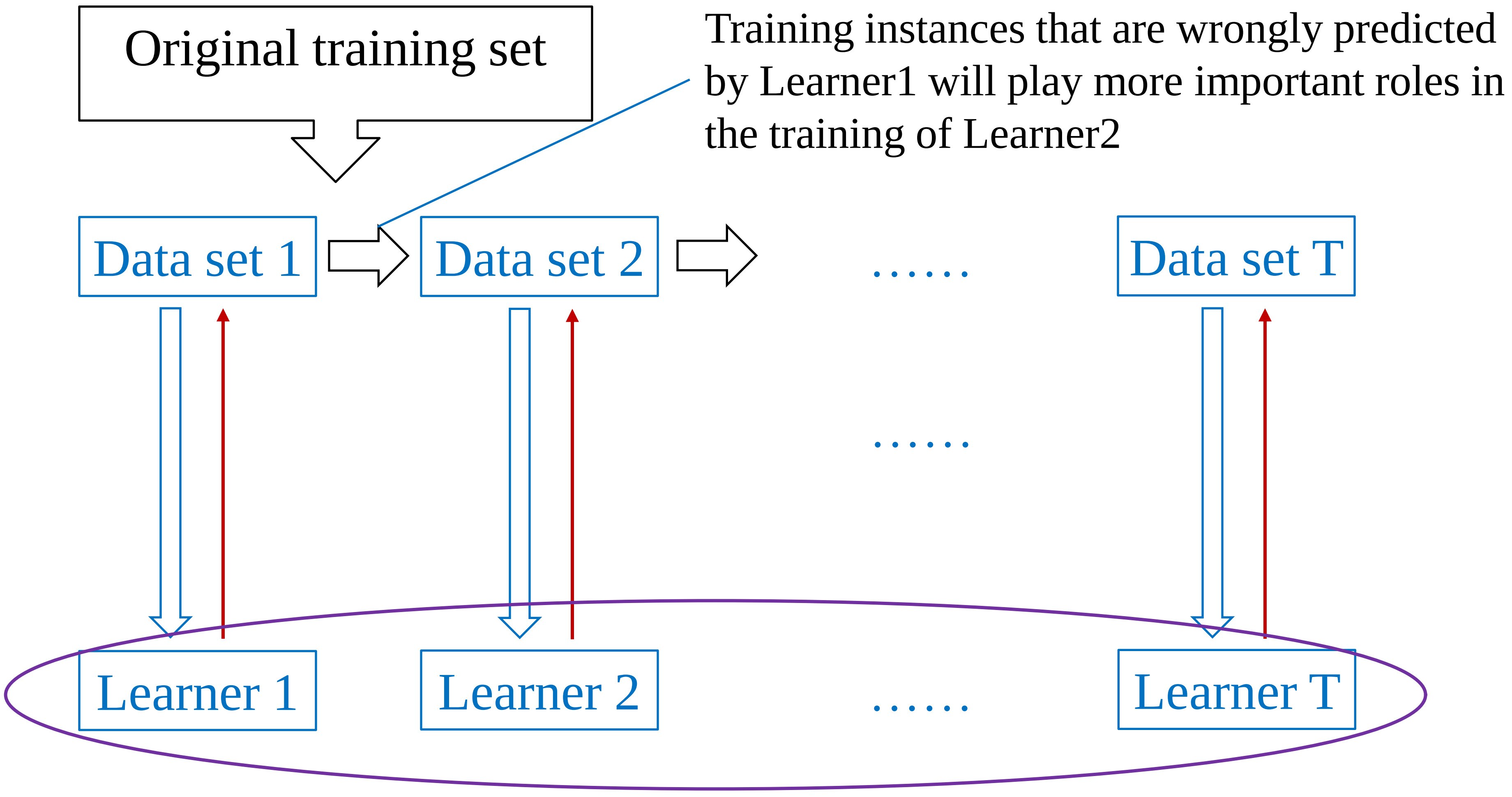
Boosting集成算法

- Boosting集成方法产生于计算学习理论（Computational Learning Theory），是一种将弱分离器组合起来形成强分类器的算法框架。

Boosting集成算法

- Boosting集成方法具有一个类似的框架
  - 根据当前的数据训练出一个弱模型
  - 根据该弱模型的表现调整数据样本的权重
    - 让做错的样本在后续的训练中获得更多的关注
    - 让做对的样本在后续的训练中获得较少的关注
  - 最后根据弱模型的表现决定该弱模型的“话语权”。

Boosting集成算法


Original training set
Training instances that are wrongly predicted by Learner1 will play more important roles in the training of Learner2
Data set 1
Data set 2
……
Data set T
……
Learner 1
Learner 2
……
Learner T

Weighted combination

Adaboost算法

- Boosting集成方法的关键在于：
  - 如何根据弱模型的表现更新训练集的权重
  - 如何根据弱模型的表现决定弱模型的“话语权”。
- Adaboost（Adaptive Boosting）算法已被证明是一种有效而实用的Boosting算法。该算法是Freund和Schapire于1995年对Boosting算法的改进得到的。

Principle of Adaboost

- Three cobblers with their wits combined equal Zhuge Liang the master mind.
- Failure is the mother of success

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

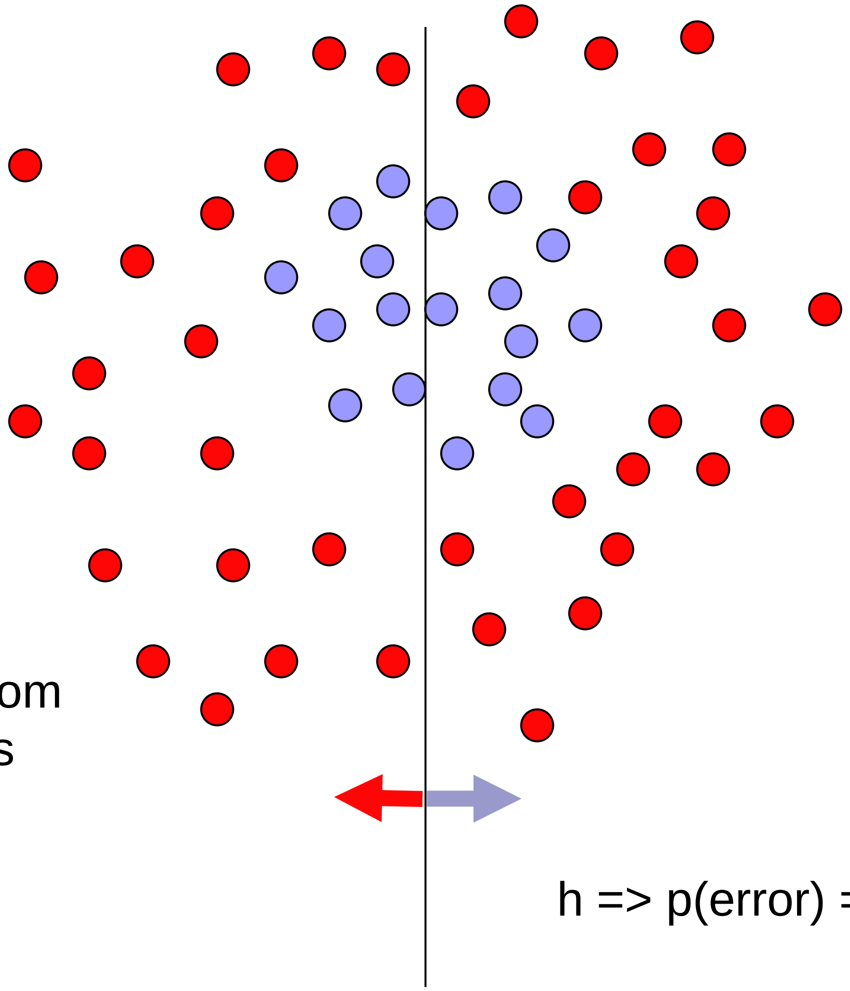
Strong classifier

Weak classifier

Weight

Features vector

Toy Example – **taken from Antonio Torralba @MIT**



Each data point has

a class label:

$$y_t = \begin{cases} +1 \ (●) \\ -1 \ (●) \end{cases}$$

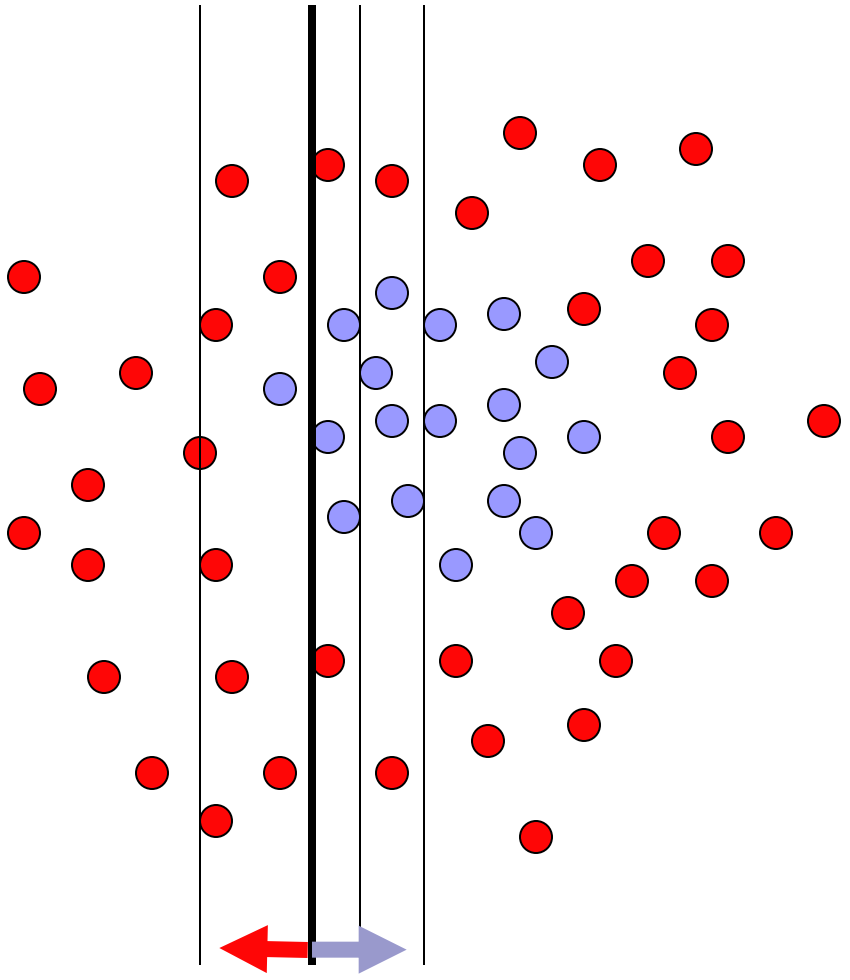
and a weight:

$w_t = 1$

Weak learners from the family of lines

h => p(error) = 0.5 it is at chance

Toy example



This one seems to be the best

Each data point has

a class label:

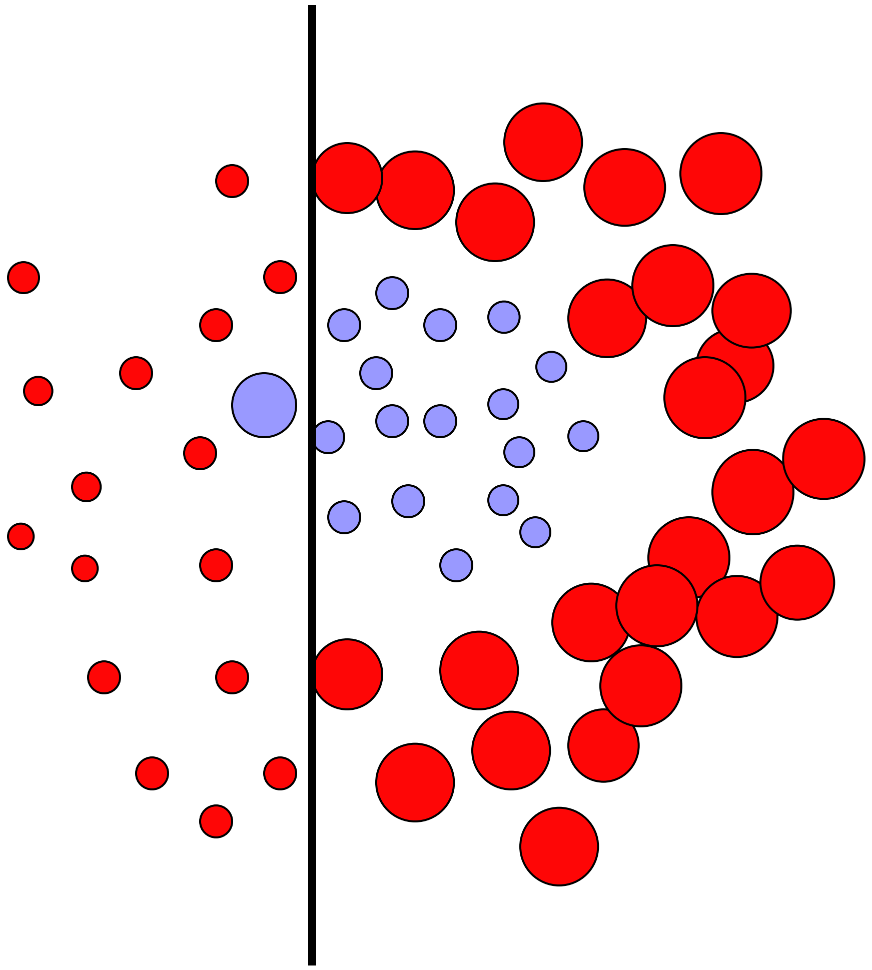
$$y_t = \begin{cases} +1 \ (●) \\ -1 \ (●) \end{cases}$$

and a weight:

$w_t = 1$

This is a ‘**weak classifier**’: It performs slightly better than chance.

Toy example



Each data point has

a class label:

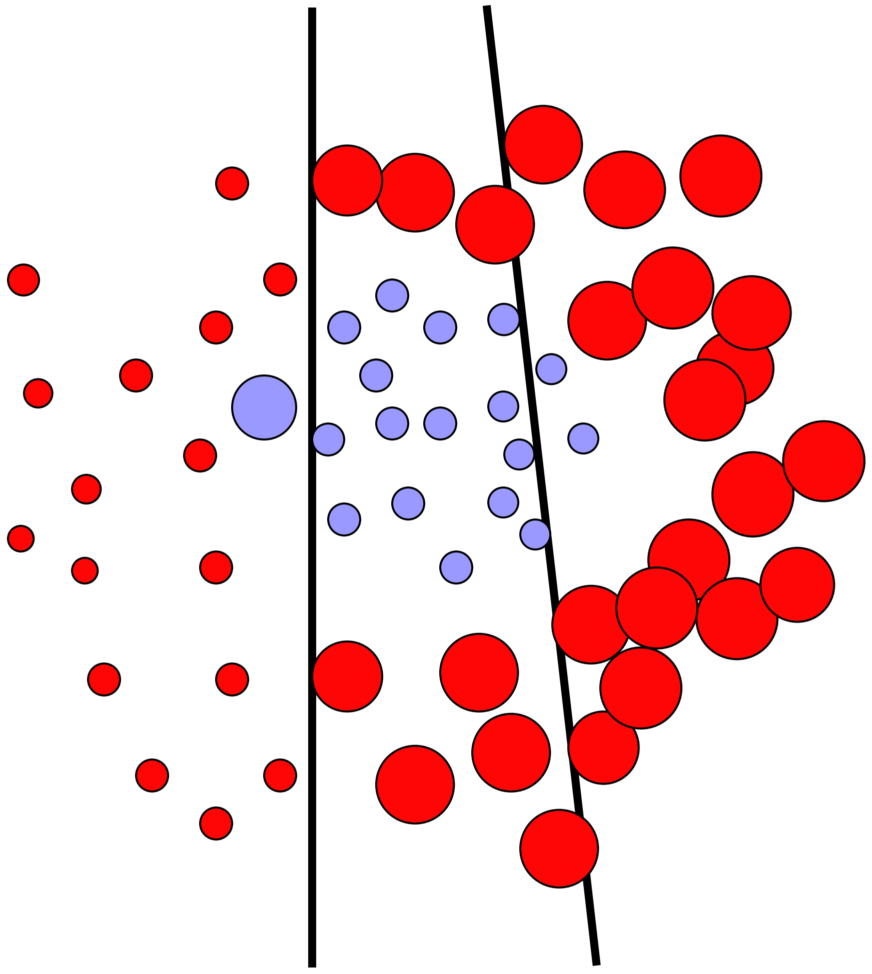
$$y_t = \begin{cases} +1 \; (●) \\ -1 \; (●) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has

a class label:

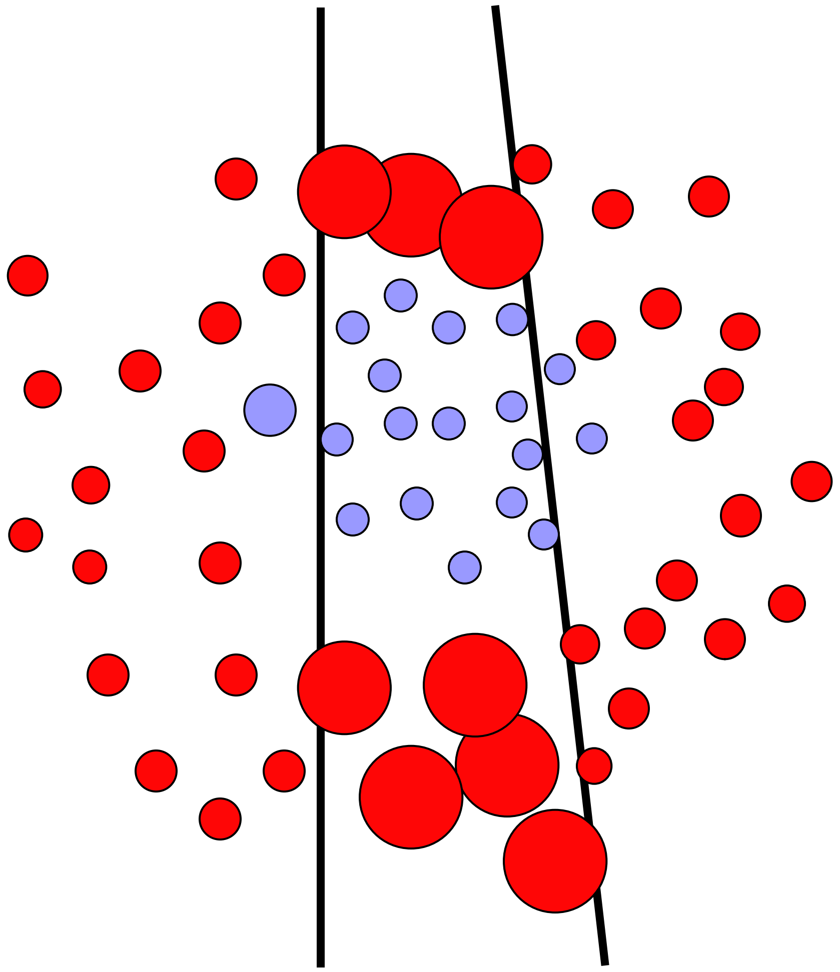
$$y_t = \begin{cases} +1 \ (●) \\ -1 \ (●) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has

a class label:

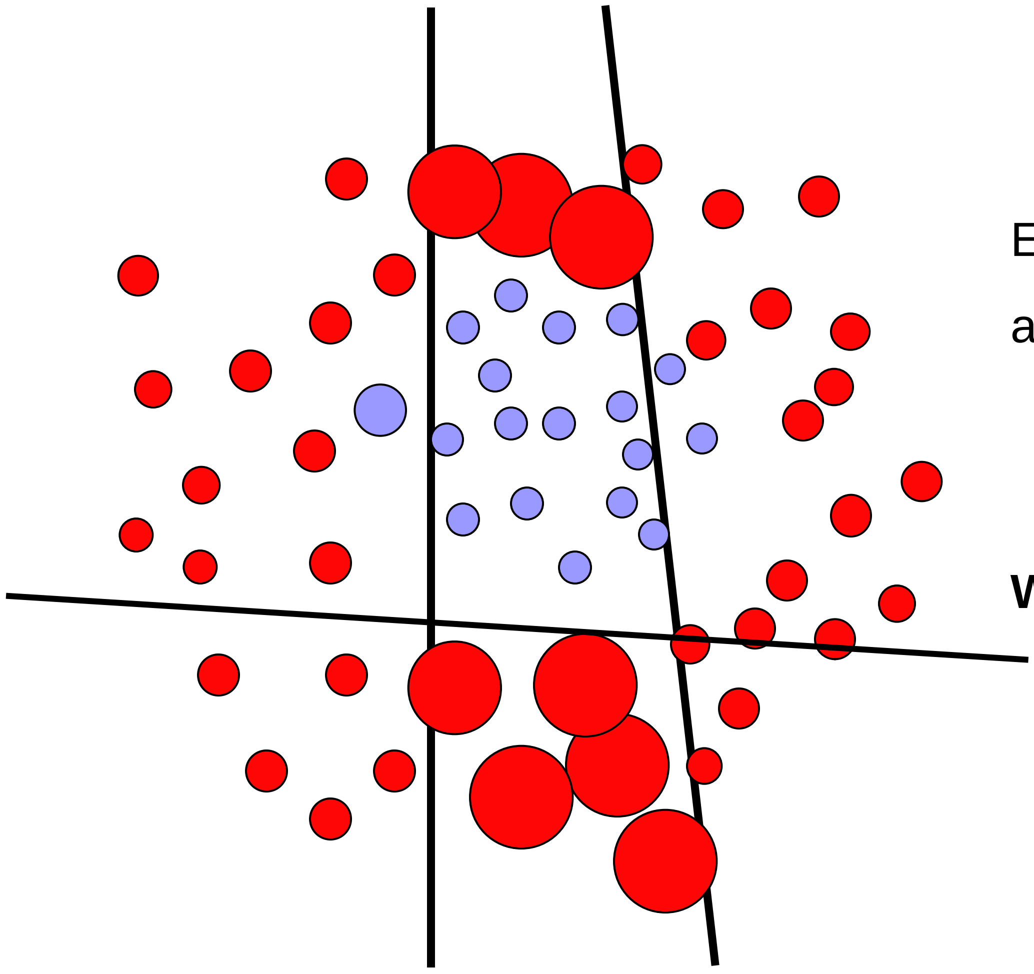
$$y_t = \begin{cases} +1 \ (🔴) \\ -1 \ (🔵) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has

a class label:

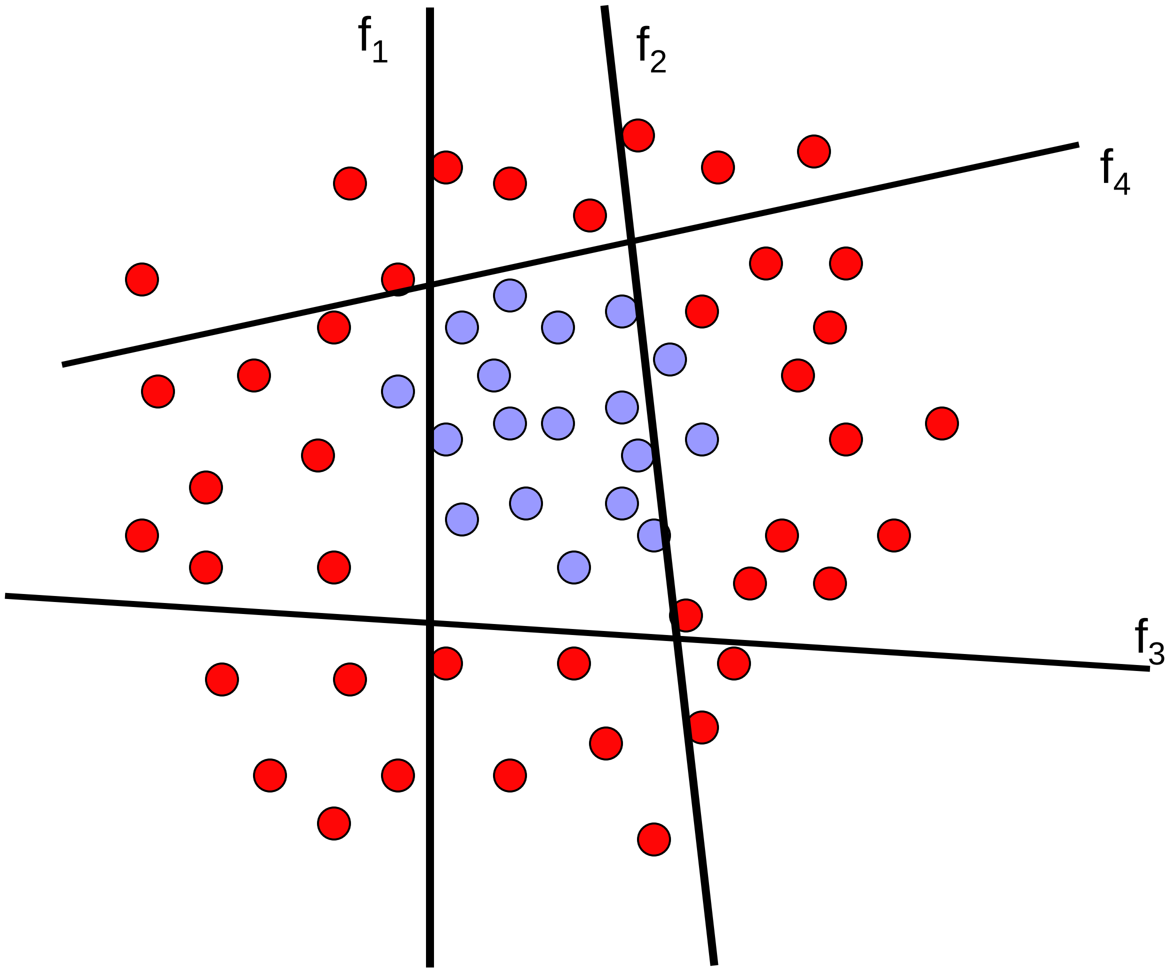
$$y_t = \begin{cases} +1 & (●) \\ -1 & (●) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Formal Procedure of AdaBoost

- given training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$:
    - construct distribution D_t on $\{1, \ldots, m\}$
    - find weak classifier ("rule of thumb")

      $$h_t : X \to \{-1, +1\}$$

      with small error ϵ_t on D_t:

      $$\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$$
- output final classifier H_{final}

Procedure of Adaboost

- constructing D_t:

$$w_{t+1,i} = w_{t,i}\, \alpha_t^{\,1-e_i}$$

  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t\, y_i\, h_t(x_i))$$

where Z_t = normalization constant

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

- final classifier:
  - $H_{\text{final}}(x) = \operatorname{sign}\left(\sum_t \alpha_t h_t(x)\right)$

Adaboost – 算法

- 对于给定的训练样本集合 $X = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- $y_i \in \{-1, +1\}$ 表示这些样本的真实标号

- For (t = 1, …, T)
  - 在 $\{1, \ldots, n\}$ 构造 D_t
  - 找到弱分类器 $h_t : X \to \{-1, +1\}$

    使得分类误差：$\varepsilon_t = \mathrm{P}_{D_t}\left[h_t(x_i) \neq y_i\right]$最小

- 输出最终的假设（分类器）

$$H(x) = sign\left(\sum_t \alpha_t \cdot h_t(x)\right)$$

Adaboost – 算法

- 改变训练过程中样本集合的重要性（权值）D_t
- $D_1(i) = 1.0 / n$
- 给定 D_t 与 h_t 令 $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$

$$= \frac{D_t(i)}{Z_t} \cdot \exp(-\alpha_t\, y_i\, h_t(x_i))$$

  - 其中 Z_t 是归一化常数

$$\alpha_t = \tfrac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

- 输出最终的假设（分类器）

$$H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$$

Error on Training Set

- Theorem:
  - write ϵ_t as $1/2 - \gamma_t$
  - then

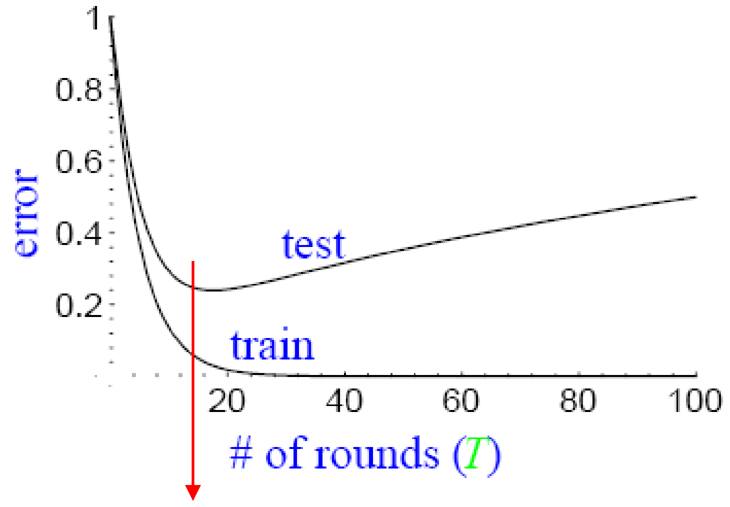
$$\text{training error}(H_{\text{final}}) \leq \exp\left(-2\sum_t \gamma_t^2\right)$$

- so: if $\forall t: \ \gamma_t \geq \gamma > 0$

  then $\text{training error}(H_{\text{final}}) \leq e^{-2\gamma^2 T}$
- AdaBoost is adaptive:
  - does not need to know γ or T a priori
  - can exploit $\gamma_t \gg \gamma$

But we are NOT interested in Training set

- Will Adaboost screw up with a fat complex classifier finally?

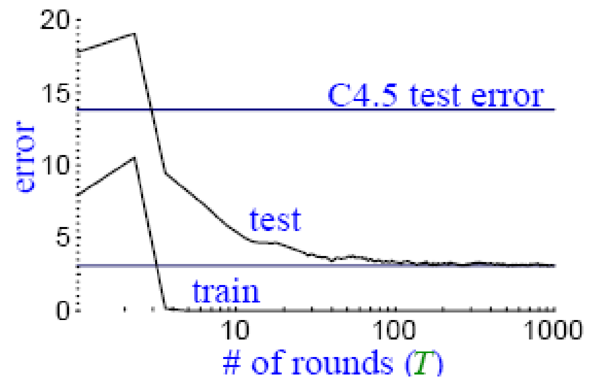


Occam's razor – simple is the best

Over fitting

Shall we stop before over fitting? If only over fitting happens.

Actual Typical Run



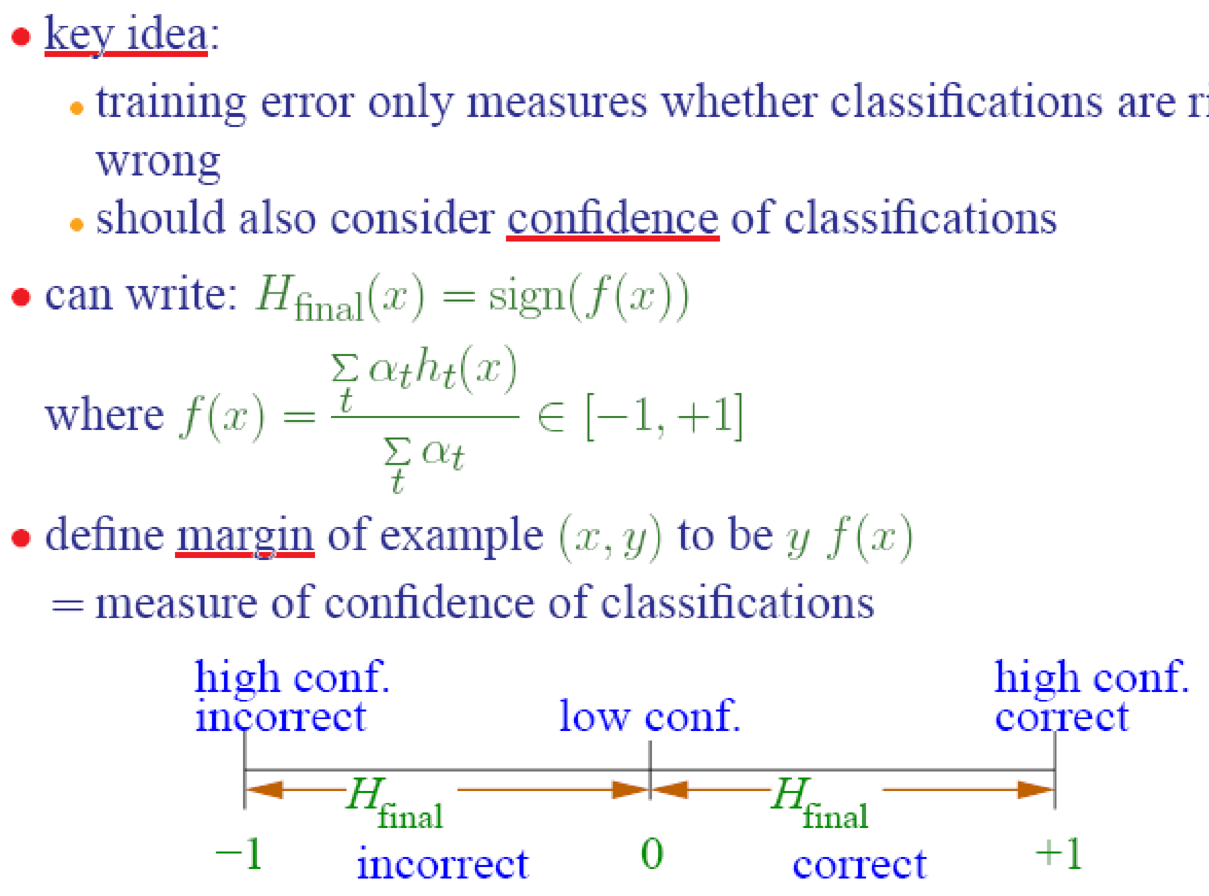
(boosting C4.5 on "letter" dataset)

- test error does not increase, even after 1000 rounds
  - (total size > 2,000,000 nodes)
- test error continues to drop even after training error is zero!

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

An explanation by margin

- This margin is not the margin in SVM



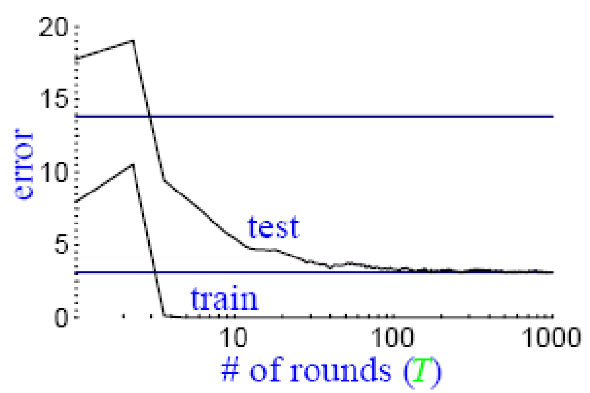
- key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications
- can write: $H_{\text{final}}(x) = \text{sign}(f(x))$

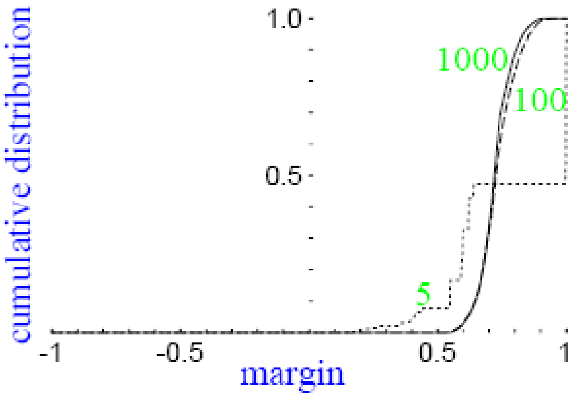
  where $f(x) = \frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \in [-1, +1]$
- define margin of example (x, y) to be $y\, f(x)$

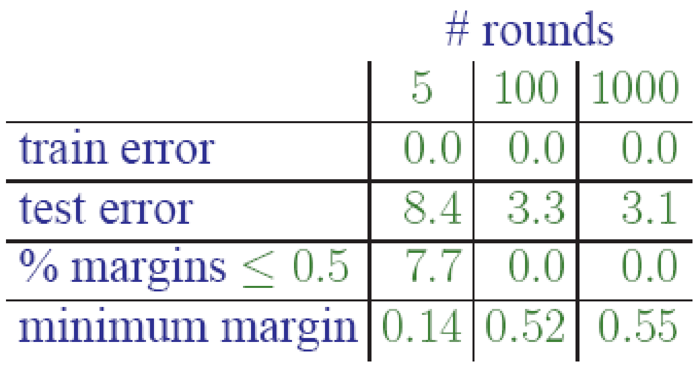
  = measure of confidence of classifications

high conf. incorrect
low conf.
high conf. correct
H_{final} incorrect
H_{final} correct
-1
0
$+1$

Margin Distribution







	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Although final classifier is getting larger, margins are still increasing

Final classifier is actually getting to simpler classifer

Two Questions

- Will adaboost always maximize the margin?

  AdaBoost may converge to a margin that is significantly below maximum. (R, Daubechies, Schapire 04)

- If finally we reach a simpler classifier, is there anyway to compress it? Or can we bypass boosting but reach a simple classifier?

XOR problem

$$(0,+1),y_1=+1)$$
$$(0,-1),y_2=+1)$$
$$(+1,0),y_3=-1)$$
$$(-1,0),y_4=-1)$$

$$h_1(x)=\begin{cases}+1, \text{ if } (x_1>-0.5) \\ -1, \quad \text{otherwise}\end{cases} \quad h_2(x)=\begin{cases}-1, \text{ if } (x_1>-0.5) \\ +1, \quad \text{otherwise}\end{cases}$$

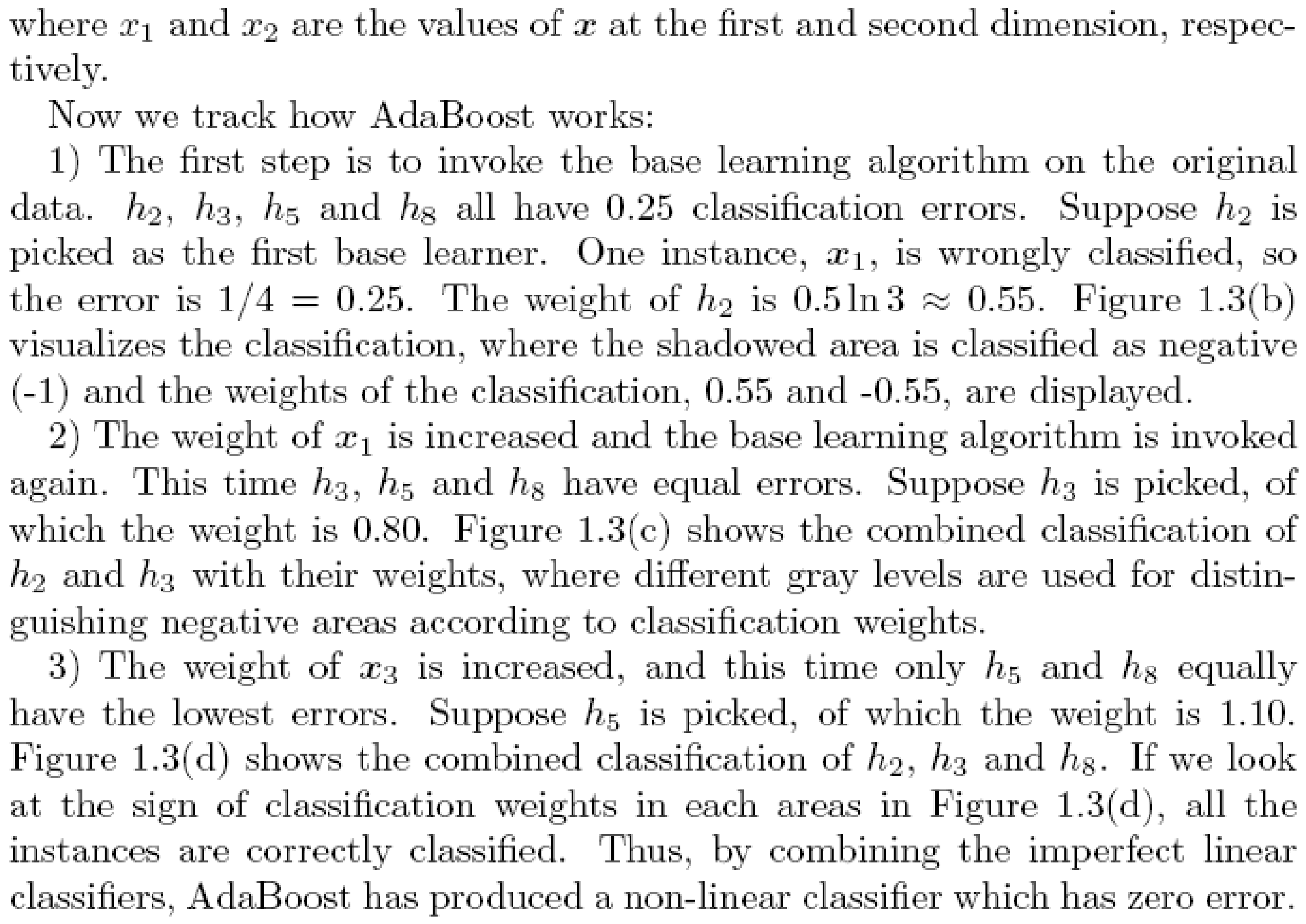
XOR problem

$$h_3(x) = \begin{cases} +1, \text{ if } (x_1 > +0.5) \\ -1, \quad \text{otherwise} \end{cases} \quad h_4(x) = \begin{cases} -1, \text{ if } (x_1 > +0.5) \\ +1, \quad \text{otherwise} \end{cases}$$

$$h_5(x) = \begin{cases} +1, \text{ if } (x_2 > -0.5) \\ -1, \quad \text{otherwise} \end{cases} \quad h_6(x) = \begin{cases} -1, \text{ if } (x_2 > -0.5) \\ +1, \quad \text{otherwise} \end{cases}$$

$$h_7(x) = \begin{cases} +1, \text{ if } (x_2 > +0.5) \\ -1, \quad \text{otherwise} \end{cases} \quad h_8(x) = \begin{cases} -1, \text{ if } (x_2 > +0.5) \\ +1, \quad \text{otherwise} \end{cases}$$

XOR problem



where x_1 and x_2 are the values of x at the first and second dimension, respectively.

Now we track how AdaBoost works:

1) The first step is to invoke the base learning algorithm on the original data. h_2, h_3, h_5 and h_8 all have 0.25 classification errors. Suppose h_2 is picked as the first base learner. One instance, x_1, is wrongly classified, so the error is $1/4 = 0.25$. The weight of h_2 is $0.5 \ln 3 \approx 0.55$. Figure 1.3(b) visualizes the classification, where the shadowed area is classified as negative (-1) and the weights of the classification, 0.55 and -0.55, are displayed.

2) The weight of x_1 is increased and the base learning algorithm is invoked again. This time h_3, h_5 and h_8 have equal errors. Suppose h_3 is picked, of which the weight is 0.80. Figure 1.3(c) shows the combined classification of h_2 and h_3 with their weights, where different gray levels are used for distinguishing negative areas according to classification weights.

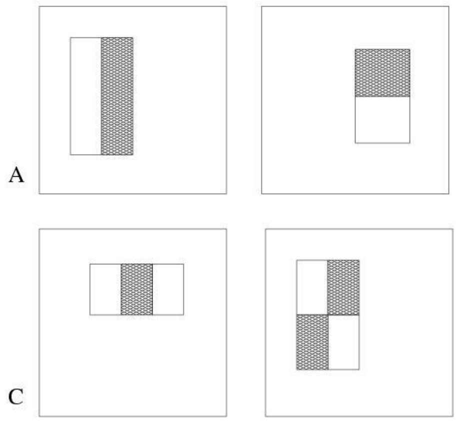
3) The weight of x_3 is increased, and this time only h_5 and h_8 equally have the lowest errors. Suppose h_5 is picked, of which the weight is 1.10. Figure 1.3(d) shows the combined classification of h_2, h_3 and h_8. If we look at the sign of classification weights in each areas in Figure 1.3(d), all the instances are correctly classified. Thus, by combining the imperfect linear classifiers, AdaBoost has produced a non-linear classifier which has zero error.

Robust Real-time Object Detection
Viola & Jones

Key Ideas

- Integral Image
- Critical feature selection and better detection using AdaBoost
- Classifier cascade to minimize computation

The features used



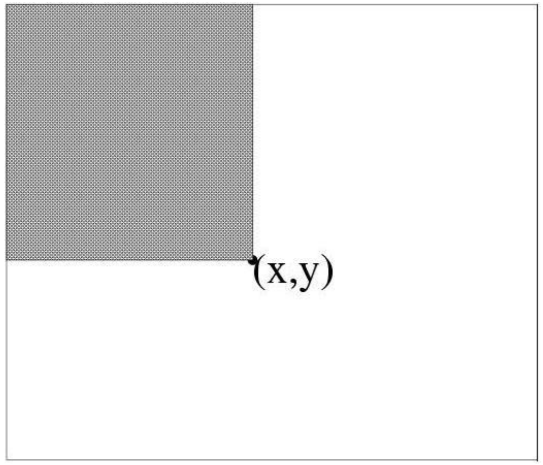
Rectangular feature types:

- *two-rectangle feature* (horizontal/vertical)
- *three-rectangle feature*
- *four-rectangle feature*

Using a 24x24 pixel base detection window, with all possible combinations of orientation, location and scale of these feature types the full set of features has 49,396 features.

The motivation behind using rectangular features, as opposed to more expressive steerable filters is their extreme computational efficiency.

Integral image



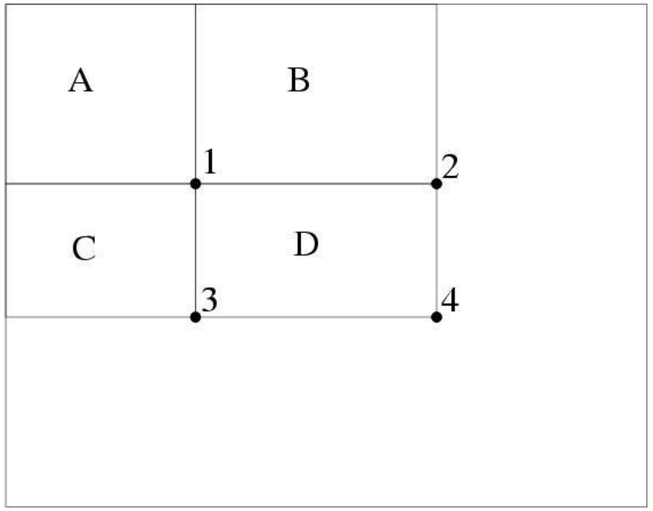
Def: The *integral image* at location (*x*,*y*), is the sum of the pixel values above and to the left of (*x*,*y*), inclusive.

Using the following two recurrences, where *i*(*x*,*y*) is the pixel value of original image at the given location and *s*(*x*,*y*) is the cumulative row sum, we can calculate the integral image representation of the image in a single pass.

$s(x,y) = s(x,y-1) + i(x,y)$ .... integration along rows

$ii(x,y) = ii(x-1,y) + s(x,y)$ .... integration along columns

Rapid evaluation of rectangular features



Using the integral image representation one can compute the value of any rectangular sum in constant time.

For example the integral sum inside rectangle D we can compute as:

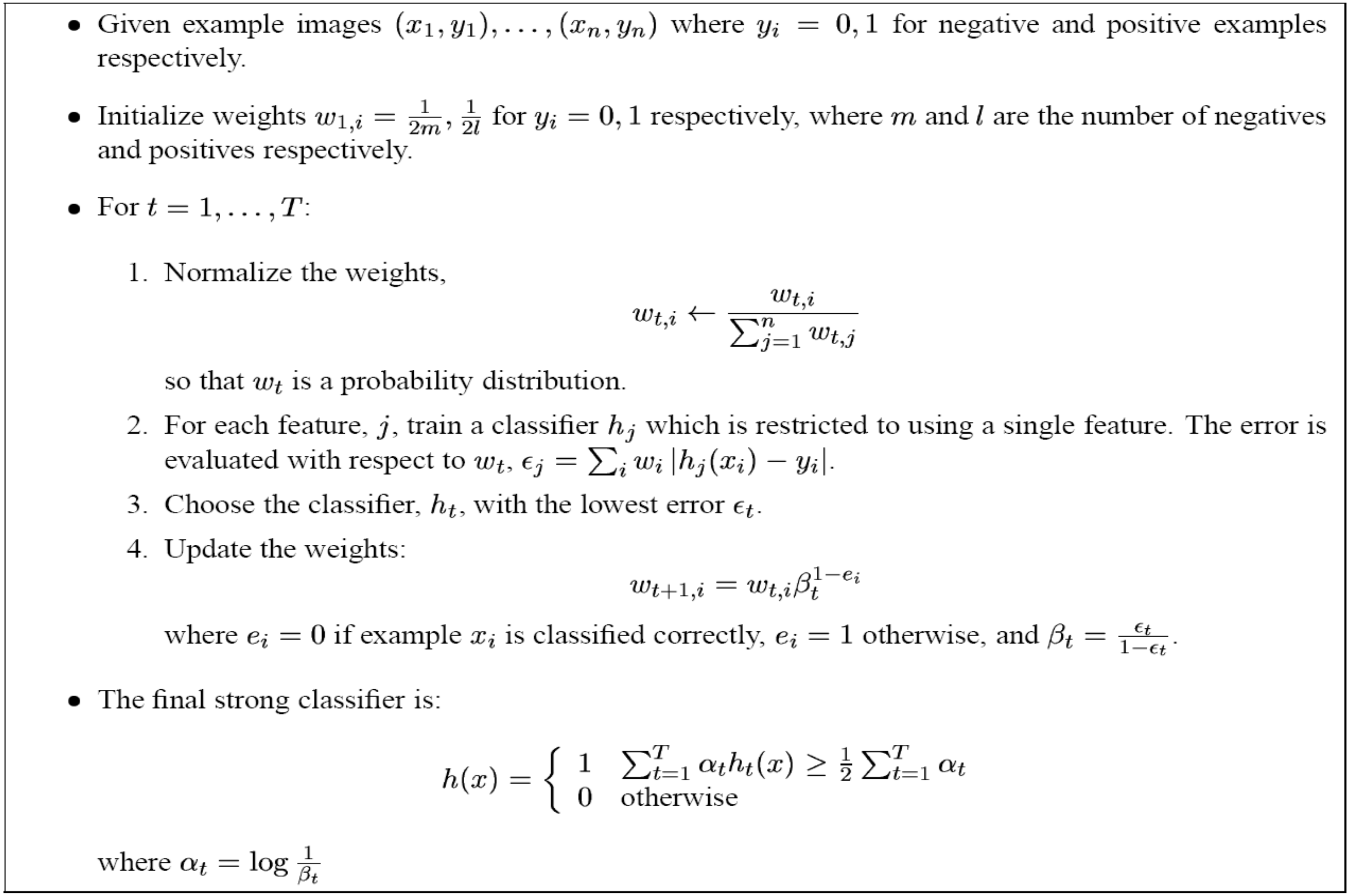
$$ii(4) + ii(1) – ii(2) – ii(3)$$

As a result two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references respectively.

Learning a classification function

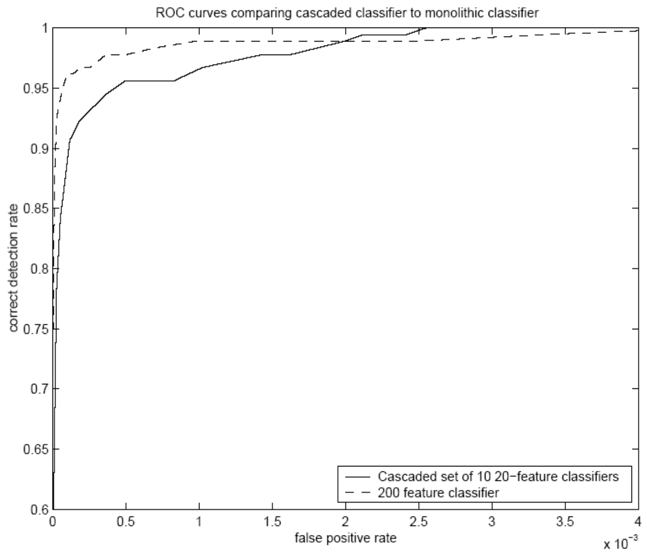
- Given a feature set and labeled training set of images one can apply several machine learning techniques.
- However, there is 45,396 features in each image sub-window, hence the computation of all features is computationally prohibitive.
- Classifier should combine a small subset of discriminative features so as to yield an effective classification.
- Challenge: Find these discriminant features.

AdaBoost for aggressive feature selection



- 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, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  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 $\alpha_t = \log \frac{1}{\beta_t}$

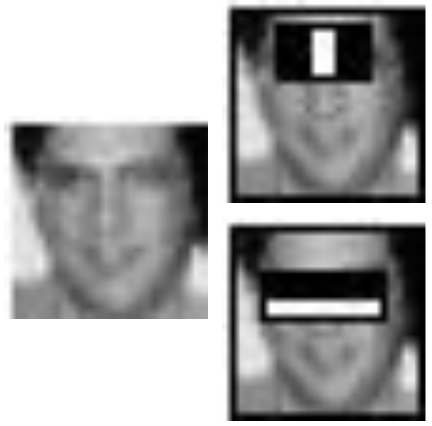
Performance of 200 feature face detector



The ROC curve of the constructed classifiers indicates that a reasonable detection rate of 0.95 can be achieved while maintaining an extremely low false positive rate of approximately 10^{-4}.

- First features selected by AdaBoost are meaningful and have high discriminative power

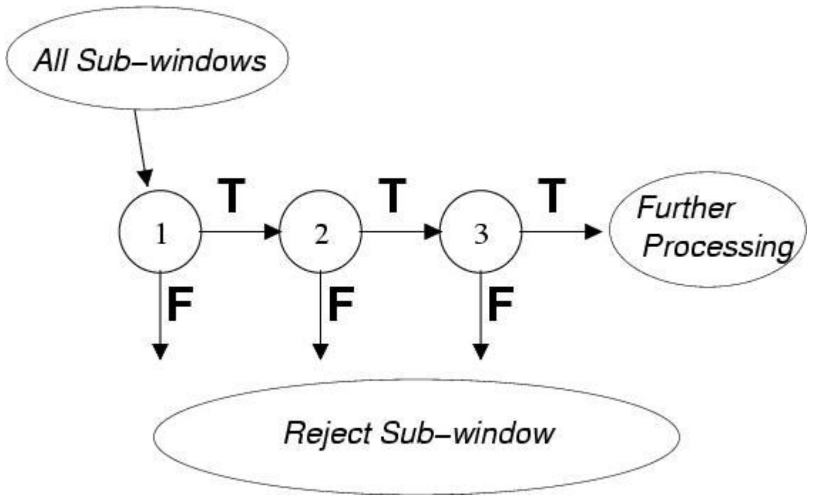
- By varying the threshold of the final classifier one can construct a two-feature classifier which has a detection rate of 1 and a false positive rate of 0.4.



Speed-up through the Attentional Cascade

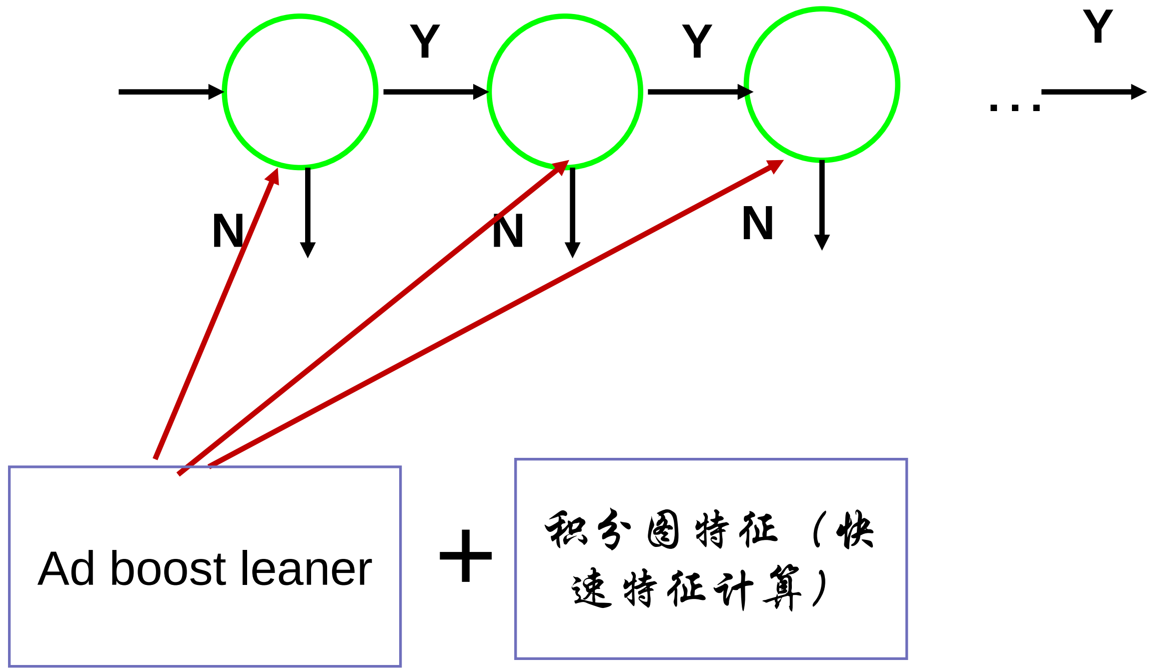
- Simple, boosted classifiers can reject many of the negative sub-windows while detecting all positive instances.
- Series of such simple classifiers can achieve good detection performance while eliminating the need for further processing of negative sub-windows.

Training: subsequent classifiers are trained only on examples which pass through all the previous classifiers.



Adaboost – 应用

- Cascade 分类器



Experiments (dataset for training)

- 4916 positive training examples were hand picked aligned, normalized, and scaled to a base resolution of 24x24
- 10,000 negative examples were selected by randomly picking sub-windows from 9500 images which did not contain faces

Experiments cont.

- The final detector had 32 layers and 4297 features total

Layer number	1	2	3 to 5	6 and 7	8 to 12	13 to 32
Number of feautures	2	5	20	50	100	200
Detection rate	100%	100%	-	-	-	-
Rejection rate	60%	80%	-	-	-	-

- Speed of the detector ~ total number of features evaluated
- On the MIT-CMU test set the average number of features evaluated per subwindow is 8 (out of 4297).
- The processing time of a 384 by 288 pixel image on a conventional personal computer is about .067 seconds.

Results

Testing of the final face detector was performed using the MIT+CMU frontal face test set which consists of:

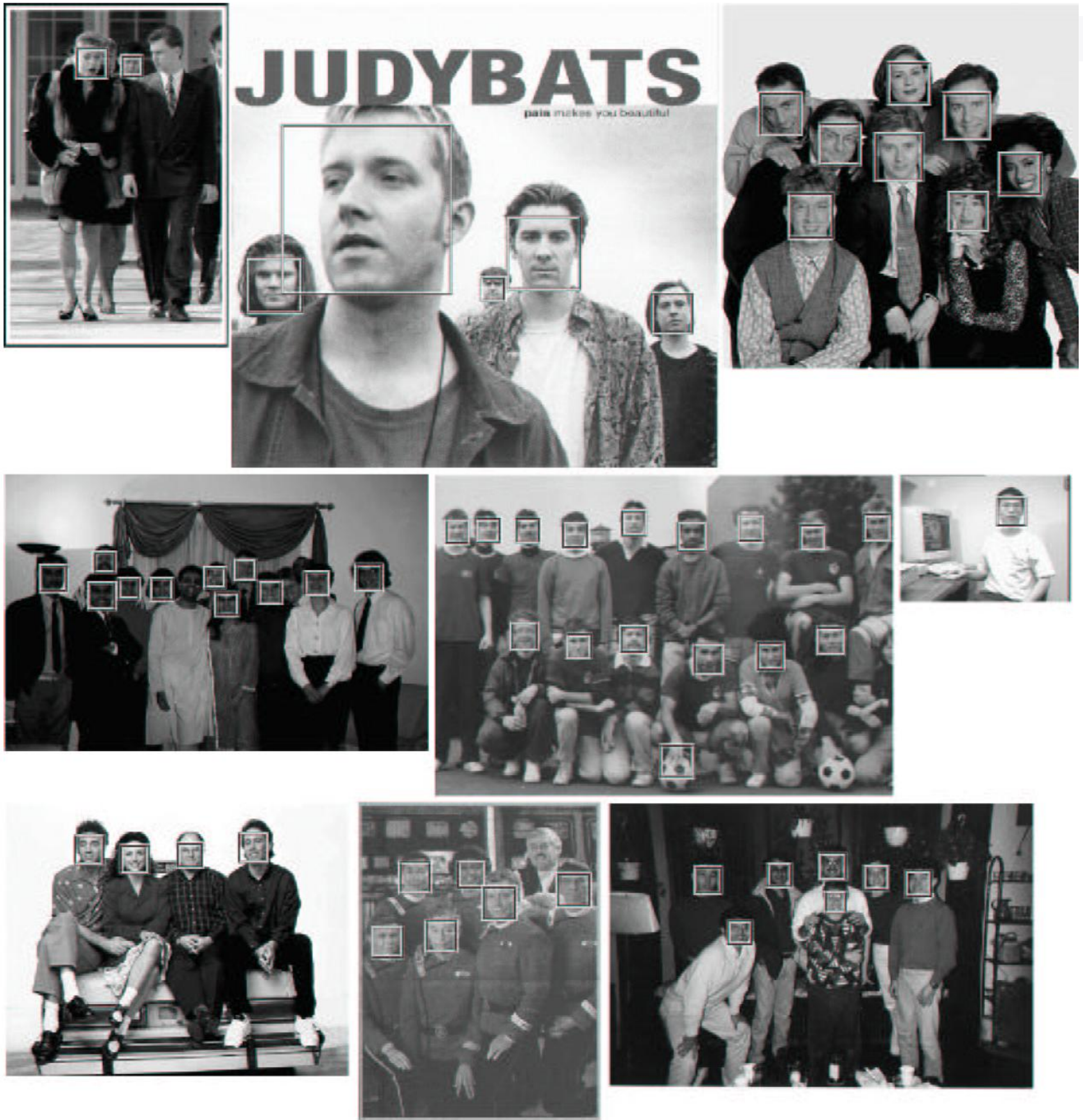
• 130 images

• 507 labeled frontal faces

Results in the table compare the performance of the detector to best face detectors known.

False detections	10	31	50	65	78	95	110	167	422
Viola-Jones	78.3%	85.2%	88.8%	89.8%	90.1%	90.8%	91.1%	91.8%	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	-	90.1%	89.9%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-	-	-
Roth-Yang-Ajuha	-	-	-	-	94.8%	-	-	-	-

Rowley at al.: use a combination of two neural networks (simple network for prescreening larger regions, complex network for detection of faces).


JUDYBATS
pain makes you beautiful

Object Detection Using the Statistics of Parts

Henry Schneiderman & Takeo Kanade

- AdaBoost based
- Parts based representation : Localized groups of discretized wavelet coefficients as features
- Likelihood obtained using probability tables and statistical independence of parts
- Uses likelihood ratio test classifier

$$\prod_{r} \frac{P_r(part_r|object)}{P_r(part_r|non\text{-}object)} > \lambda$$

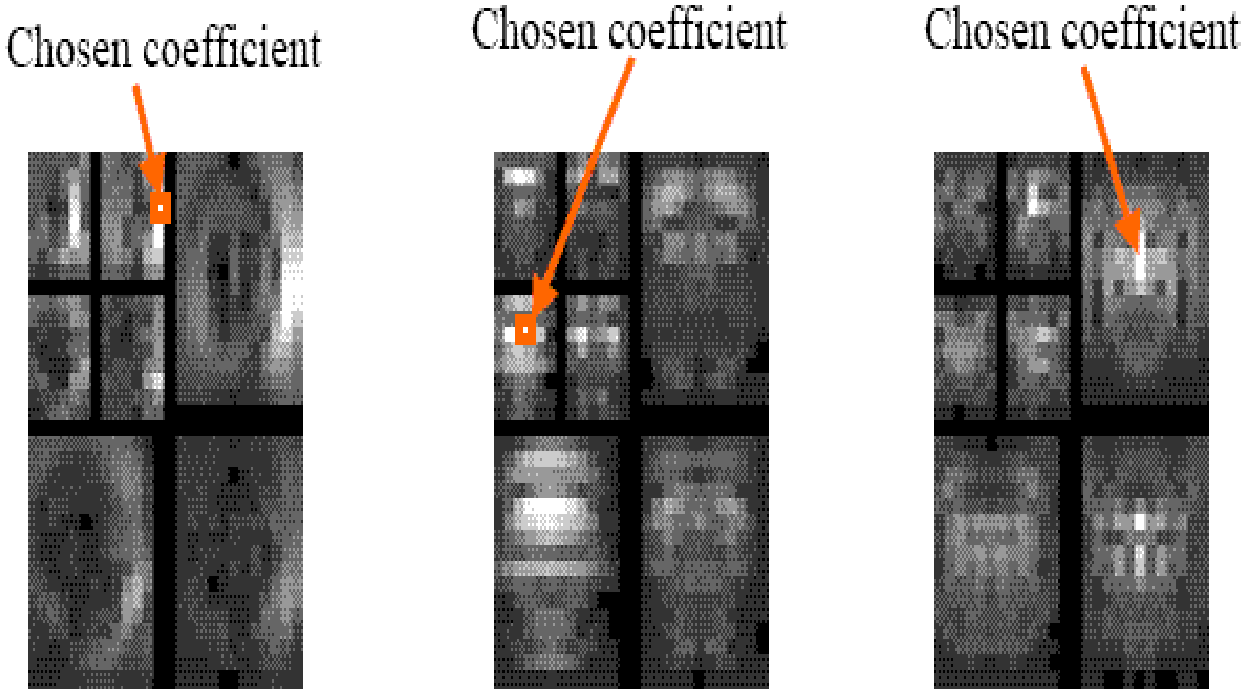
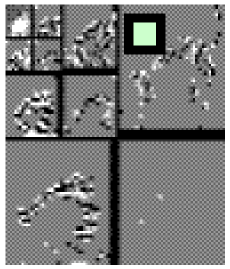
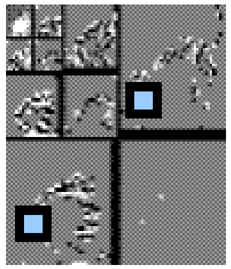
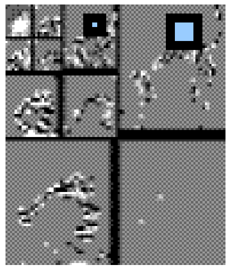


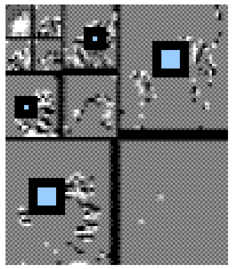
Figure 4: Pair-wise mutual information between a chosen coefficient location and all other locations in the wavelet transform for frontal faces

Parts are localized in position and frequency


Intra-subband

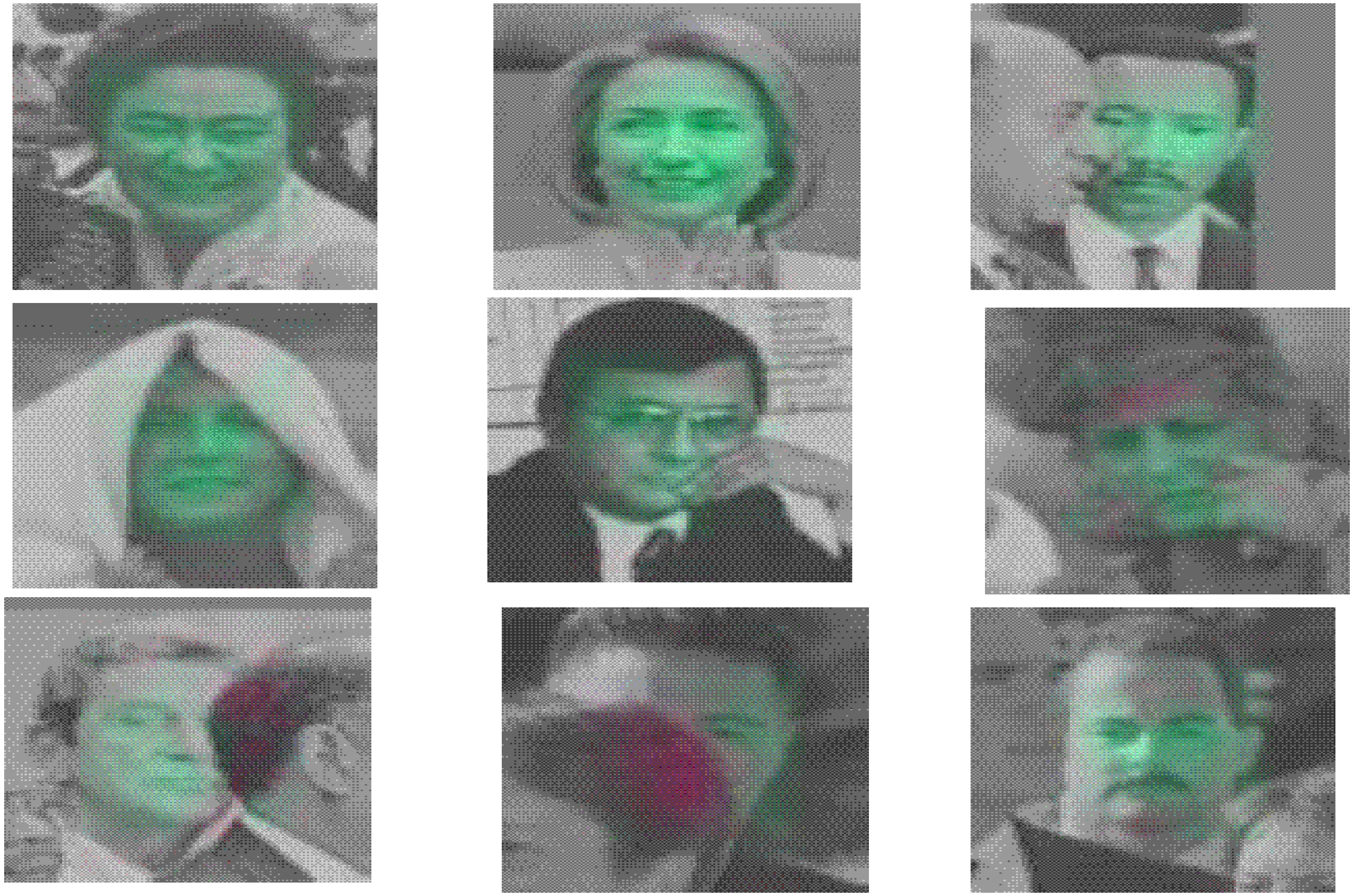

Inter-orientation


Inter-frequency


Inter-frequency/
Inter-orientation

- Algorithm uses exhaustive search across position, size, orientation, alignment and intensity.
- Course to Fine Evaluation
- Wavelet Transform coefficients can be reused for multiple scales
- Color preprocessing
- Time – 5 s for 240x256 image (PII 450 MHz)

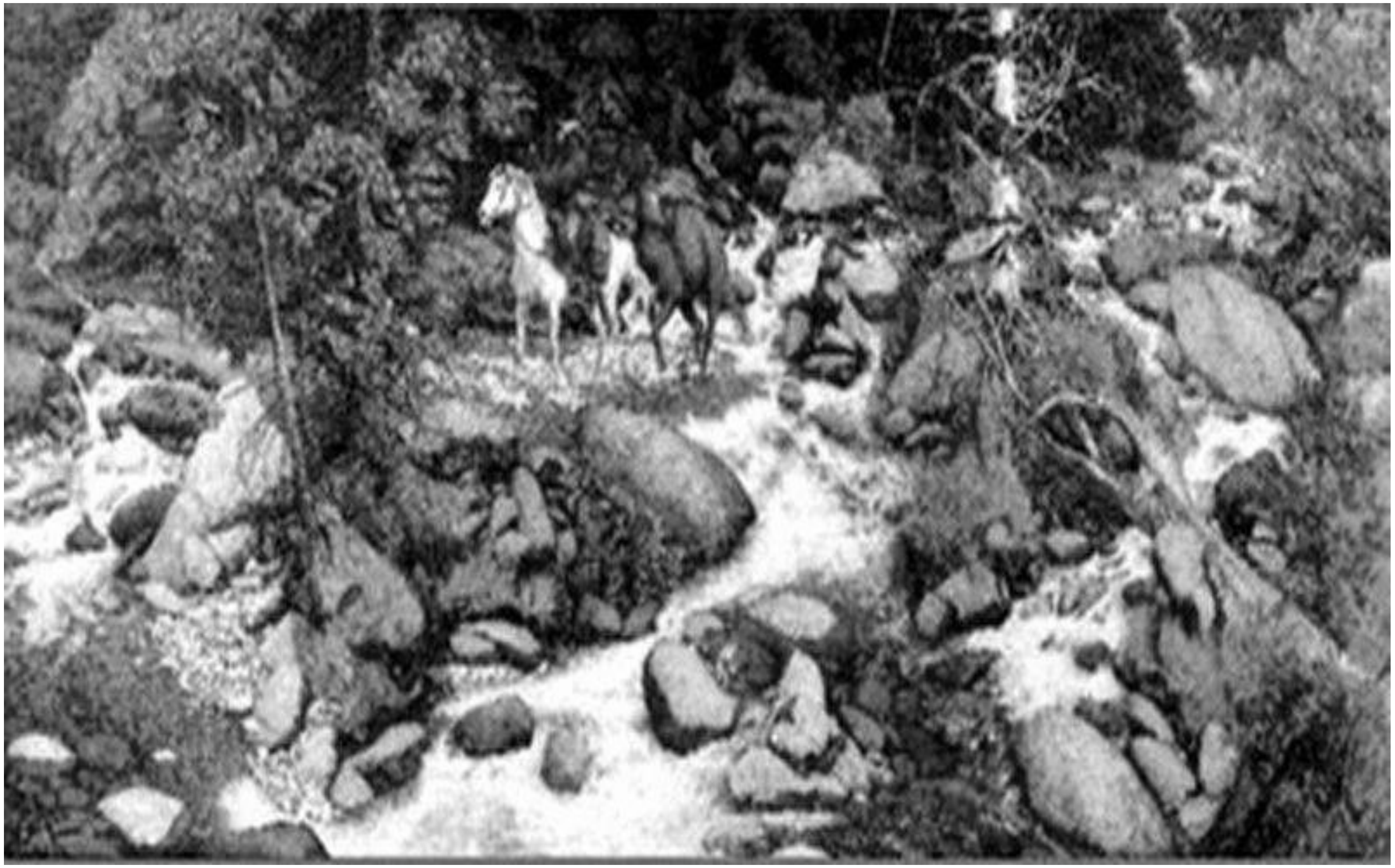
What are the important parts?



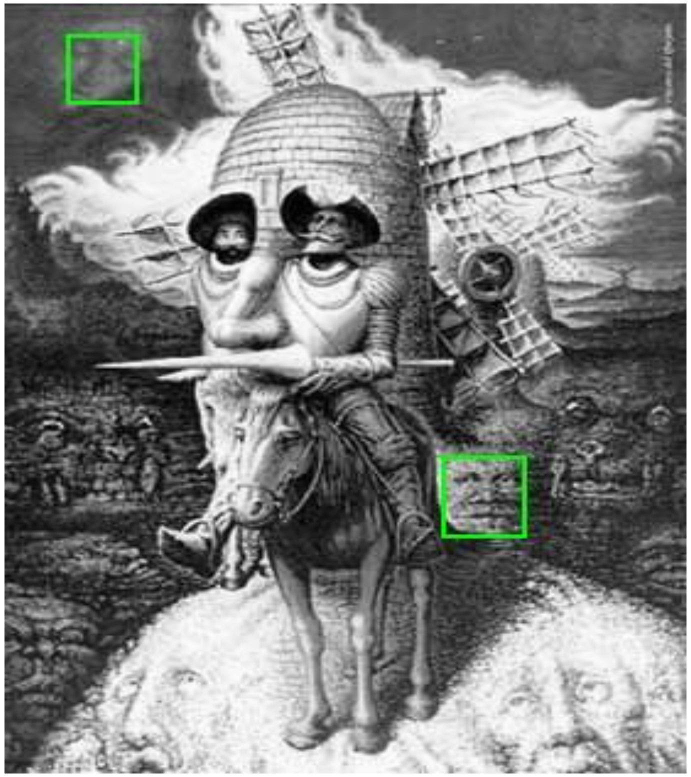
Conclusions

- The Viola&Jones paper uses very simple features which are very fast to compute.
- Integral image representation is used to speed up the feature calculation.
- AdaBoost used for improving the classification and efficient feature selection.
- A cascade of classifiers is used to minimize the computation without sacrificing the classification performance.
- The final face detector is comparable in performance to other existing classifiers, but orders of magnitude faster.
- The Schneiderman & Kanade paper uses part based features using wavelet coefficients.
- Classifier is based on likelihood ratio test. The likelihoods are obtained from probability tables constructed while training.
- AdaBoost is used to improve the performance..

How many faces in this picture ?



What about this ?



A demo of Viola and Jones

- http://mplab.ucsd.edu/

思考题:

- ➢与SVM的区别?
- ➢针对多类问题如何扩展?