Boosting: Combining Classifiers

Zhang Hongxin
zhx@cad.zju.edu.cn

State Key Lab of CAD&CG, ZJU
2005-06-16

The most material of this part come from:
http://sifaka.cs.uiuc.edu/taotao/stat/chap10.ppt

Boosting

- **INTUITION**
  - *Combining Predictions of an ensemble is more accurate than a single classifier*
- Reasons
  - Easy to find quite correct “rules of thumb” however hard to find single highly accurate prediction rule.
  - If the training examples are few and the hypothesis space is large then there are several equally accurate classifiers.
  - Hypothesis space does not contain the true function, but it has several good approximations.
  - Exhaustive global search in the hypothesis space is expensive so we can combine the predictions of several locally accurate classifiers.

Cross Validation

- k-fold Cross Validation
  - Divide the data set into k sub samples
  - Use k-1 sub samples as the training data and one sub sample as the test data.
  - Repeat the second step by choosing different sub samples as the testing set.
- Leave one out Cross validation
  - Used when the training data set is small.
  - Learn several classifiers each one with one data sample left out
  - The final prediction is the aggregate of the predictions of the individual classifiers.

Bagging

- Generate a random sample from training set
- Repeat this sampling procedure, getting a sequence of K independent training sets
- A corresponding sequence of classifiers C1,C2,…,Ck is constructed for each of these training sets, by using the same classification algorithm
- To classify an unknown sample X, let each classifier predict.
- The Bagged Classifier C* then combines the predictions of the individual classifiers to generate the final outcome. (sometimes combination is simple voting)

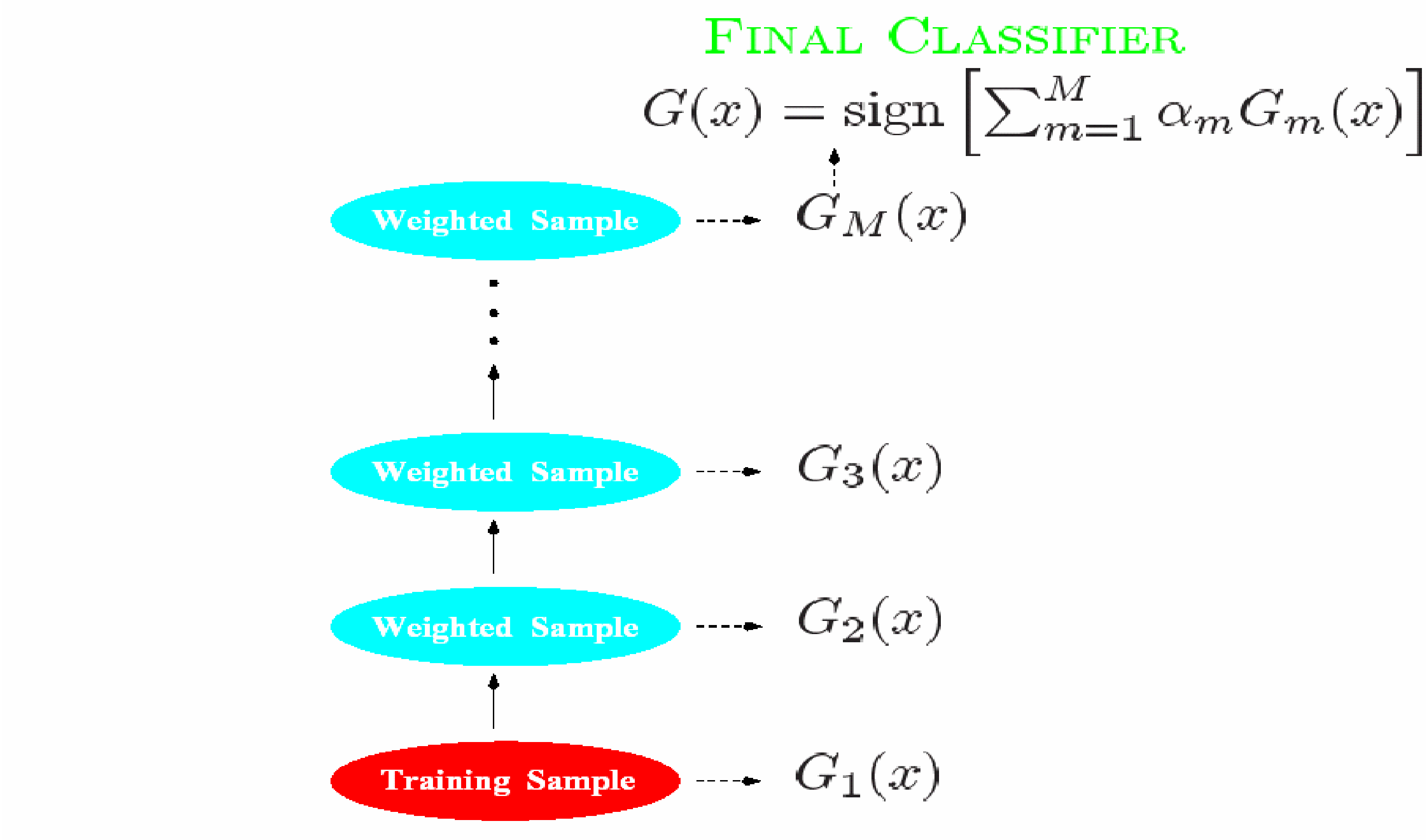
Boosting

- The final prediction is a combination of the prediction of several predictors.
- Differences between Boosting and previous methods?
  - It is iterative.
  - Boosting: Successive classifiers depends upon its predecessors.
    - Previous methods : Individual classifiers were independent.
  - Training Examples may have unequal weights.
  - Look at errors from previous classifier step to decide how to focus on next iteration over data
  - Set weights to focus more on ‘hard’ examples. (the ones on which we committed mistakes in the previous iterations)

Boosting(Algorithm)

- $W(x)$ is the distribution of weights over the N training points $\sum W(x_i)=1$
- Initially assign uniform weights $W_0(x) = 1/N$ for all x, *step* $k=0$
- At each iteration k :
  - Find best weak classifier $C_k(x)$ using weights $W_k(x)$
  - With error rate ε_k and based on a loss function:
    - weight α_k the classifier C_k's weight in the final hypothesis
    - For each x_i , update weights based on ε_k to get $W_{k+1}(x_i)$
- $C_{FINAL}(x) = \text{sign}\ [\ \sum \alpha_i C_i(x)\ ]$

Boosting (Algorithm)

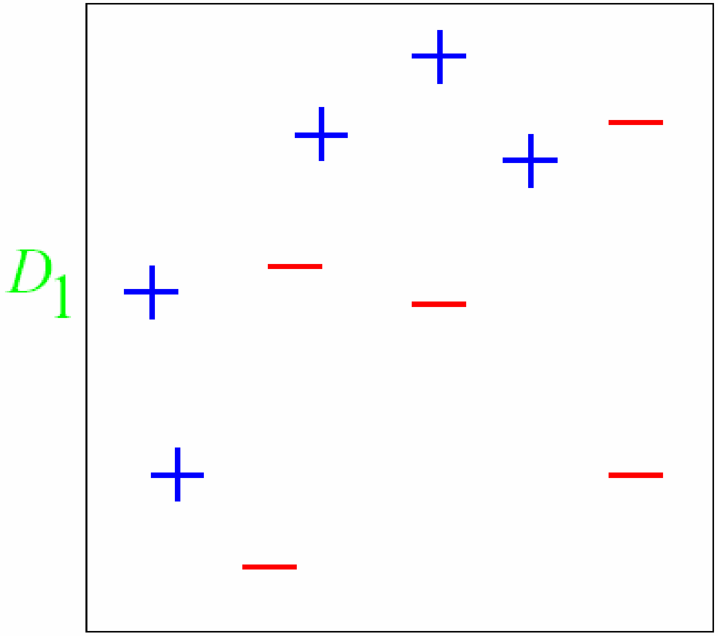


AdaBoost(Algorithm)

- $W(x)$ is the distribution of weights over the N training points
  $\sum W(x_i)=1$
- Initially assign uniform weights $W_0(x) = 1/N$ for all x.
- At each iteration k :
  - Find best weak classifier $C_k(x)$ using weights $W_k(x)$
  - Compute ε_k the error rate as
    $\varepsilon_k = [\sum W(x_i) \cdot \mathrm{I}(y_i \neq C_k(x_i))] \,/\, [\sum W(x_i)]$
  - weight α_k the classifier C_k's weight in the final hypothesis Set
    $\alpha_k = \log((1 - \varepsilon_k)/\varepsilon_k)$
  - For each x_i, $W_{k+1}(x_i) = W_k(x_i) \cdot \exp[\alpha_k \cdot \mathrm{I}(y_i \neq C_k(x_i))]$
- $C_{FINAL}(x) = \mathrm{sign}[\sum \alpha_i C_i(x)]$

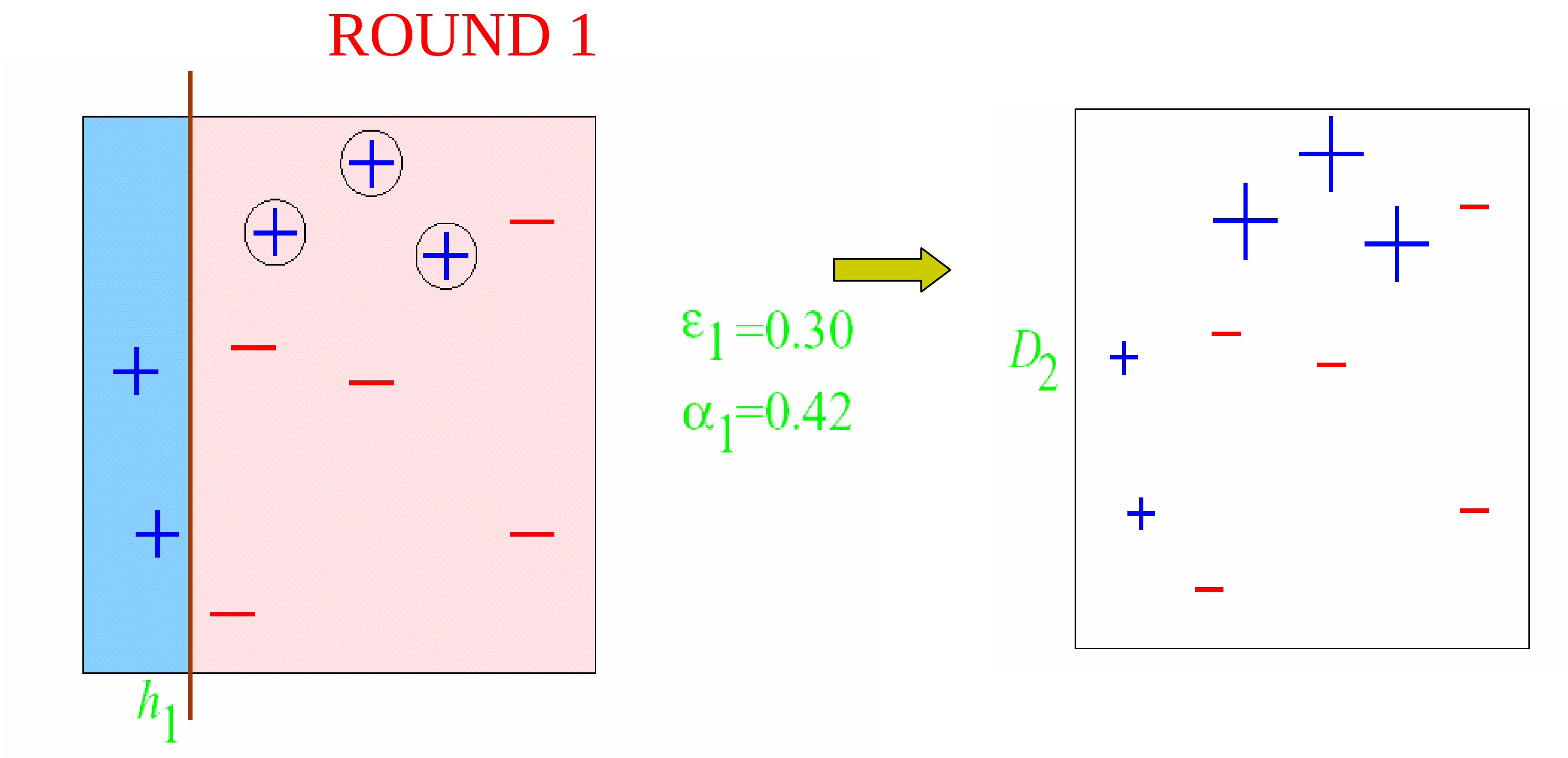
$L(y, f(x)) = \exp(-y \cdot f(x))$ - the exponential loss function

AdaBoost(Example)



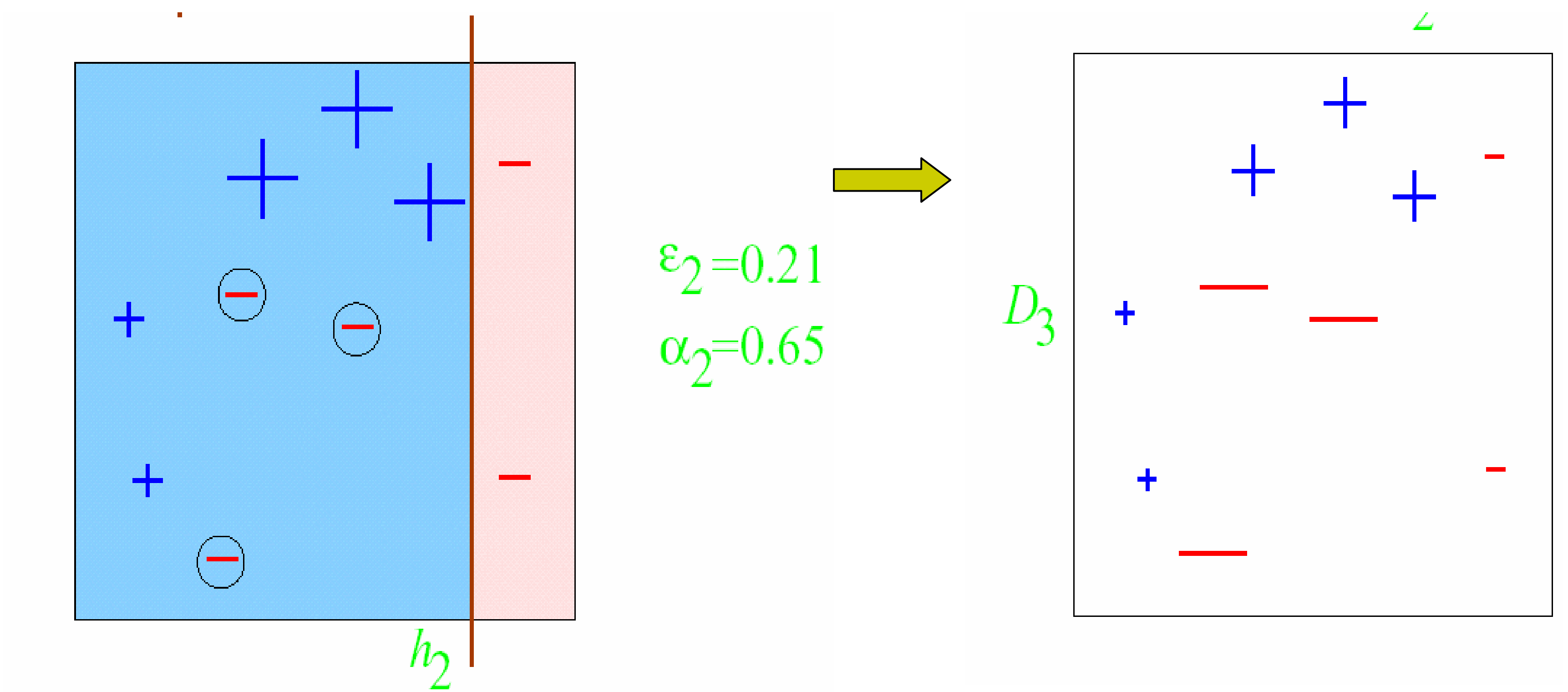
Original Training set : Equal Weights to all training samples

AdaBoost(Example)



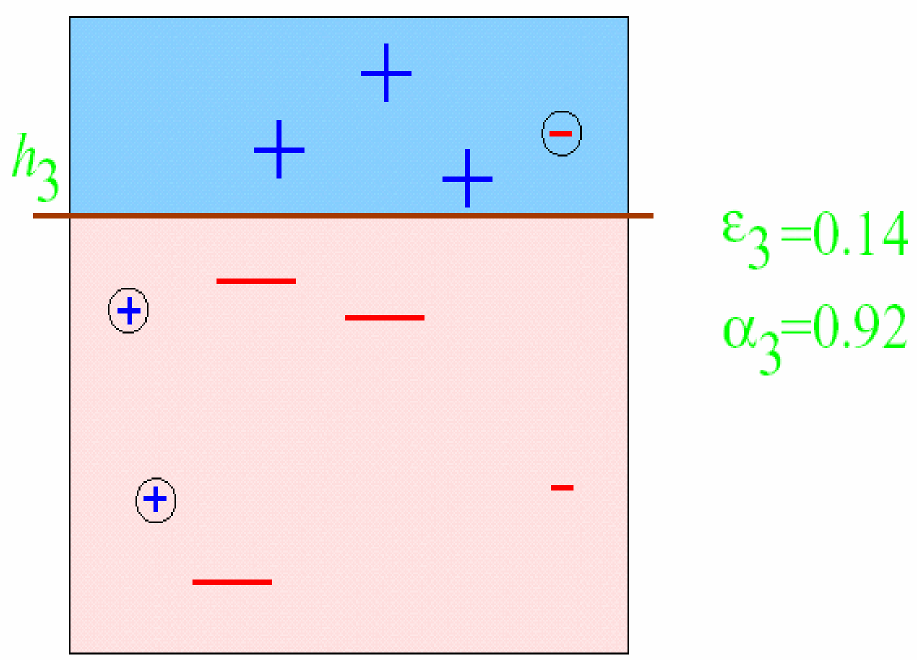
AdaBoost(Example)

ROUND 2



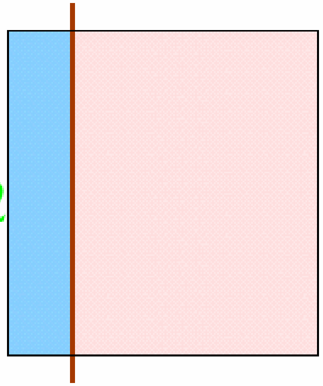
AdaBoost(Example)

ROUND 3

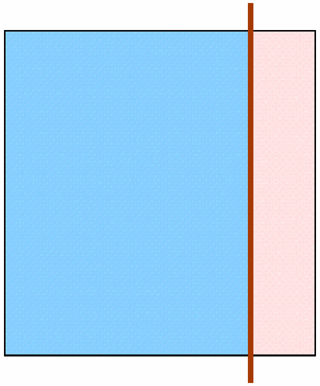


AdaBoost(Example)

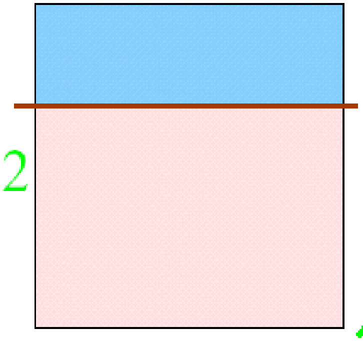
$H_{\text{final}} = \text{sign}\Big(0.42$



$+\ 0.65$



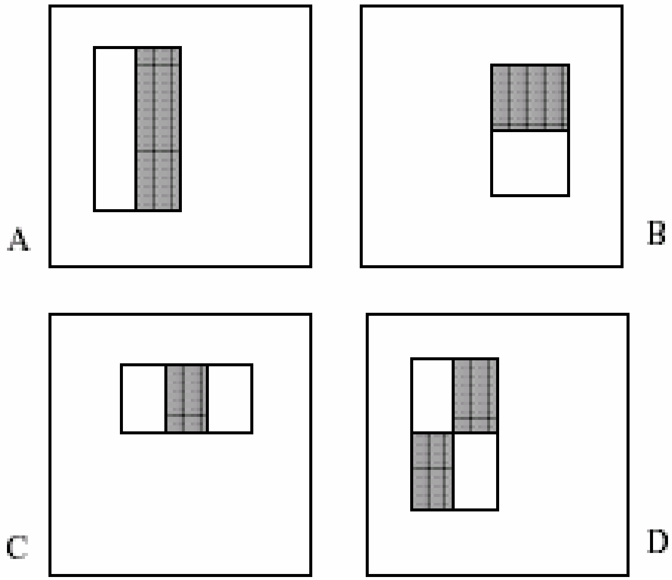
$+\ 0.92$



$\Big)$

AdaBoost Case Study:
Rapid Object Detection using a Boosted Cascade of Simple Features(CVPR01)

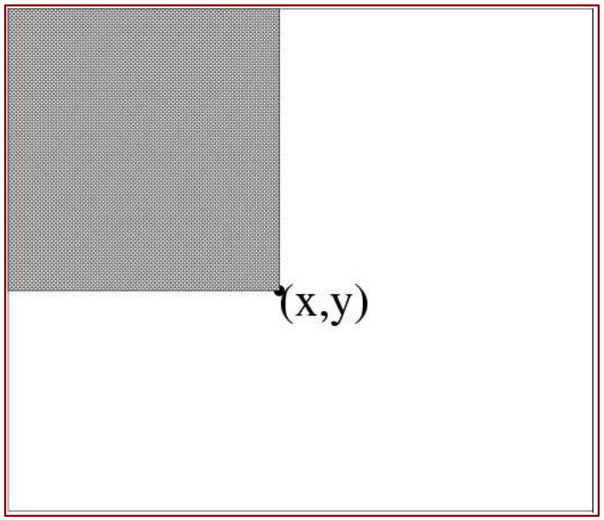
- Object Detection
- Features
  - *two-rectangle*
  - *three-rectangle*
  - *four-rectangle*



Size: 24x24

Feature: 180,000

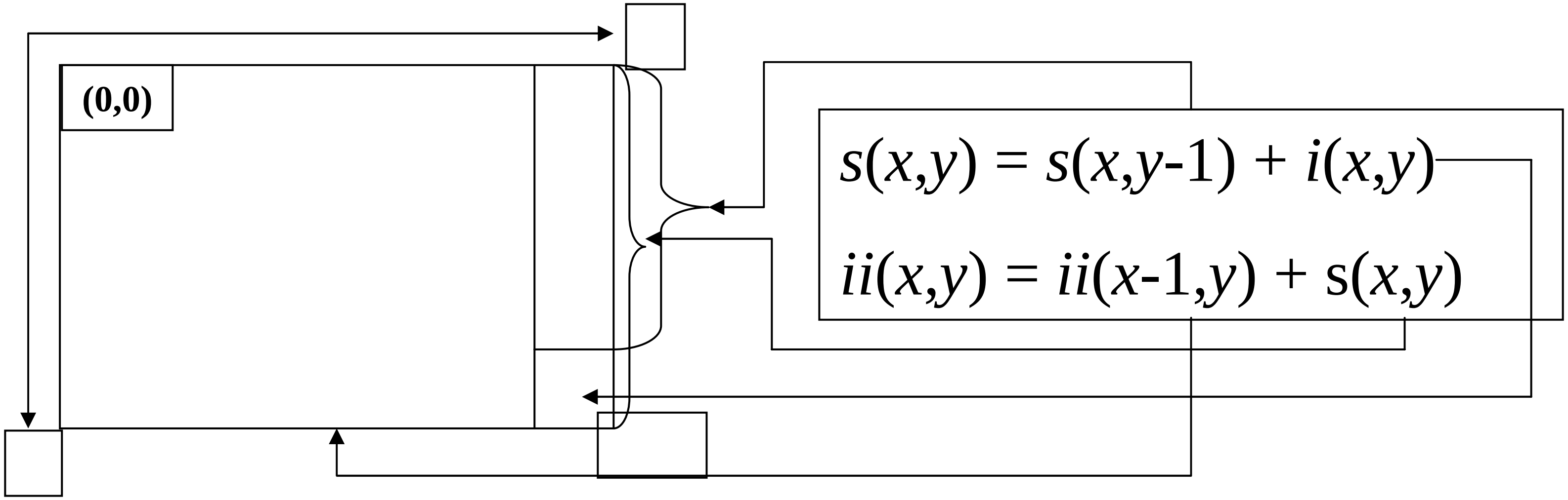
* Integral Image



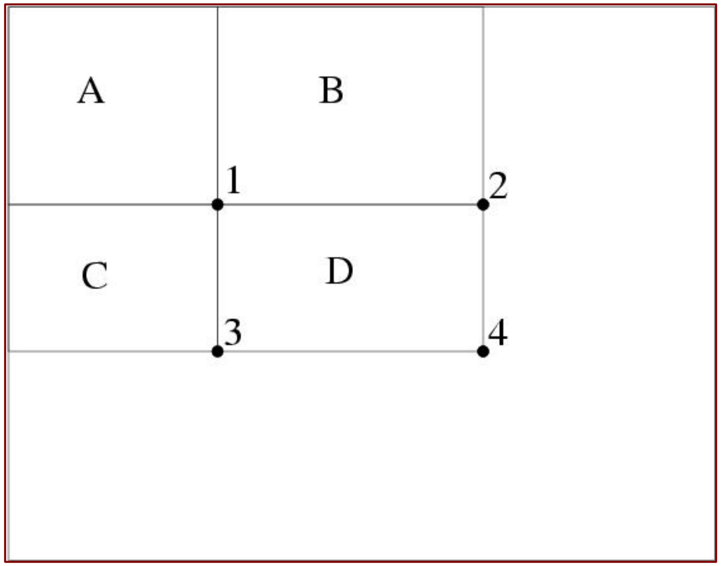
Definition: The integral image at location (x,y) contains the sum of the pixels above and to the left of (x,y) , inclusive:

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

Using the following pair of recurrences:



- Features Computation

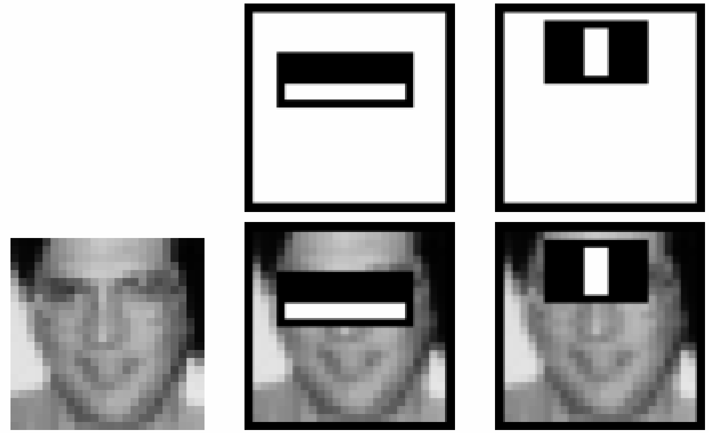


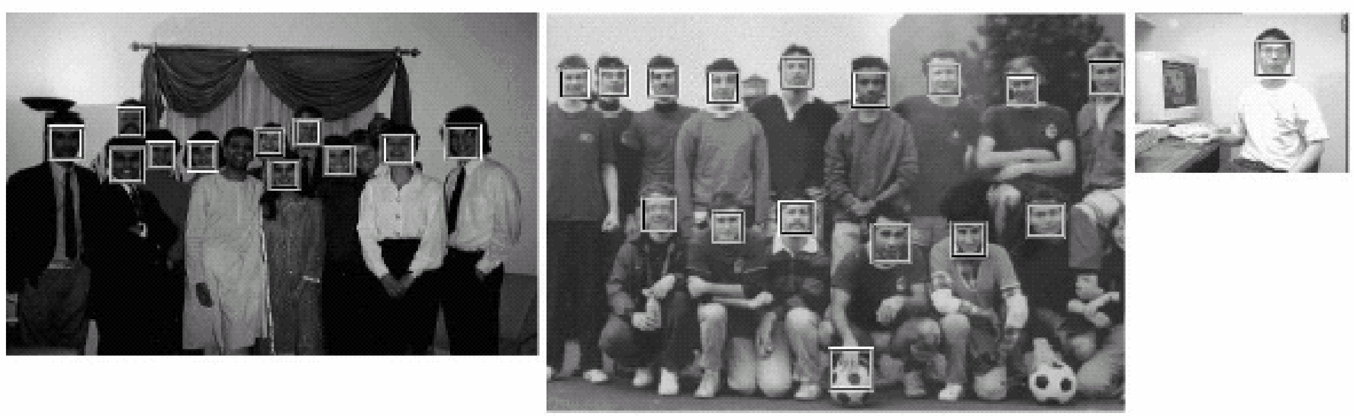
Using the integral image any rectangular sum can be computed in four array references

ii(4) + ii(1) – ii(2) – ii(3)

AdaBoost algorithm for classifier learning

- 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 $\alpha_t = \log \frac{1}{\beta_t}$





Homework

- Implement this CVPR paper.
  - Hint: You can use OpenCV.

Thank you