Adaboost for faces. Material

Gonzalo Vaca-Castano

Adaboost for faces paper

- Robust Real-Time Face Detection. International Journal of Computer Vision 57(2), 2004. Paul Viola, and Mike Jones
- Rapid object detection using a boosted cascade of simple features. Viola P., And Jones, M. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Dec. 2001).

Boosting

- Defines a classifier using an additive model:

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

Strong classifier

Features vector

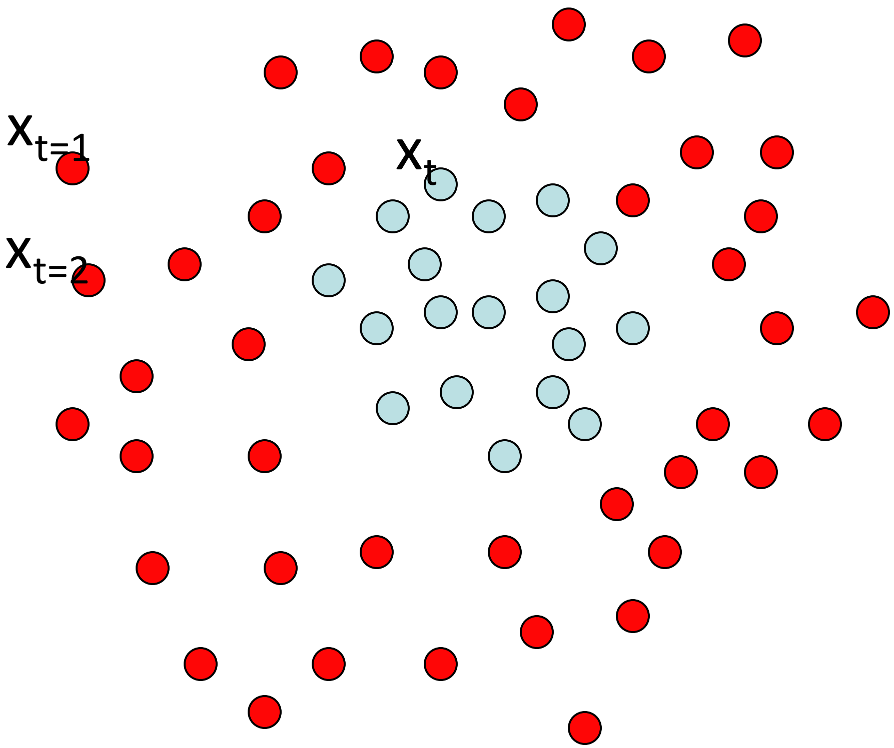
Weight

Weak classifier

Slide Credits: Andras Ferencz

Boosting

- It is a sequential procedure:



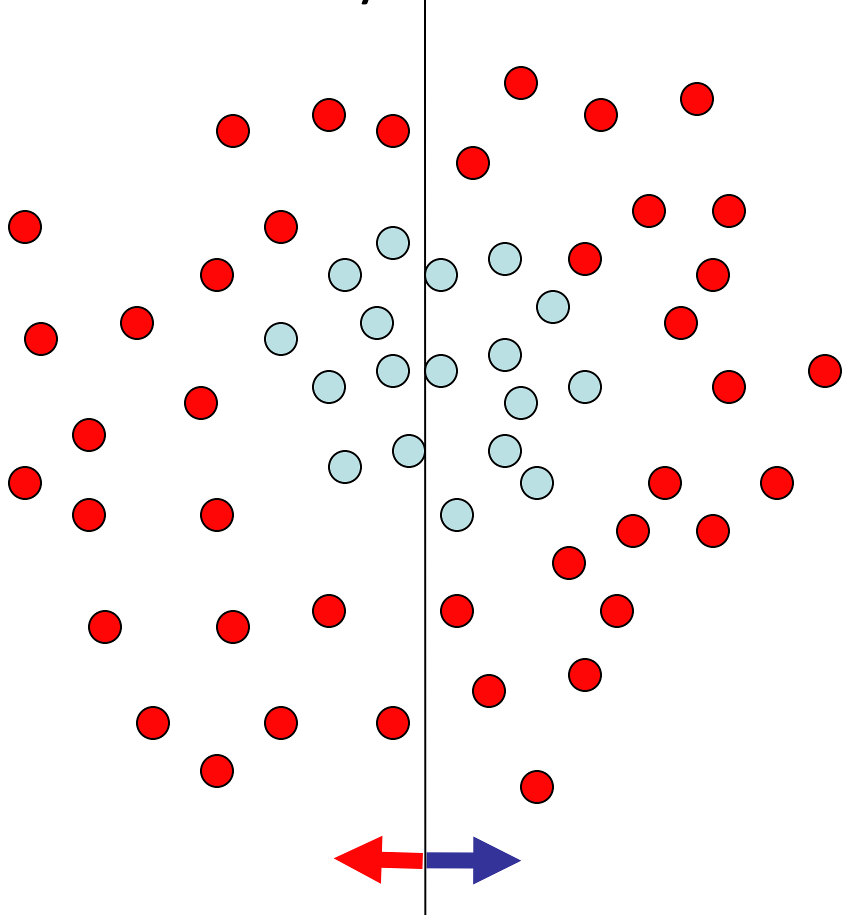
Each data point has

a class label: $y_t = \begin{cases} +1 \ (\ ) \\ -1 \ (\ ) \end{cases}$

and a weight:
$w_t = 1$

Toy example

Weak learners from the family of lines



Each data point has a class label:

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

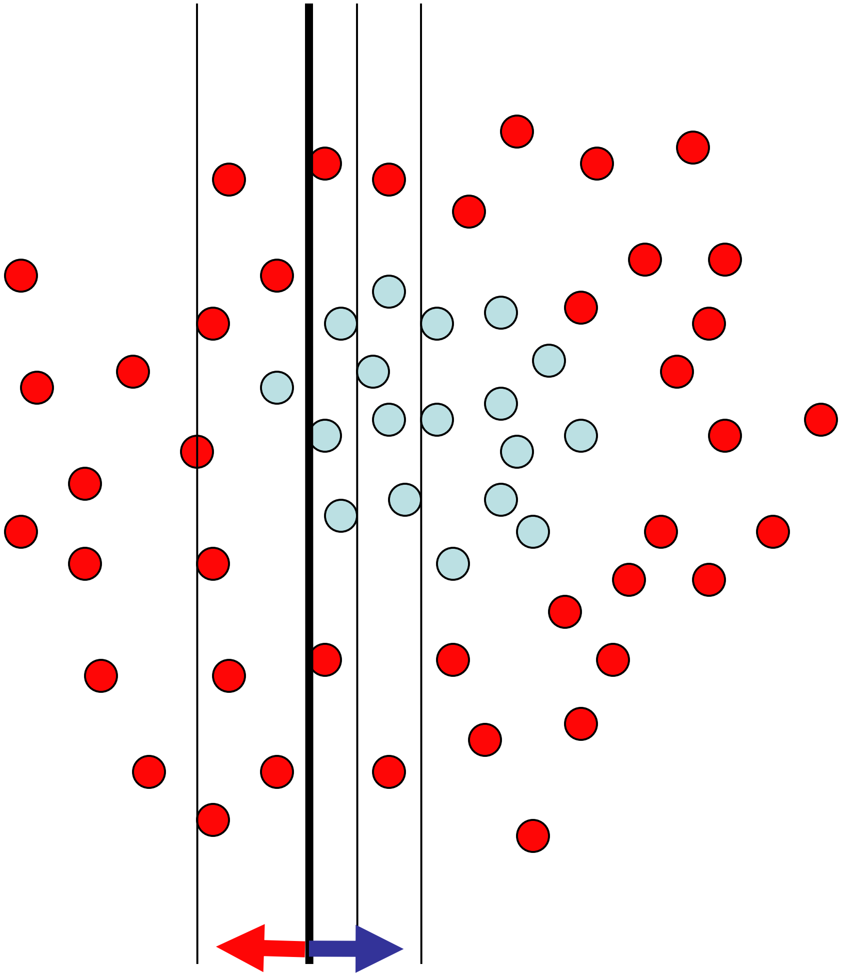
and a weight:

$w_t = 1$

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

Slide Credits: Andras Ferencz

Toy example



Each data point has a class label:

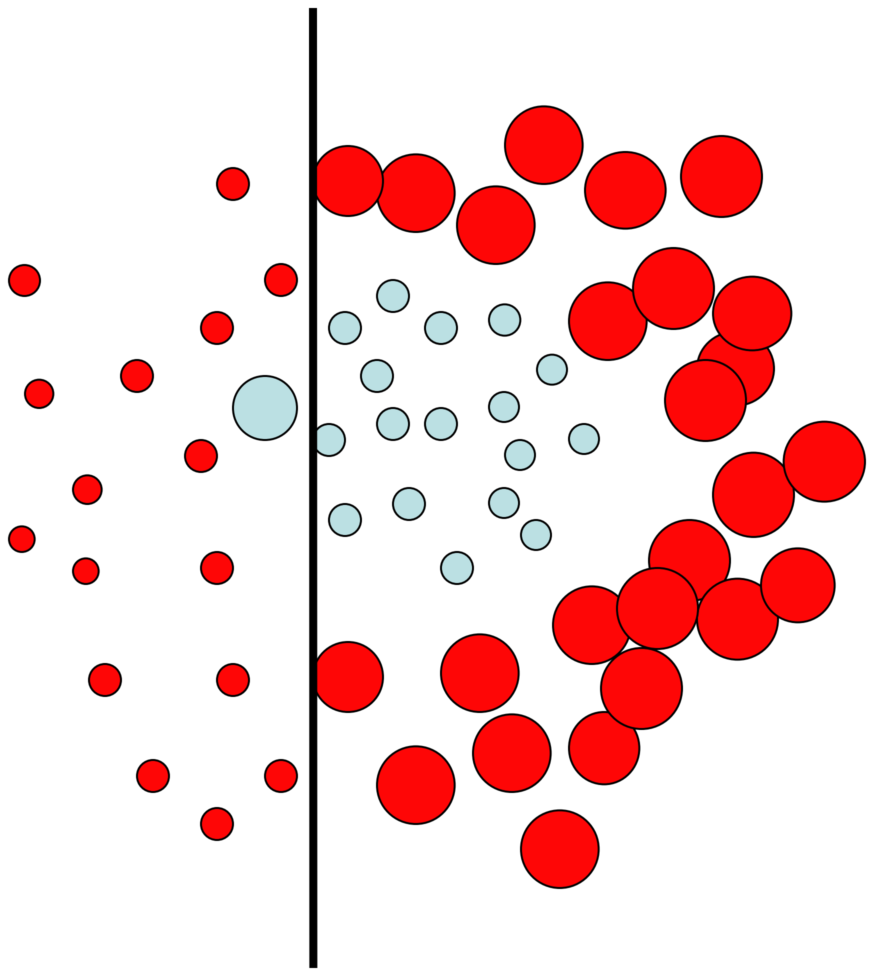
$$y_t = \begin{cases} +1 \; (\bullet) \\ -1 \; (\circ) \end{cases}$$

and a weight:
$w_t = 1$

This one seems to be the best

This is a '**weak classifier**': It performs slightly better than chance

Toy example



Each data point has a class label:

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

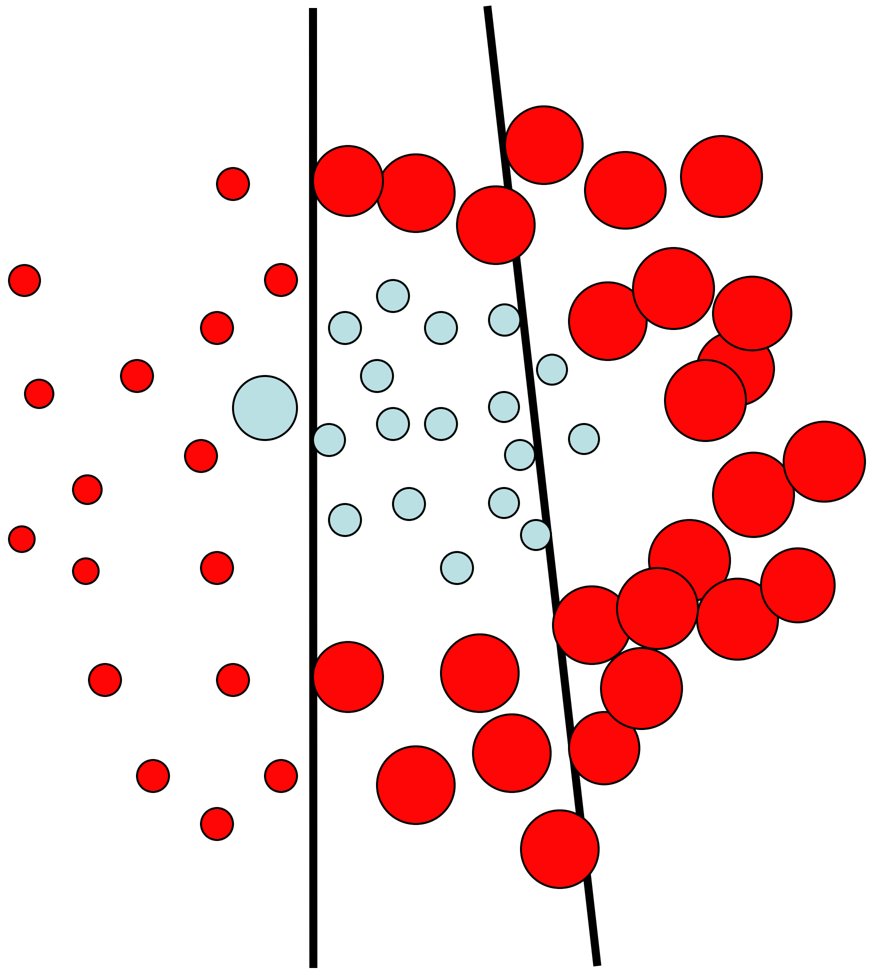
We update the weights:

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

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

Slide Credits: Andras Ferencz

Toy example



Each data point has a class label:

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

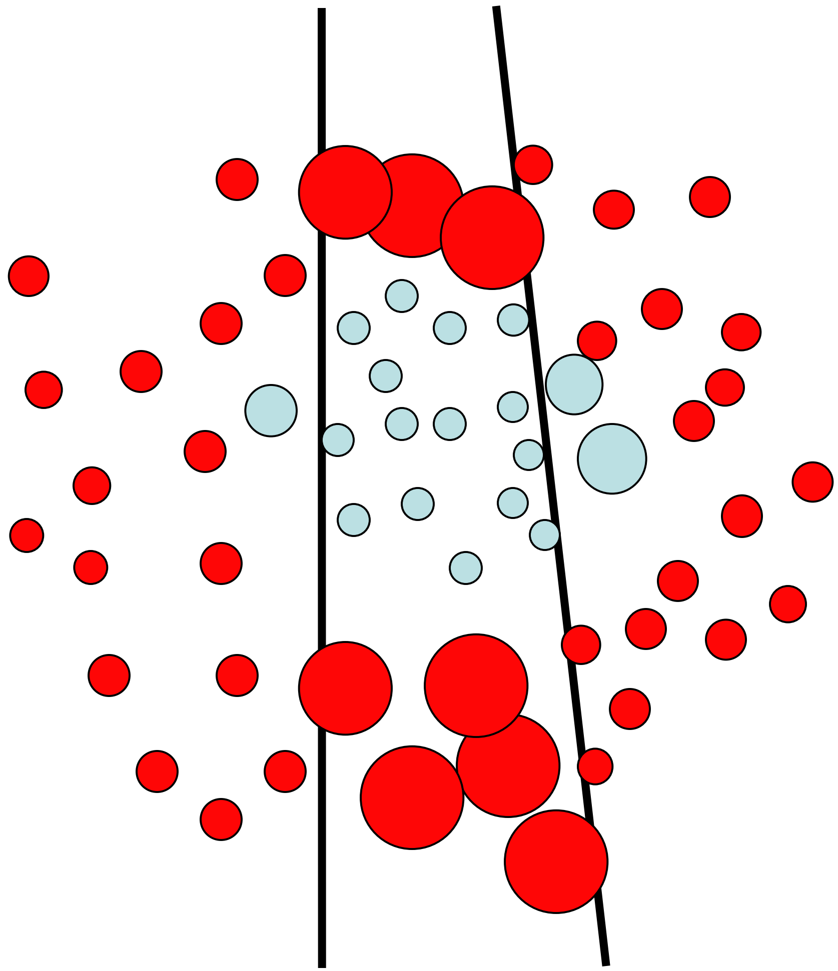
We update the weights:

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

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

Slide Credits: Andras Ferencz

Toy example



Each data point has a class label:

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

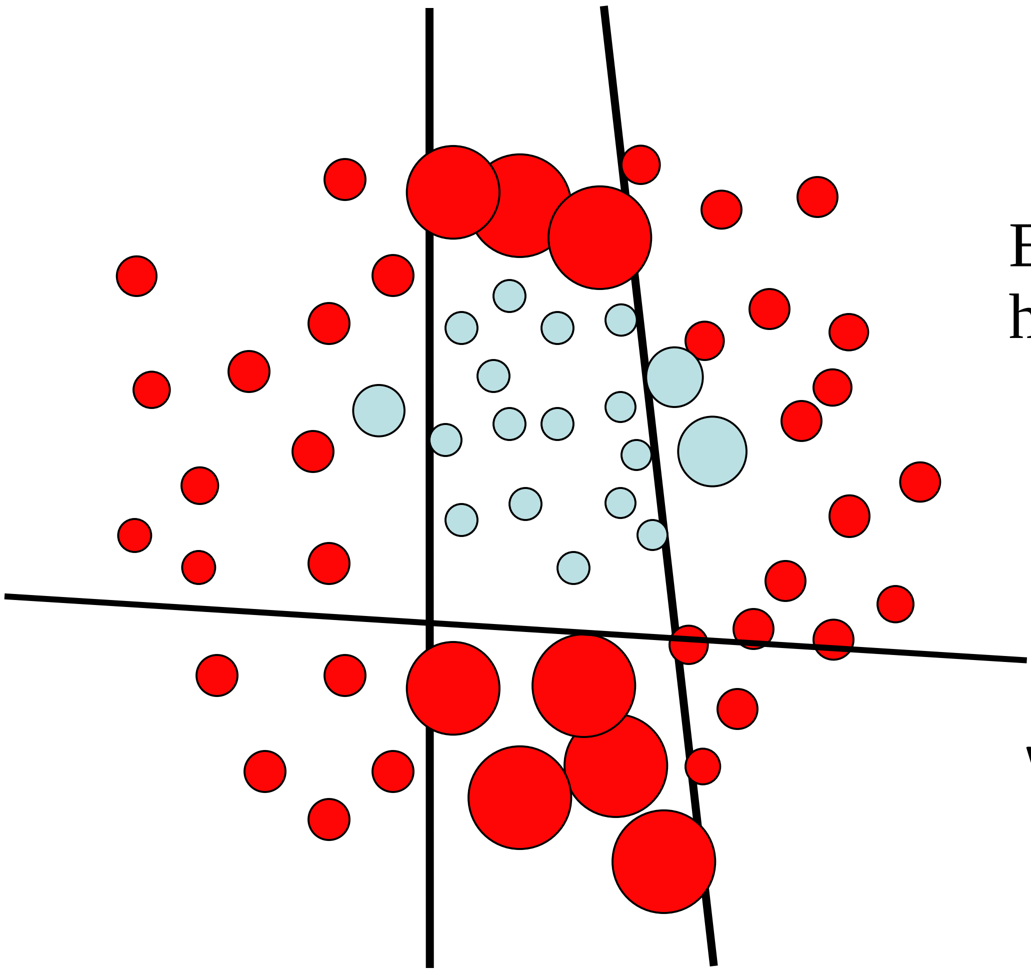
We update the weights:

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

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

Slide Credits: Andras Ferencz

Toy example



Each data point has a class label:

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

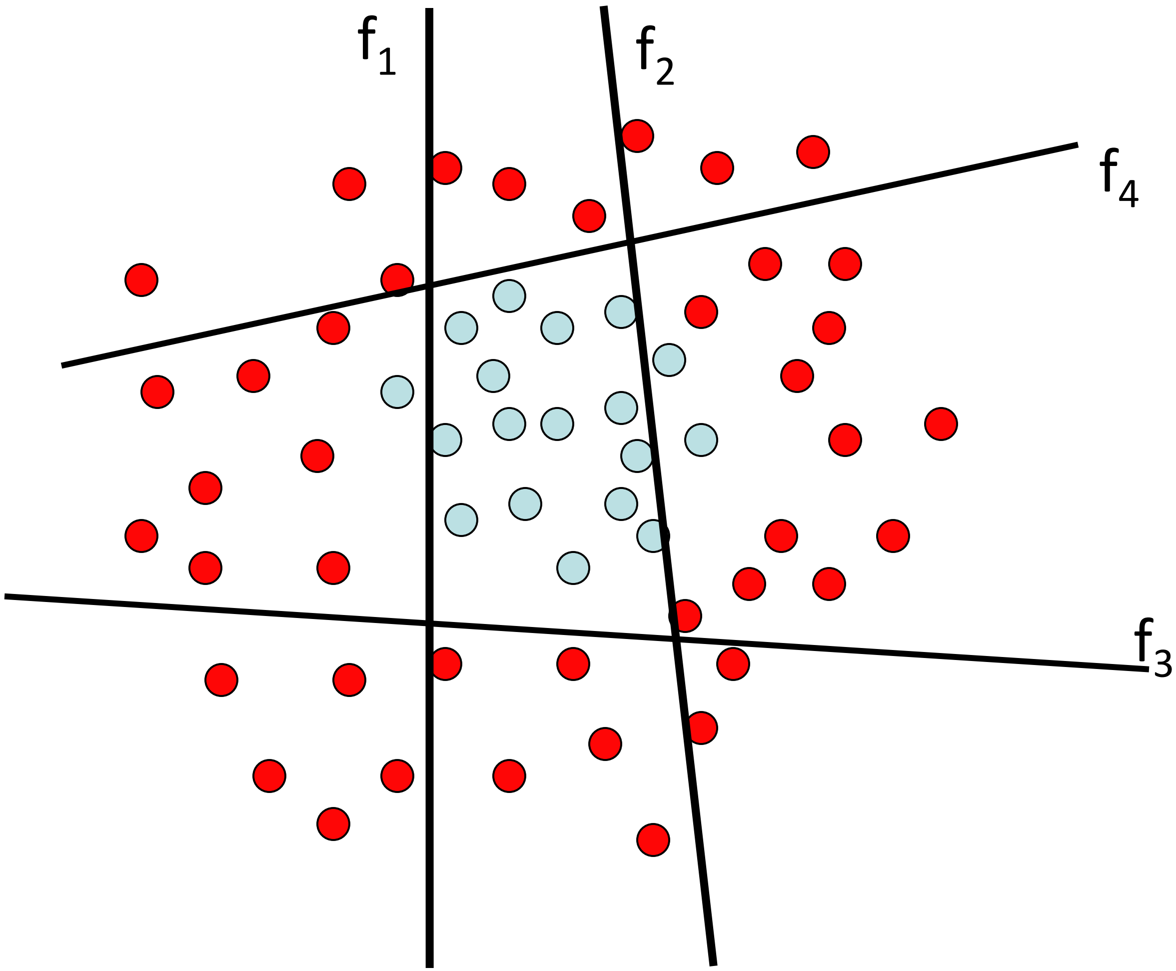
We update the weights:

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

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

Slide Credits: Andras Ferencz

Toy example



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

Slide Credits: Andras Ferencz

AdaBoost Algorithm

Given: m examples $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y=\{-1, +1\}$

Initialize $D_1(i) = 1/m$

For t = 1 to T

1. Train learner $\boldsymbol{h_t}$ with min error $\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$
2. Compute the hypothesis weight $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
3. For each example i = 1 to m

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

Output

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

The goodness of h_t is calculated over D_t and the bad guesses.

The weight **Ada**pts. The bigger ε_t becomes the smaller α_t becomes.

Boost example if incorrectly predicted.

Z_t is a normalization factor.

Linear combination of models.

Slide Credits: Andras Ferencz

Algorithm

Train

- 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, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
  2. Select the best weak classifier with respect to the weighted error
  $$\epsilon_t = \min_{f,p,\theta} \sum_i w_i \, | h(x_i, f, p, \theta) - y_i | .$$
  See Section 3.1 for a discussion of an efficient implementation.
  3. Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t, and θ_t are the minimizers of 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:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Test

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

$(h(x, f, p, \theta))$ thus consists of a feature (f), a threshold (θ) and a polarity (p) indicating the direction of the inequality:

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta \\ 0 & \text{otherwise} \end{cases}$$

Weak Classifiers

- 4 types of rectangular filters (24x24 picture)

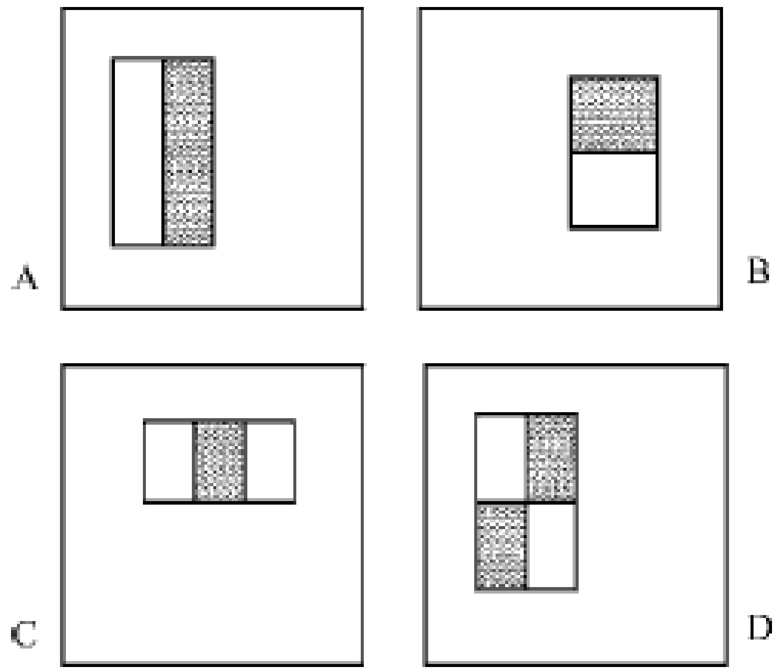


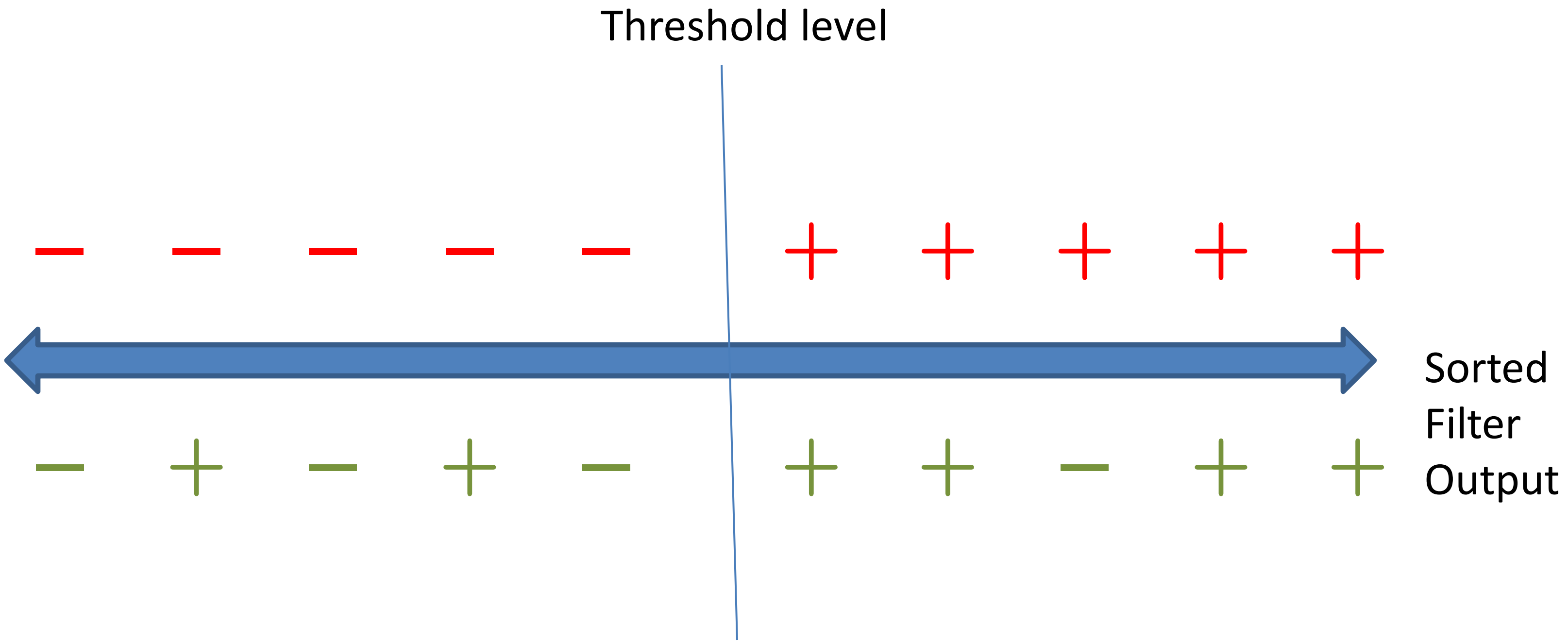
Figure 1. Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

Step 2

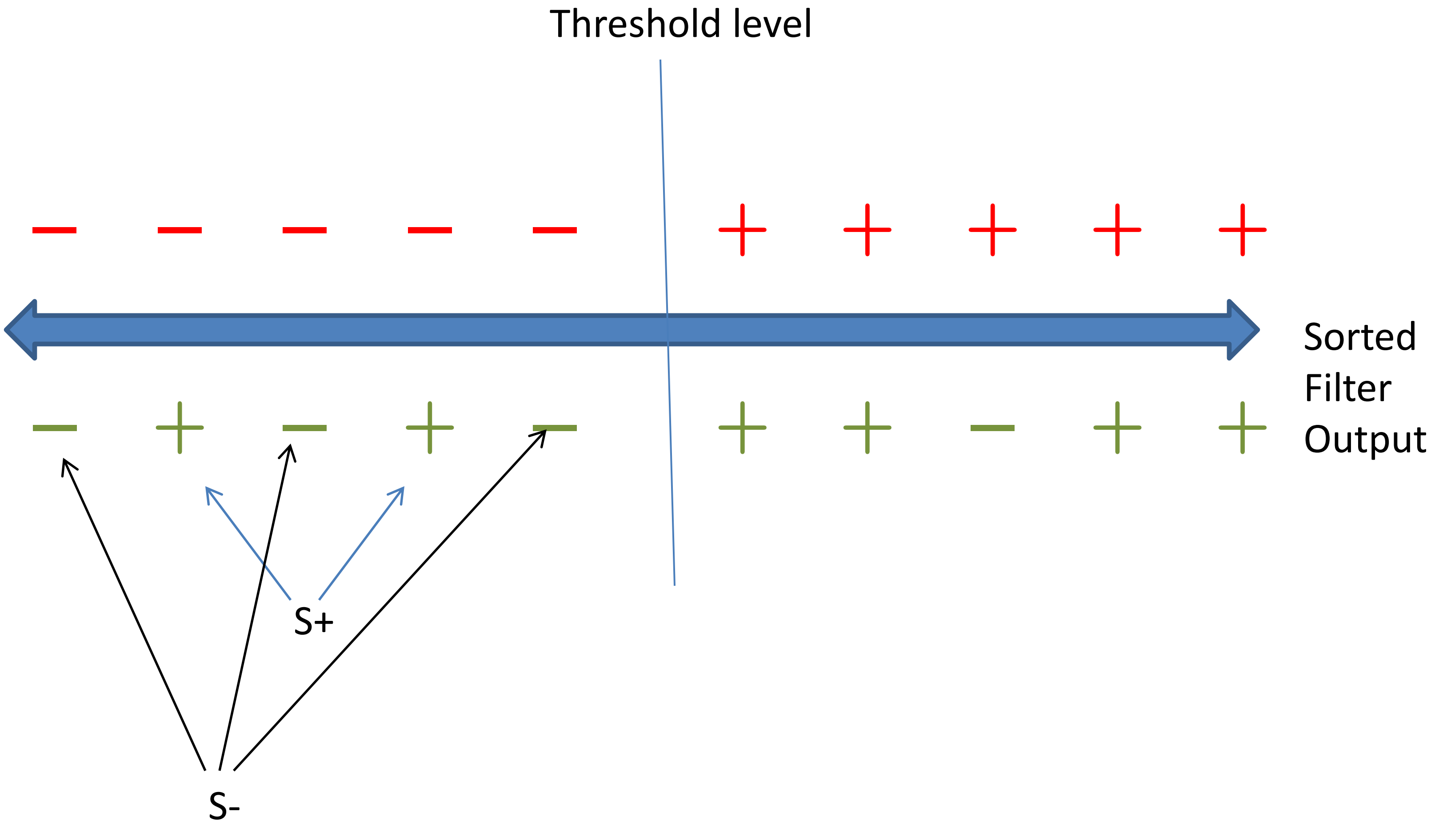
The weak classifier selection algorithm proceeds as follows. For each feature, the examples are sorted based on feature value. The AdaBoost optimal threshold for that feature can then be computed in a single pass over this sorted list. For each element in the sorted list, four sums are maintained and evaluated: the total sum of positive example weights T^+, the total sum of negative example weights T^-, the sum of positive weights below the current example S^+ and the sum of negative weights below the current example S^-. The error for a threshold which splits the range between the current and previous example in the sorted list is:

$$e = \min\left(S^+ + (T^- - S^-), S^- + (T^+ - S^+)\right),$$

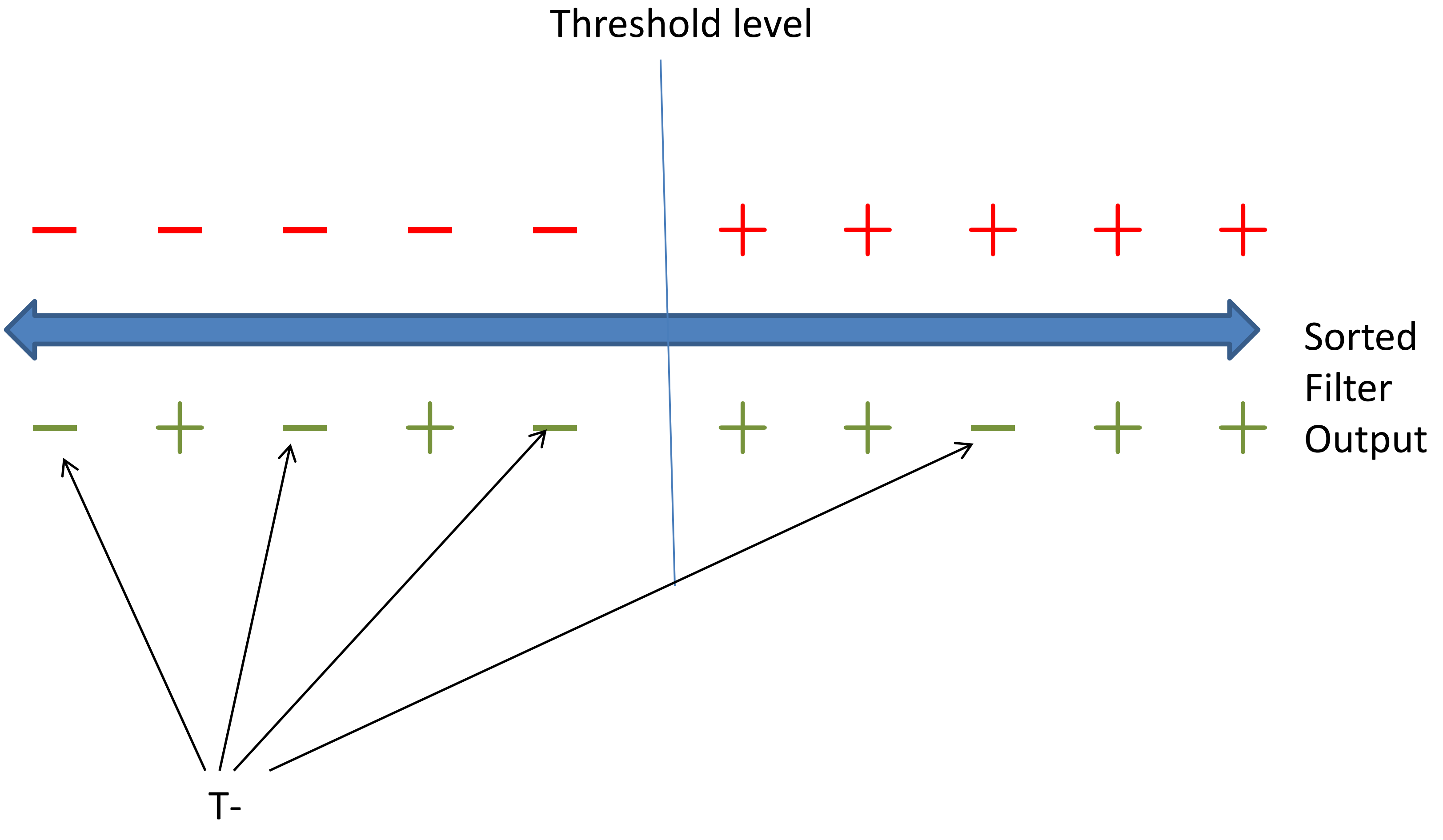
Step 2



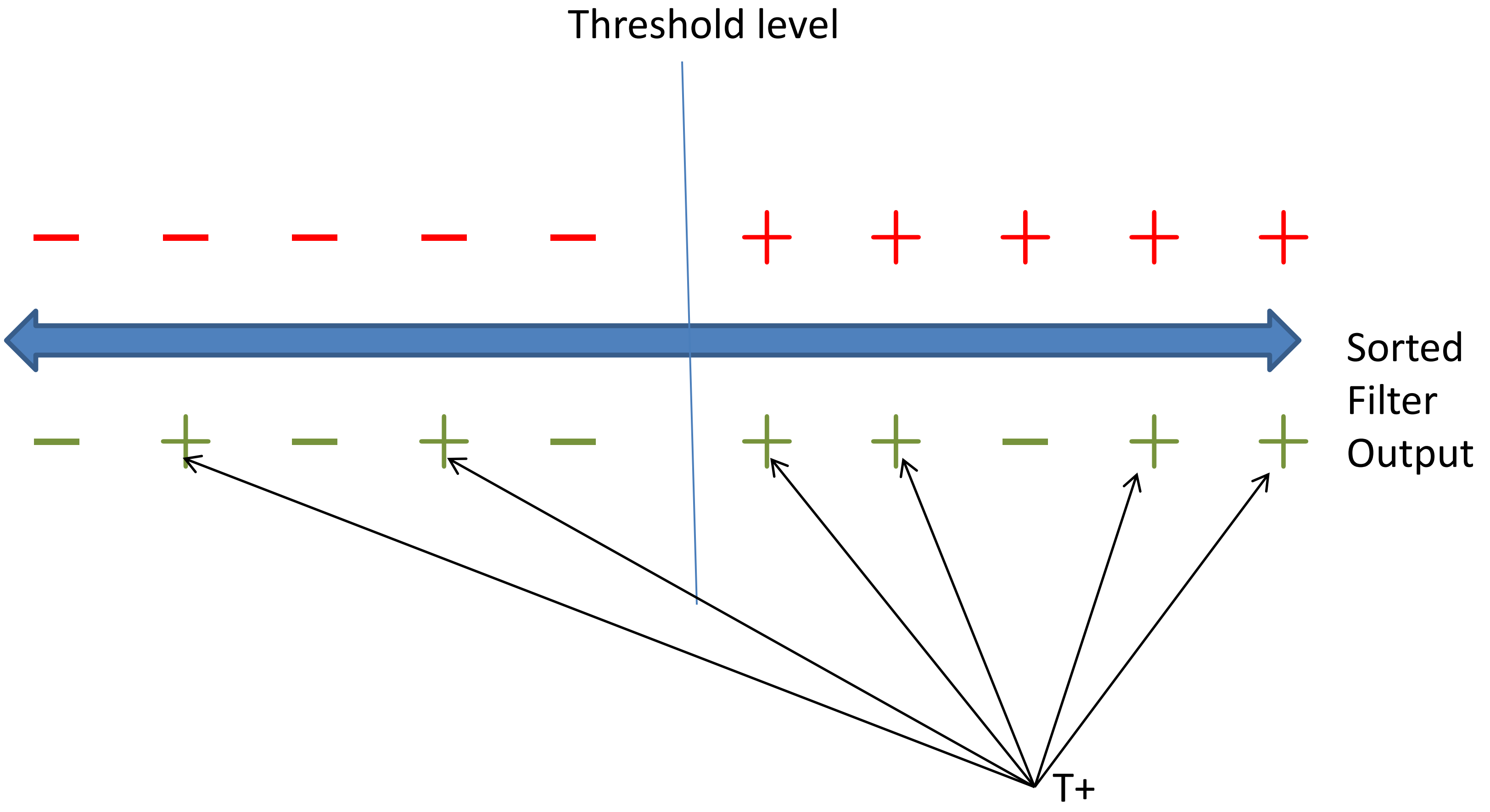
Step 2



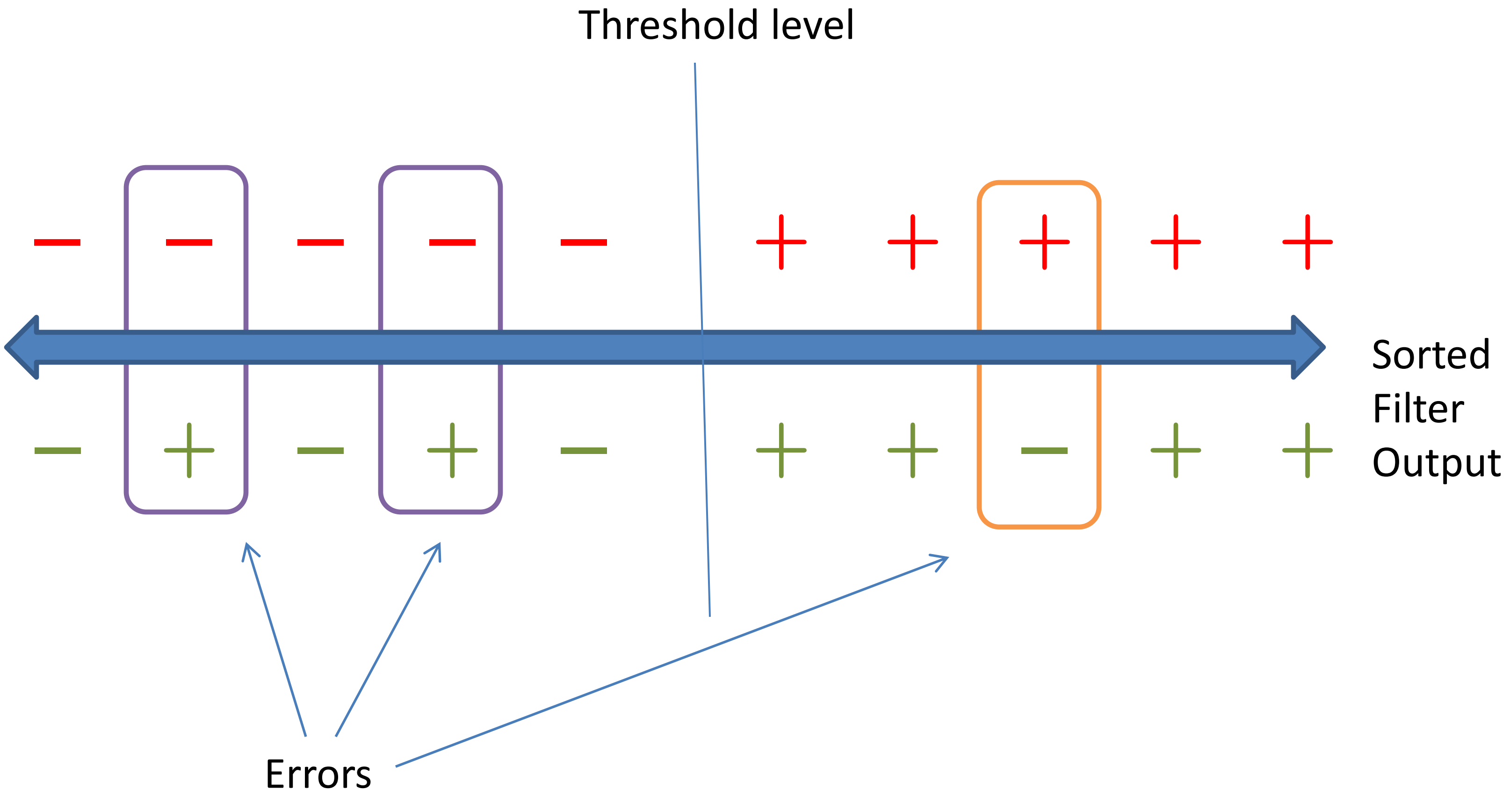
Step 2



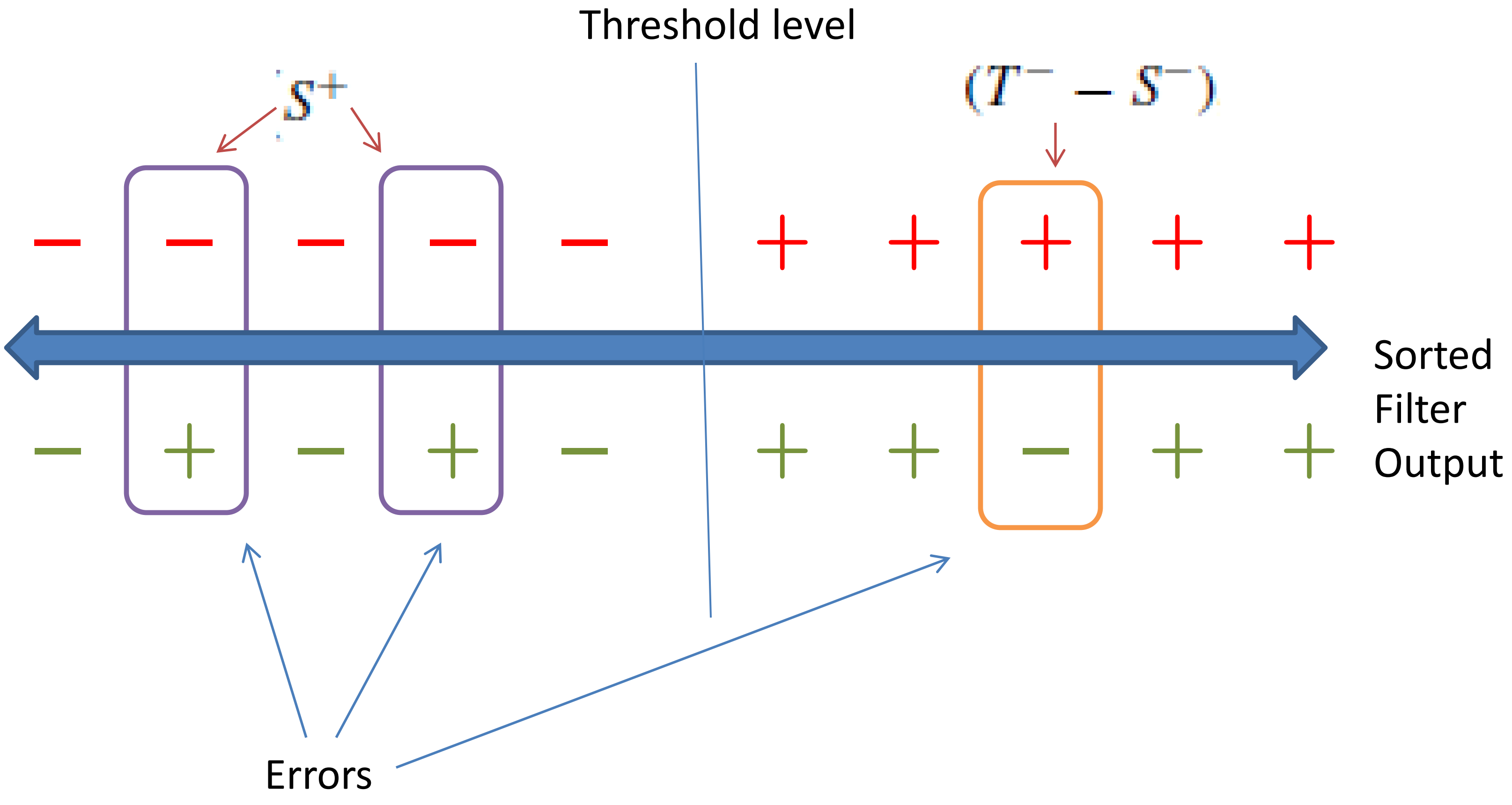
Step 2



Step 2

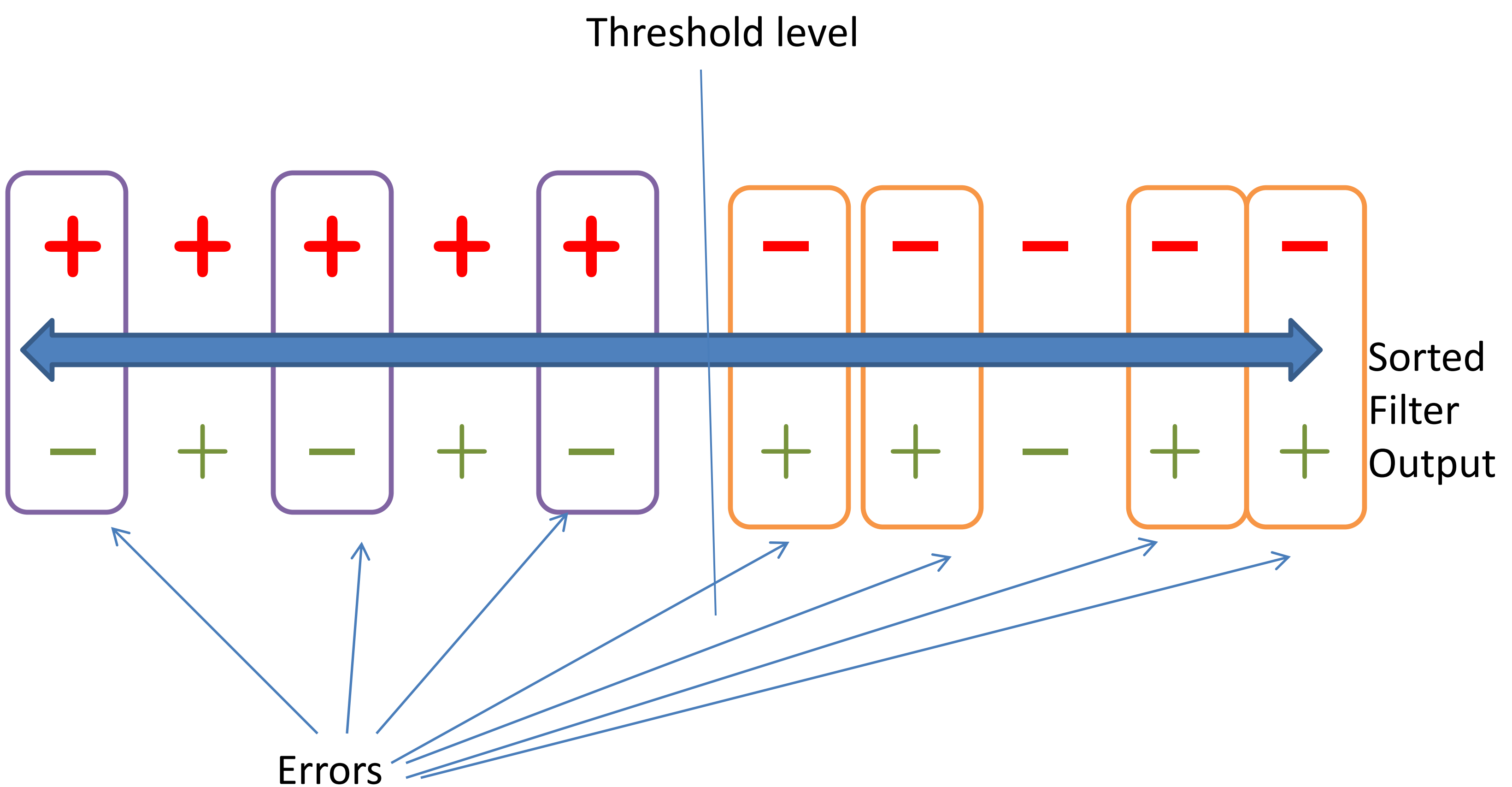


Step 2



$$e = \min\left(S^+ + (T^- - S^-), S^- + (T^+ - S^+)\right),$$

Step 2



$$e = \min\left(S^{+} + (T^{-} - S^{-}), S^{-} + (T^{+} - S^{+})\right),$$

Train

```
function TrainData = Boosting(PosData, NegData, T)

W=19; H=19;
np = size(PosData.feat, 1);
nn = size(NegData.feat, 1);

ys = [ones(np, 1); zeros(nn, 1)];
ws = [ones(np, 1)/(2*np); ones(nn, 1)/(2*nn)];

alphas = zeros(T, 1);
Thetas = zeros(T, 3);
feat = [PosData.feat; NegData.feat];
for t=1:T
    ws = ws / sum(ws);
    [theta, p, err] = LearnThreshClassifier(ws, feat, ys);
    [val, j] = min(err);
    e = err(j);
    beta = e/(1-e);
    hs = p(j).*feat(:,j) < p(j).*theta(j);
    wsu = (beta.^(1-abs(hs-ys)));
    Thetas(t,:) = [j, theta(j), p(j)];
    alphas(t) = log(1/beta);
    ws = ws .* wsu;
    disp(sprintf('Boosting iteration %d complete', t));
end

TrainData.alphas = alphas;
TrainData.Thetas = Thetas;
TrainData.featMat = PrepareFeatMat(PosData.featTypes(Thetas(:,1),:),W,H);
TrainData.featTypes = PosData.featTypes;
```

Train. Learn Classifier

```
function [theta, p, err] = LearnThreshClassifier(ws, feat, ys)
nof = size(feat, 2);
theta = [];
p = [];
err = [];

ysl = logical(ys);
wspos = ws;
wspos(~ysl) = 0;
wsneg = ws;
wsneg(ysl) = 0;

Tp = sum(wspos);
Tn = sum(wsneg);

for i=1:nof
    fsi = feat(:,i);
    [fsis,idx] = sort(fsi);
    Spc = cumsum(wspos(idx));
    Snc = cumsum(wsneg(idx));

    [emin,eminInd] =  min(min(Spc + (Tn - Snc),Snc + (Tp - Spc)));
    theta(i) = fsis(eminInd);

    p(i)=1;
    err(i)= sum(ws.*abs((p(i).*fsi<p(i).*theta(i)) - ys));

    etmp = sum(ws.*abs((-1.*fsi<-1.*theta(i)) - ys));
    if etmp < err(i)
        p(i)=-1;
        err(i) = etmp;
    end;
end;
```

Test.

```
function newDet = DetectFace(TrainData, im, min_scale,max_scale)
W=19; H=19;
threshold = TrainData.thresh;
featMat = TrainData.featMat';
alphas = TrainData.alphas';
theta = TrainData.Thetas(:,2);
p = TrainData.Thetas(:,3);
pTheta = p.*theta;
SSTEP = .1;
SAMPLE_RATE = 1;
EDGE = 25;
sz = [W H];
im = double(im);
detections= [];
for s= min_scale:SSTEP:max_scale
    ims = imresize(im,s);
    for indY=EDGE :SAMPLE_RATE:size(ims,1)- EDGE
        for indX=EDGE:SAMPLE_RATE:size(ims,2) - EDGE
            limy = indY-floor(H/2):indY+floor(H/2);
            limx = indX-floor(W/2):indX+floor(W/2);
            imw = ims(limy,limx);
            m = mean(imw(:));
            stnd = std(imw(:));
            if stnd > 20
                imw = (imw-m)/stnd;
                fs = featMat * iImw(:);
                Score = alphas*(p.*fs<pTheta);
                if Score>threshold
                    scaleddetloc = [limy(1) limx(1) sz(1) sz(2) Score]./s;
                    detections = [detections;scaleddetloc];
                end
            end;
        end;
    end;
end;
```