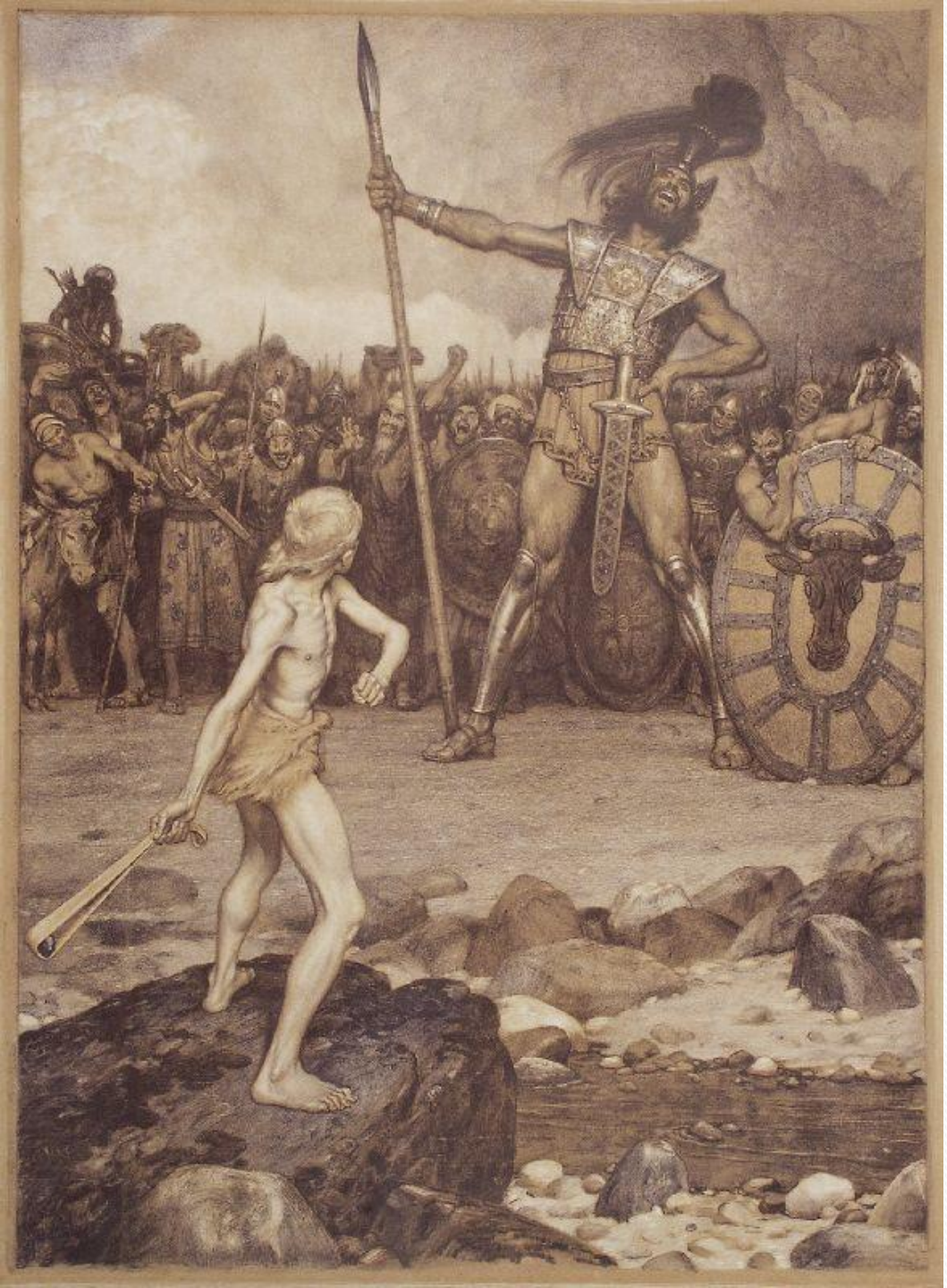


Boosting and face detection

September 20, 2012

Imagine this scenario

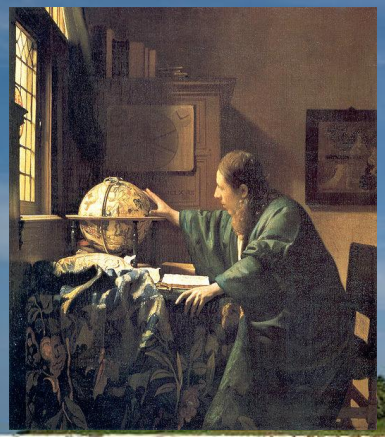
Imagine this scenario

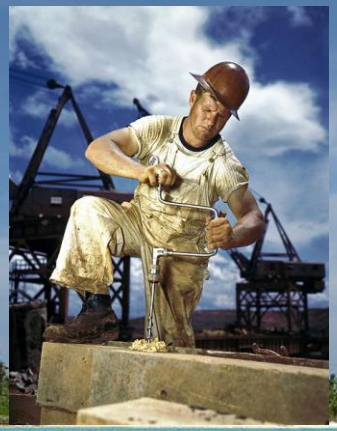
Instead,

Instead,

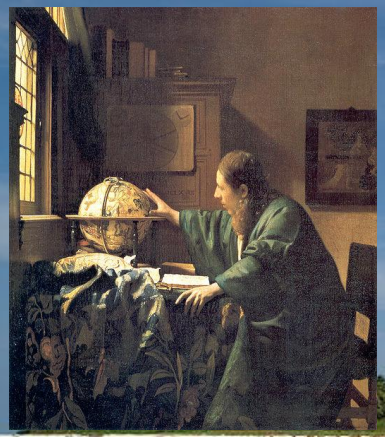
Instead,

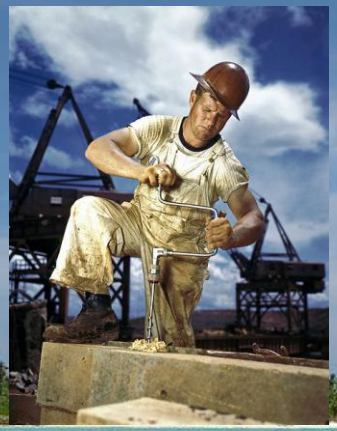
Instead,

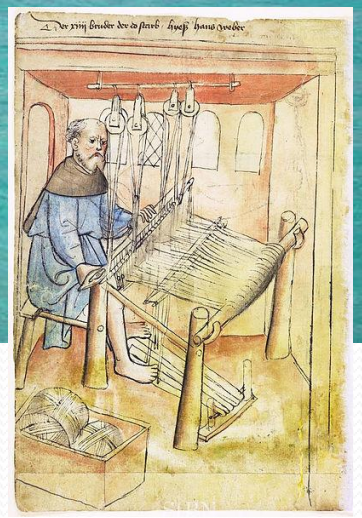




Instead,



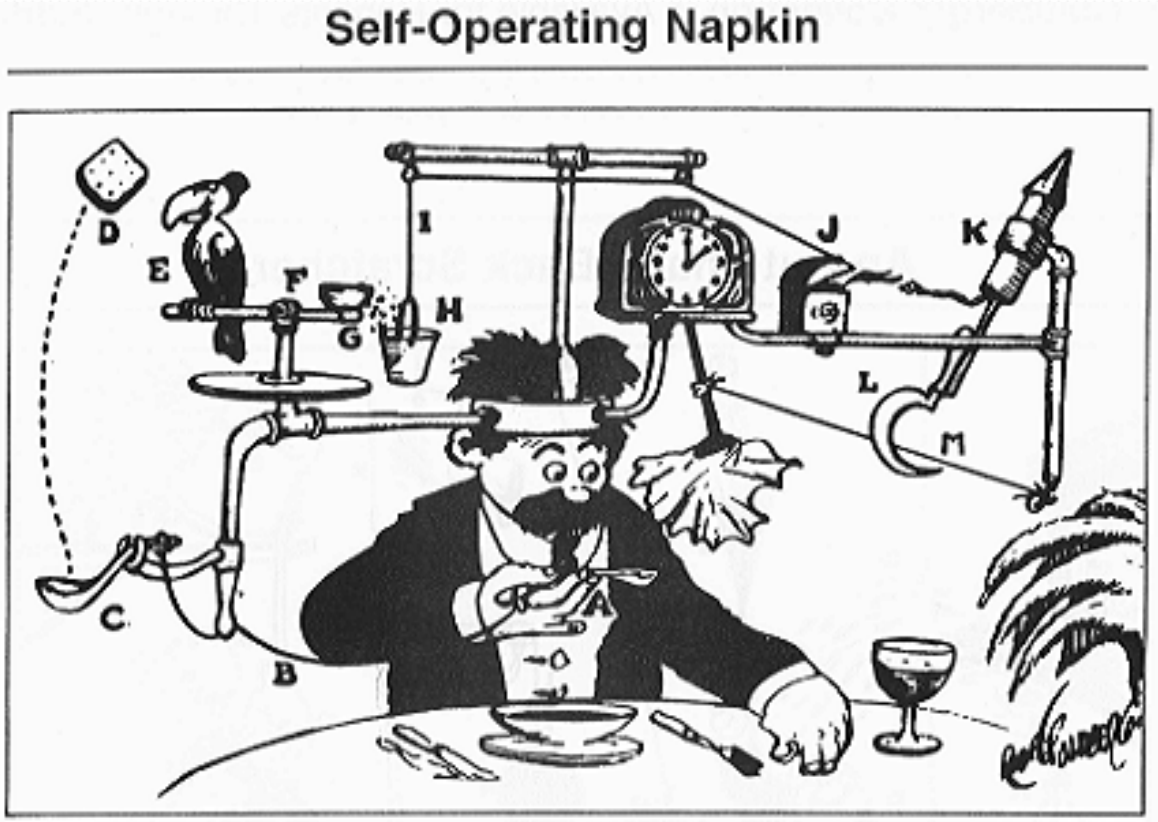




Individually, none of them can get off the island but…

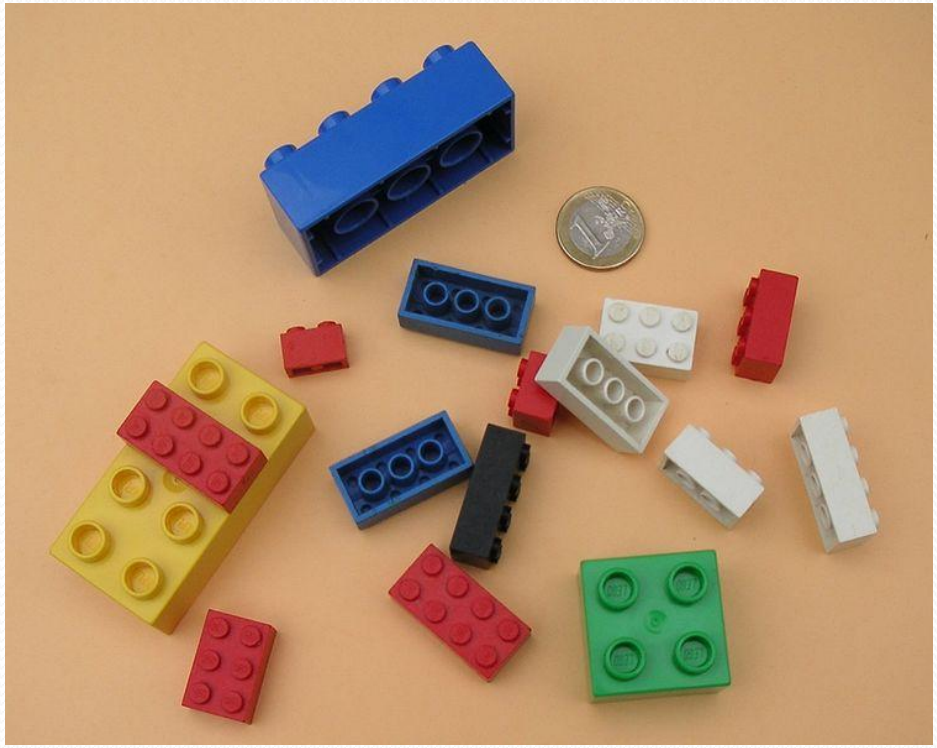
Strong and Weak learners

- Strong Learner:



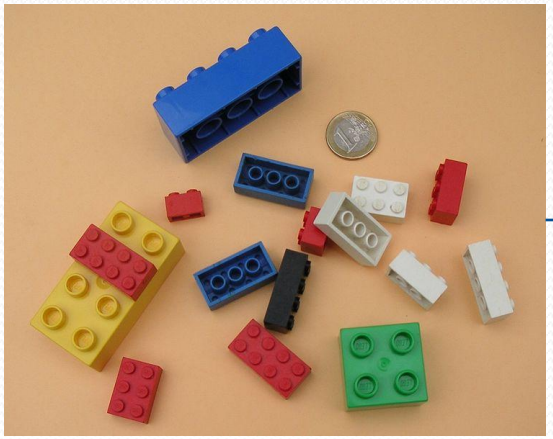
Strong and Weak learners

- Weak Learner:



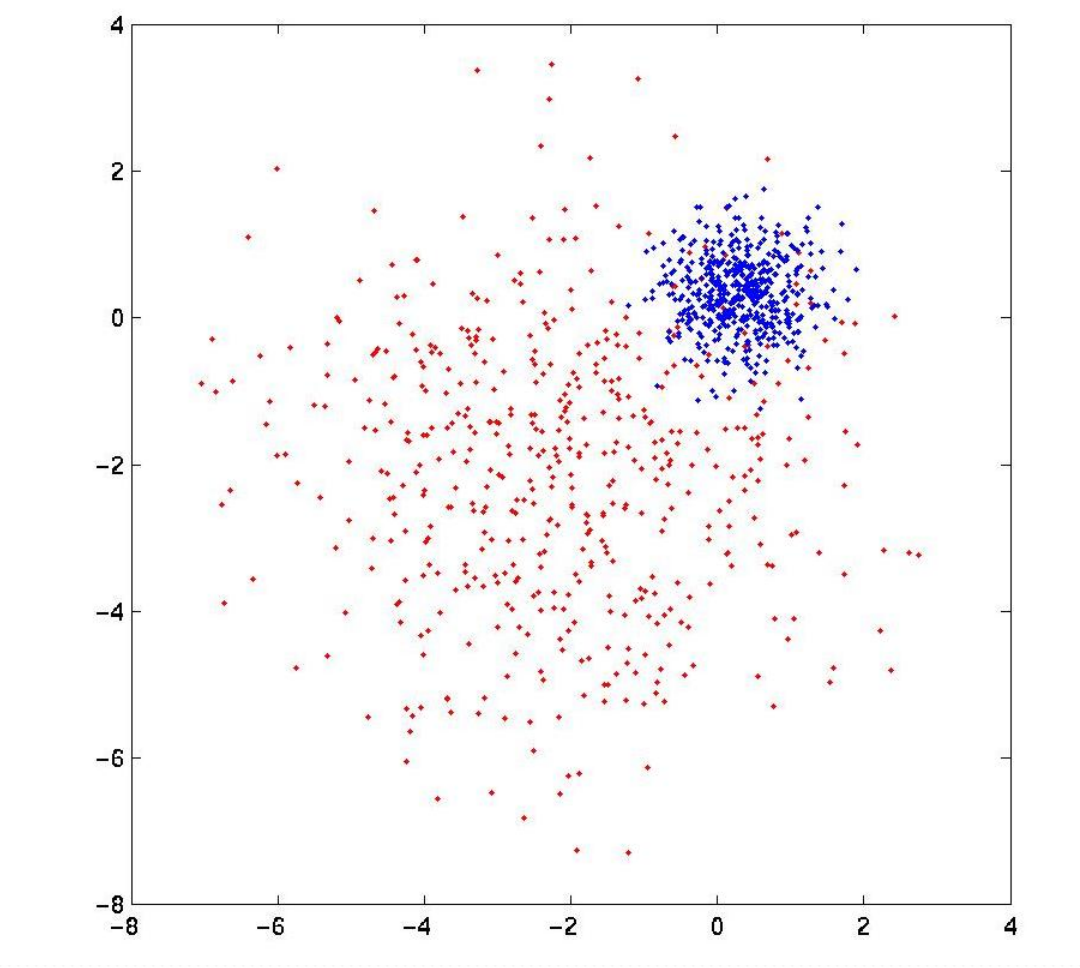
Strong and Weak learners

- Boosting:

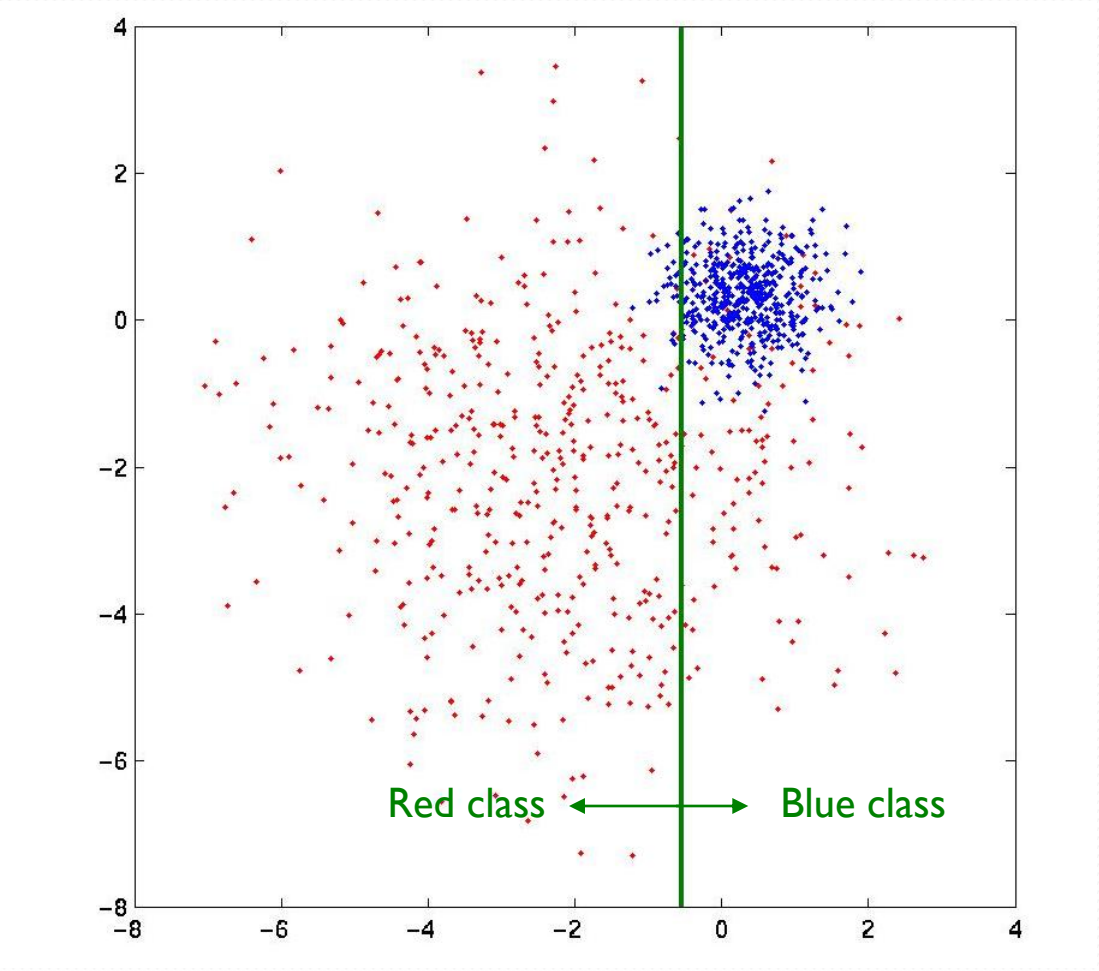


Combining
weak learners

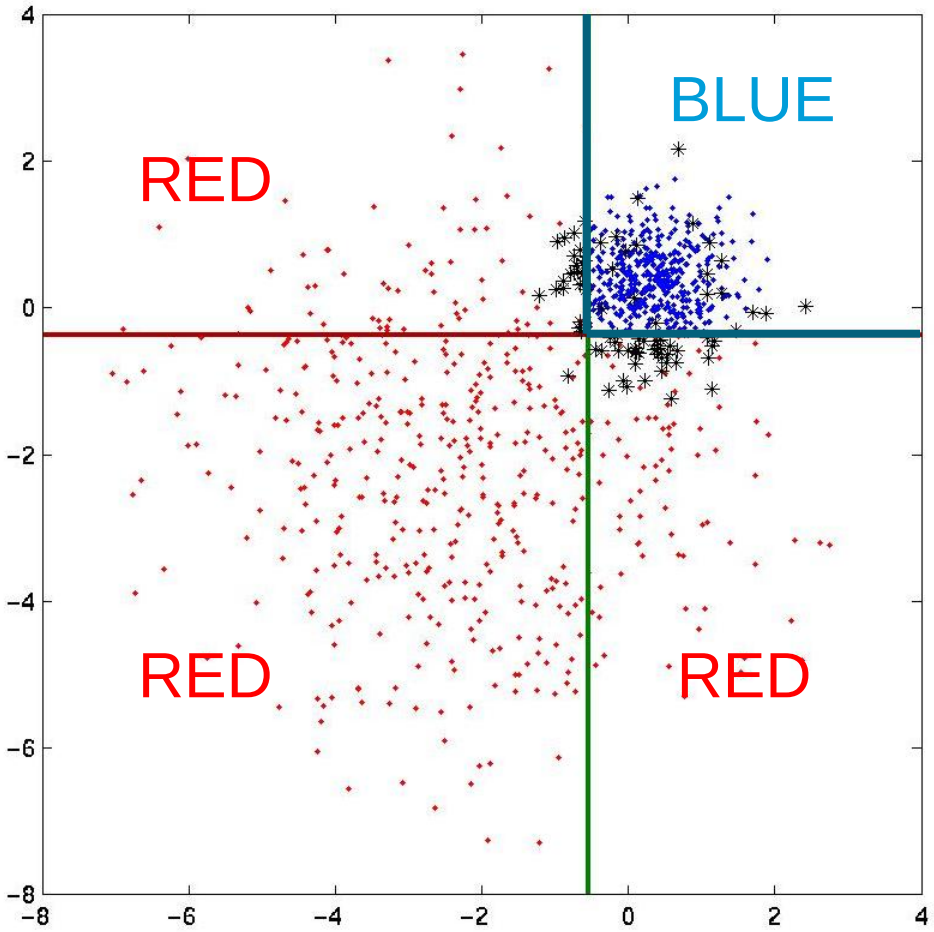
Boosting: An example



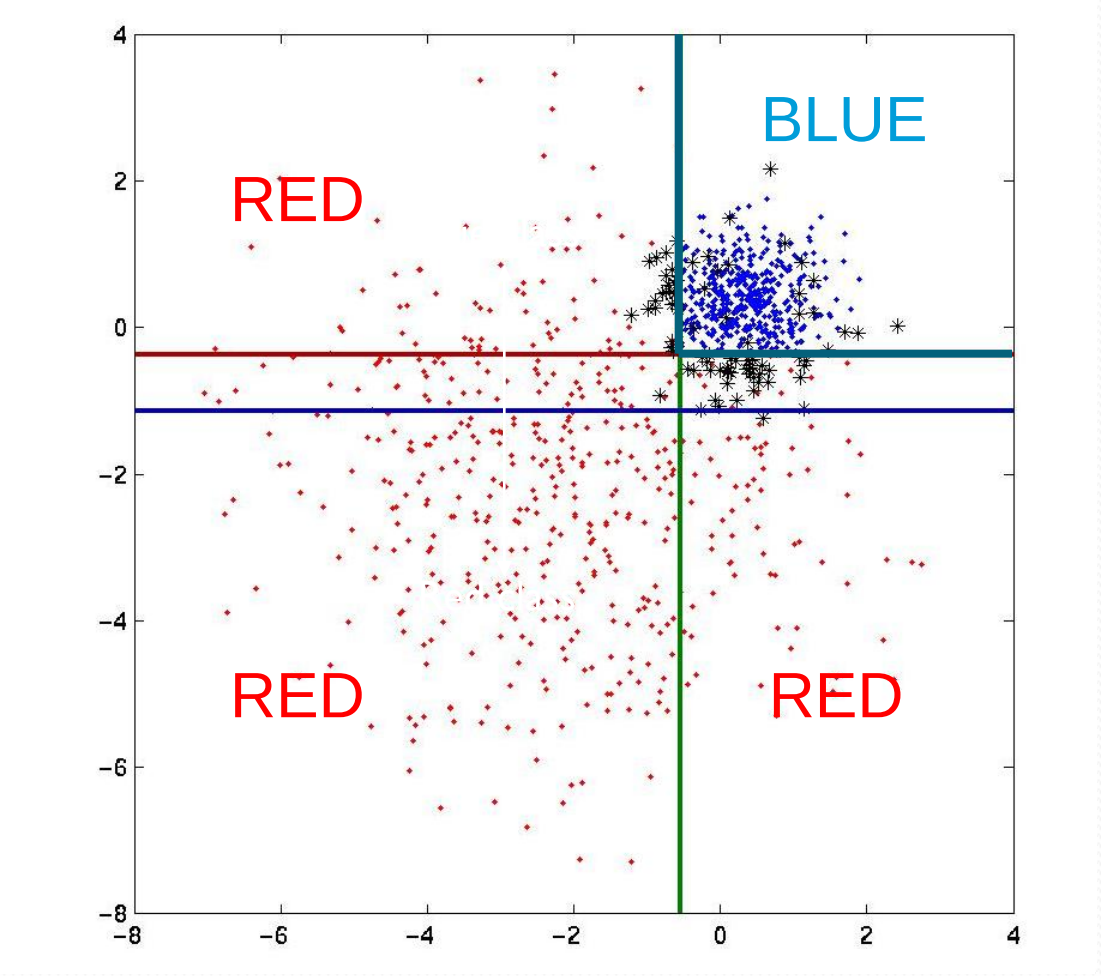
Boosting: An example



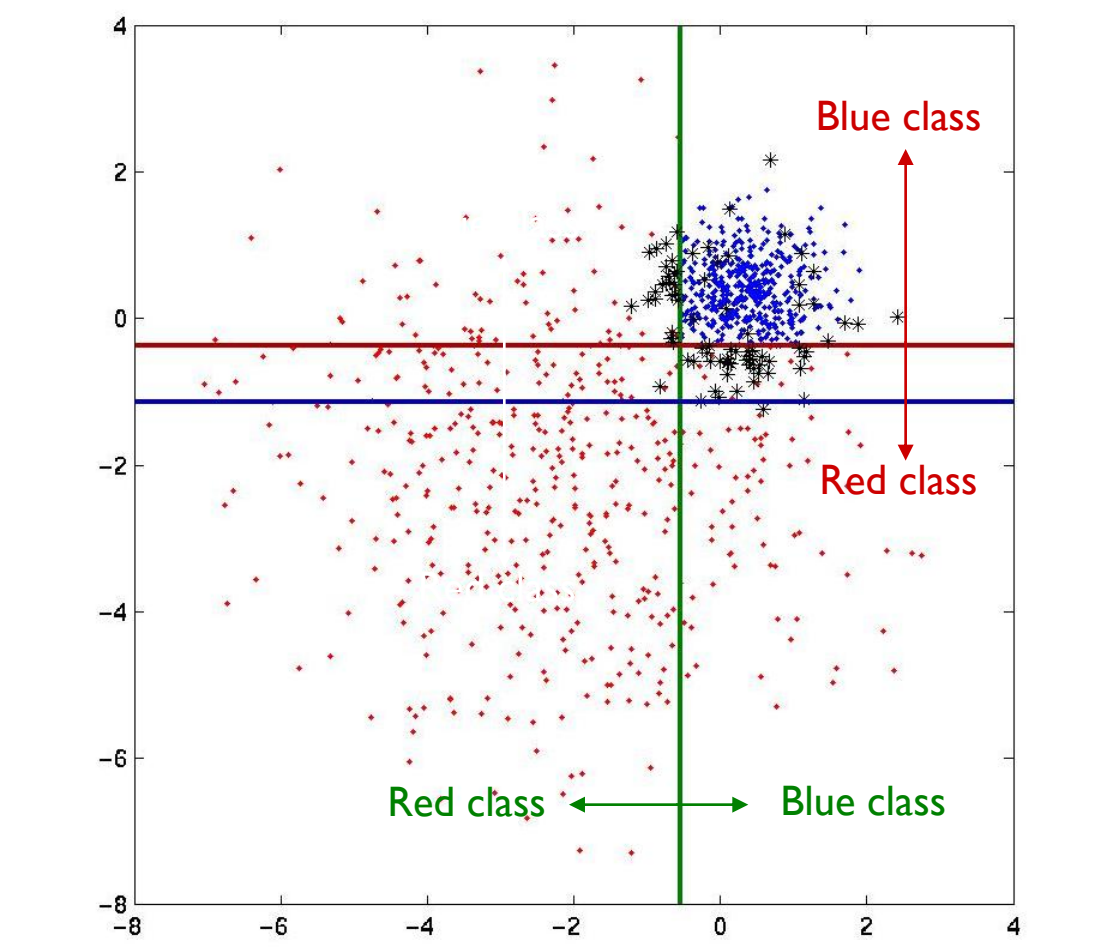
Boosting: An example



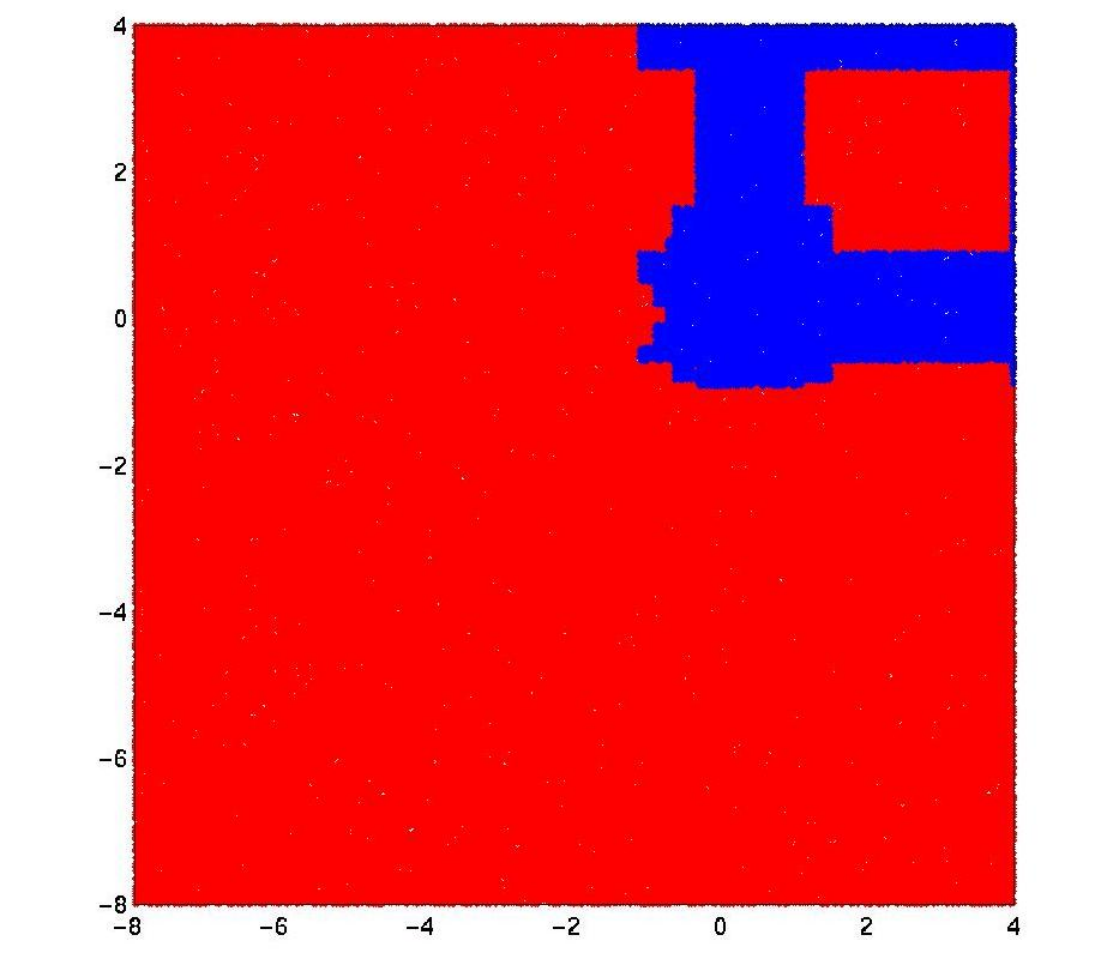
Boosting: An example



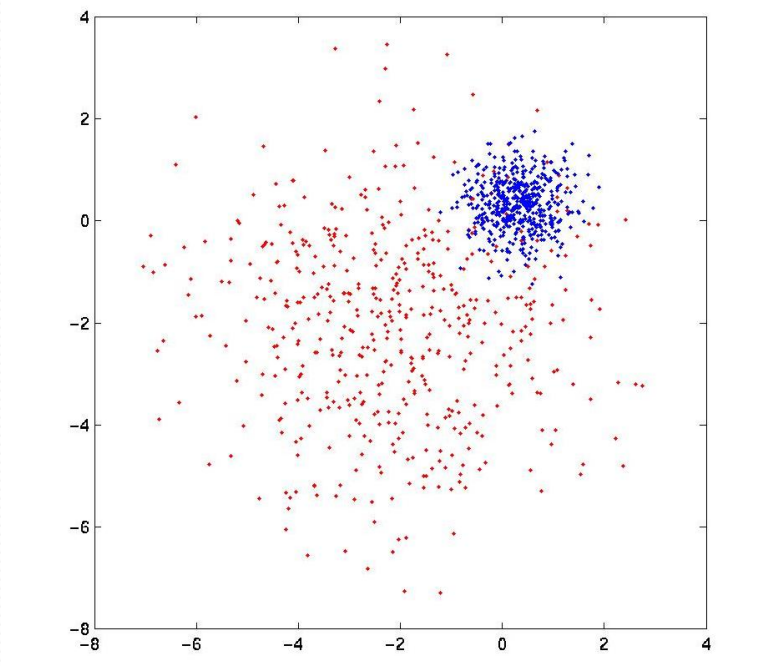
Boosting: An example

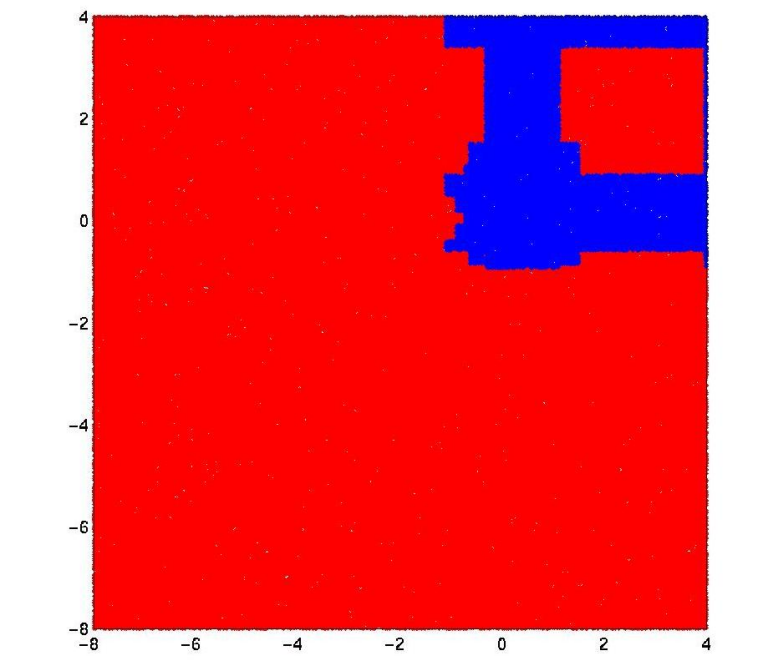


Boosting: An example

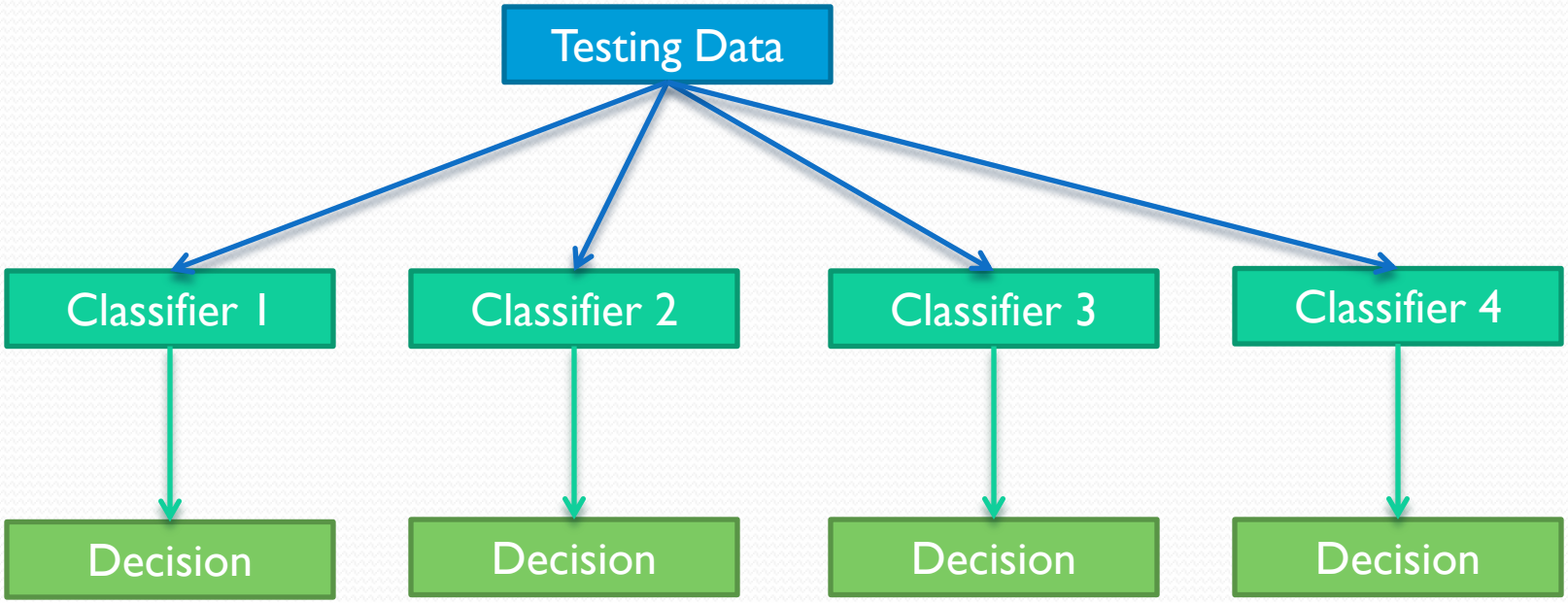


Boosting: An example



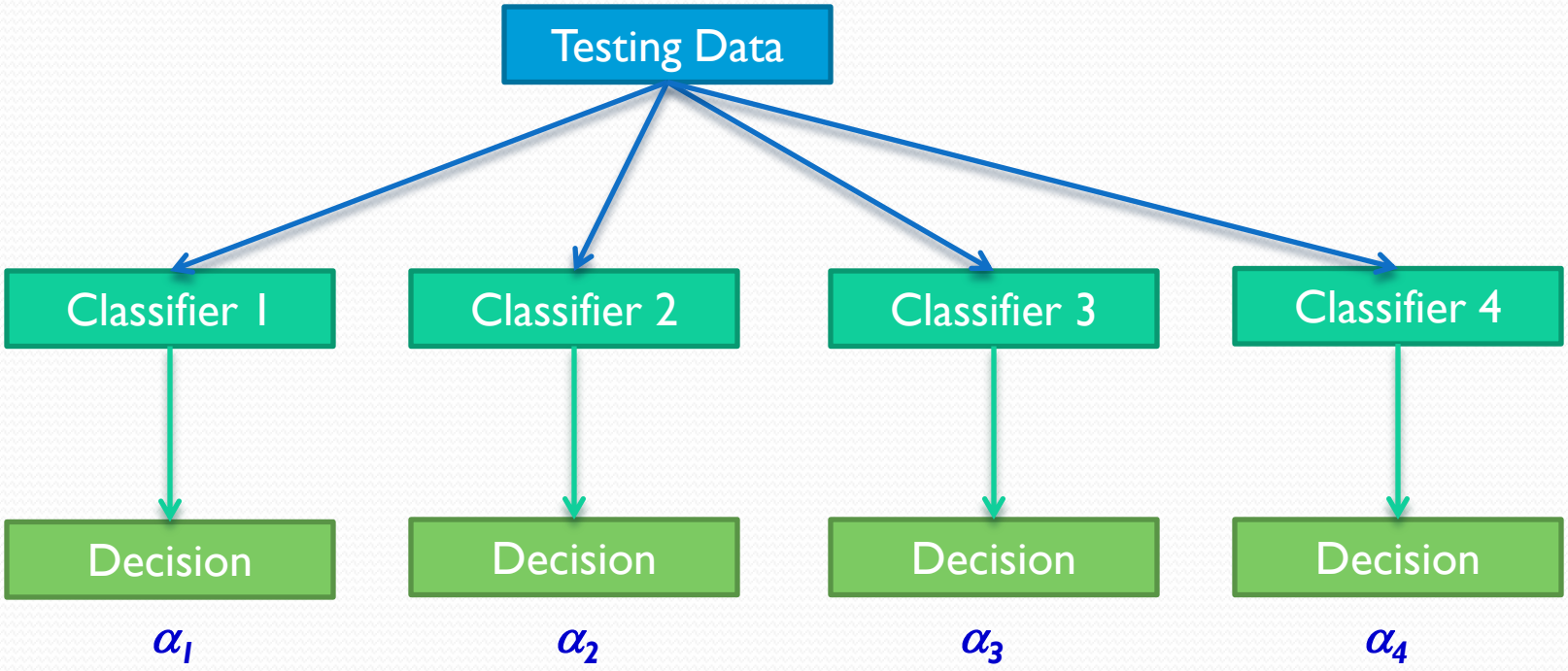


Using voting



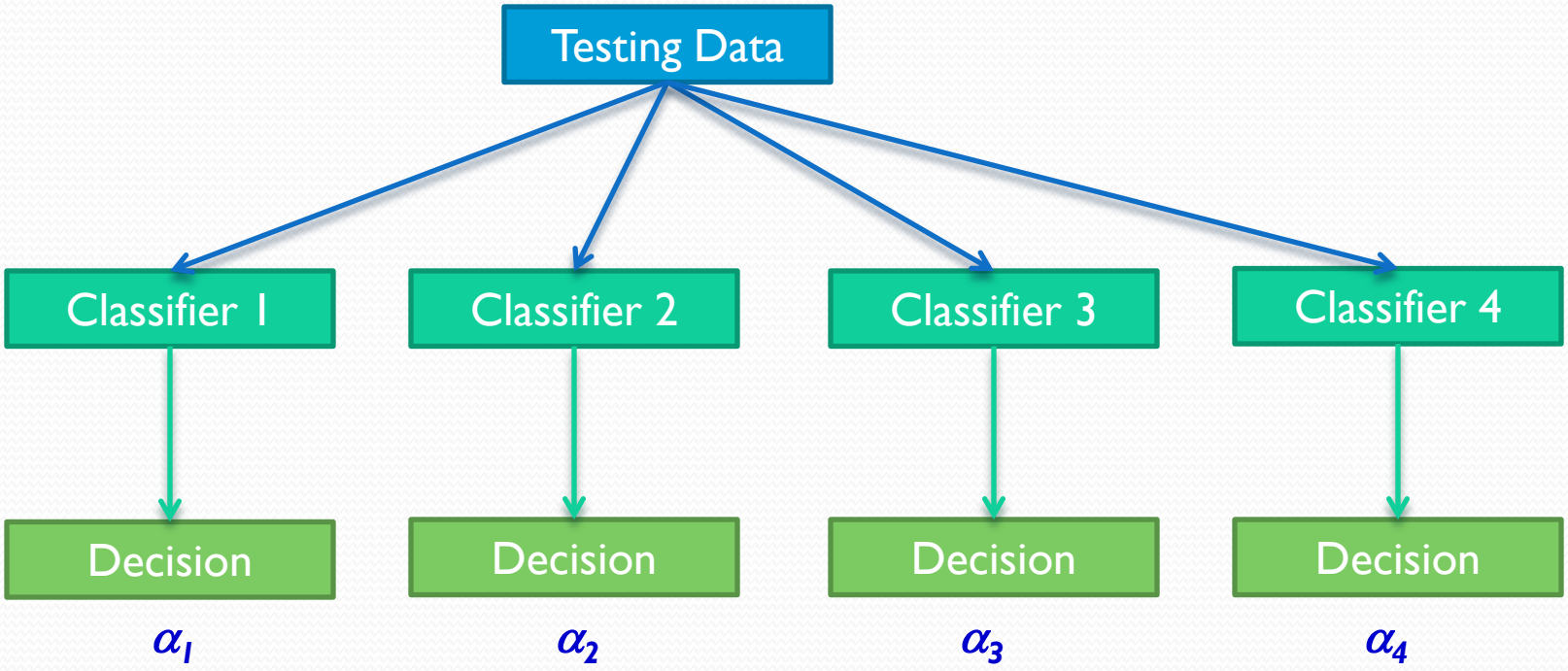
Final Decision = majority decision of classifiers

Using confidence measures



Final Decision = **weighted** majority decision of classifiers

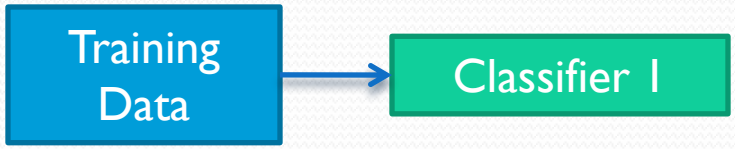
Using confidence measures



Final Decision = **weighted** majority decision of classifiers

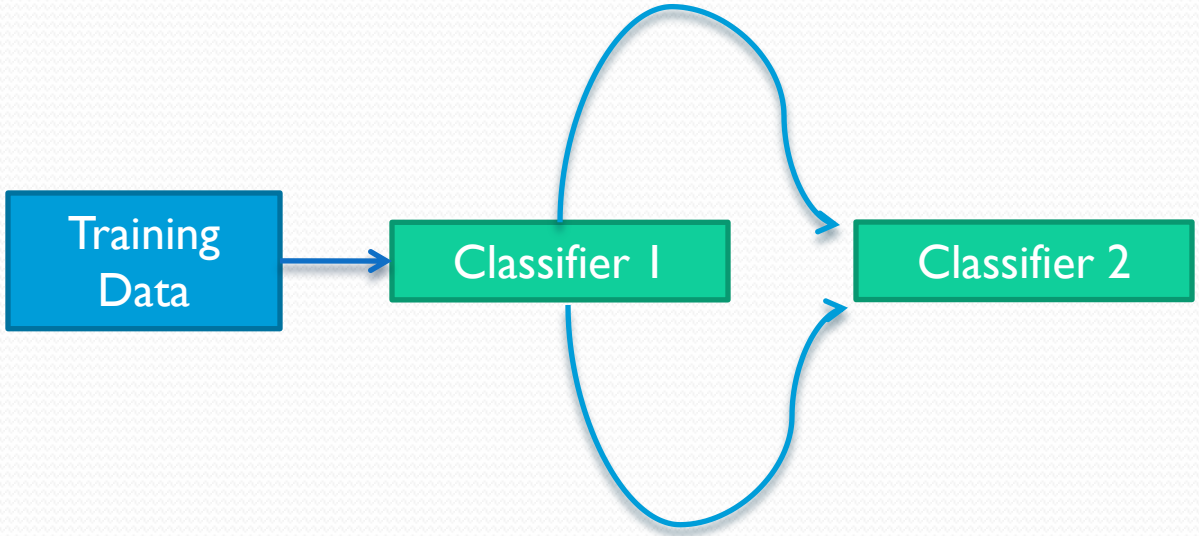
But we really aren't making the best use of the classifiers !!

Better way, AdaBoost



Better way, AdaBoost

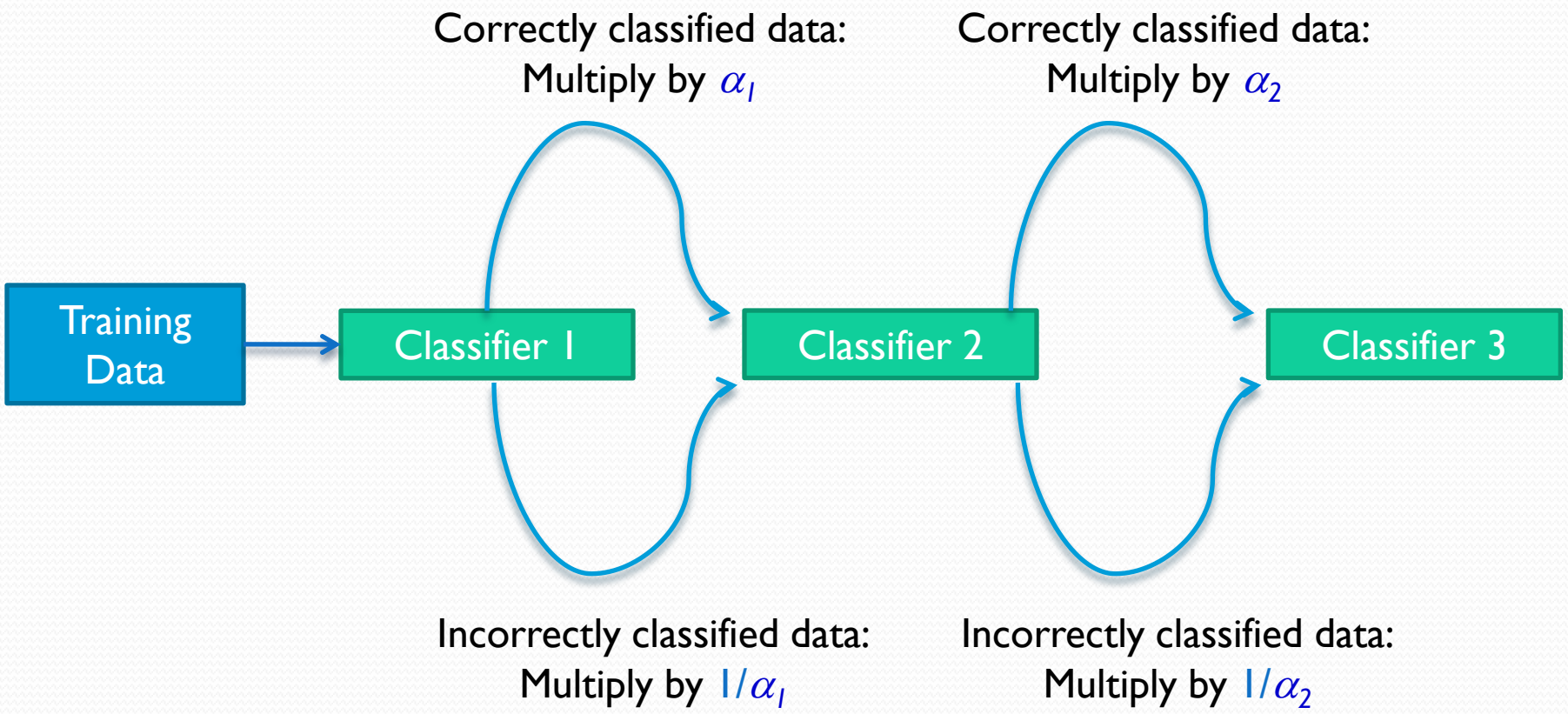
Correctly classified data:
Multiply by α_1



Incorrectly classified data:
Multiply by $1/\alpha_1$

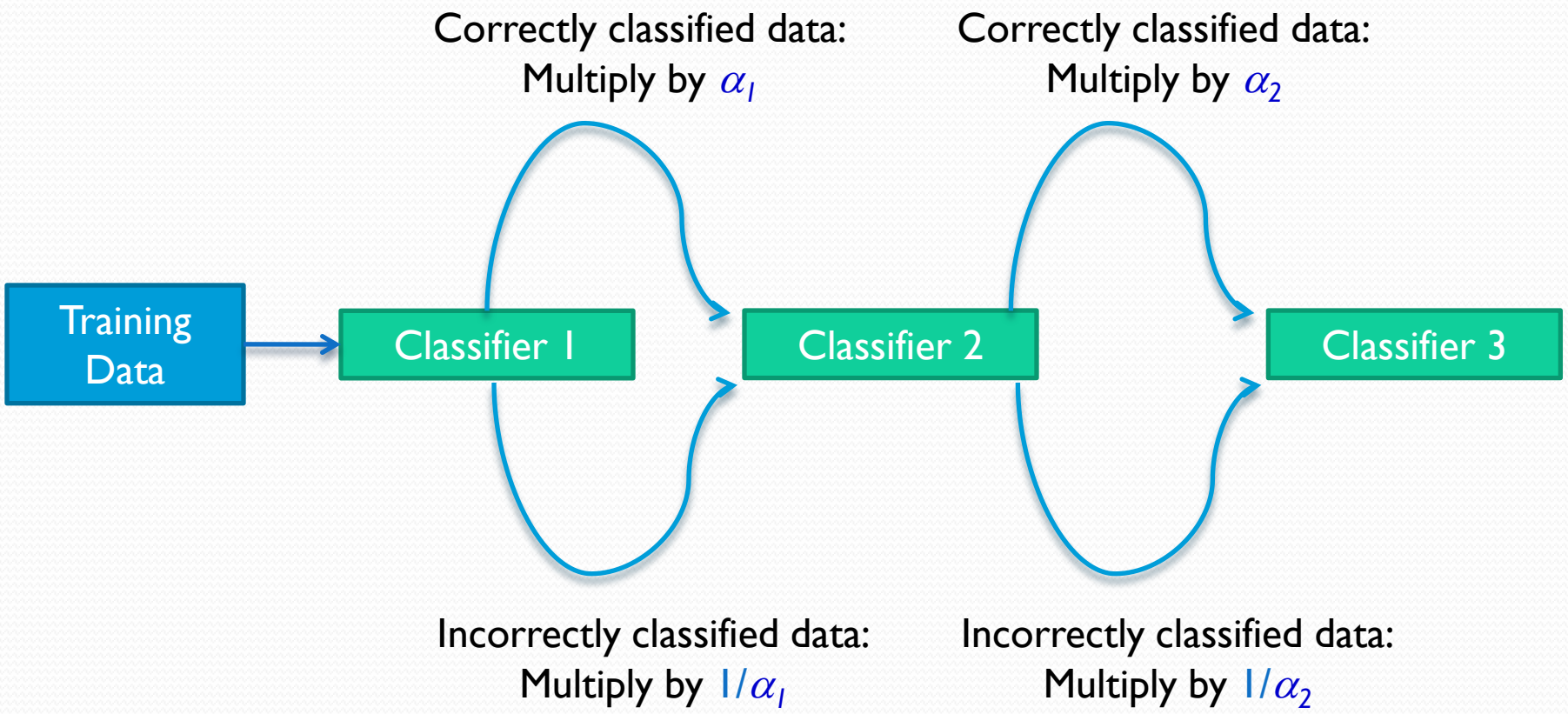
where $\alpha_i < 1$

Better way, AdaBoost



where $\alpha_i < 1$

Better way, AdaBoost



Final classifier: $f(x) = \Sigma_i \alpha_i h_i(x)$

,where $h_i(x)$ is the i'th classifier

Formalizing the concept

- Training data:

(x_i, y_i)
where x_i is the input to the classifier
and y_i is the correct output, either +1 or -1

Formalizing the concept

- Training data:

  (x_i, y_i)
  where x_i is the input to the classifier
  and y_i is the correct output, either +1 or -1

- Set of classifiers: $h_1, h_2, h_3, \ldots h_T$

  where $h_i(x_i)$ classifies the input x_i as +1 or -1
  $y*h(x)$ is +1 if correctly classified and -1 if incorrectly classified

Formalizing the concept

- Training data:

  (x_i, y_i)
  where x_i is the input to the classifier
  and y_i is the correct output, either +1 or -1

- Set of classifiers: $h_1, h_2, h_3, \ldots h_T$

  where $h_i(x_i)$ classifies the input x_i as +1 or -1
  $y*h(x)$ is +1 if correctly classified and -1 if incorrectly classified

- Devise a function $f(h_1(x), h_2(x), \ldots, h_T(x))$ such that classification based on $f()$ is superior to classification by any $h_i(x)$

Boosting

- Voting
  - $f(x) = \Sigma_i h_i(x)$
  - Classifier $H(x) = \text{sign}(f(x)) = \text{sign}(\Sigma_i h_i(x))$

Boosting

- Voting
  - $f(x) = \Sigma_i h_i(x)$
  - Classifier $H(x) = \text{sign}(f(x)) = \text{sign}(\Sigma_i h_i(x))$

- Using weights:
  - $f(x) = \Sigma_i \alpha_i h_i(x)$
  - Classifier $H(x) = \text{sign}(f(x)) = \text{sign}(\Sigma_i \alpha_i h_i(x))$
    - The weight α_i for any $h_i(x)$ is a measure of our trust in $h_i(x)$

AdaBoost

- As before:
  - y_i is either +1 or -1
  - $H(x_i)$ is also either +1 or -1

AdaBoost

- As before:
  - y_i is either +1 or -1
  - $H(x_i)$ is also either +1 or -1
- Correctly classified: $y_i * H(x_i) = +1$
- Incorrectly classified: $y_i * H(x_i) = -1$

AdaBoost

- As before:
  - y_i is either +1 or -1
  - $H(x_i)$ is also either +1 or -1
- Correctly classified: $y_i * H(x_i) = +1$
- Incorrectly classified: $y_i * H(x_i) = -1$
- Error function: $\frac{1}{2} * (1 - y_i * H(x_i))$
  - Correct classification: 0 Incorrect classification: 1

The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - ε_t = Sum $\{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\ \alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

AdaBoost Example: Face detection

- Training data:

 = 0.3 E1 - 0.6 E2

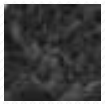
 = 0.5 E1 - 0.5 E2

 = 0.7 E1 - 0.1 E2

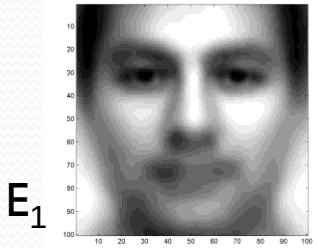
 = 0.6 E1 - 0.4 E2

 = 0.2 E1 + 0.4 E2

 = -0.8 E1 - 0.1 E2

 = 0.4 E1 - 0.9 E2

 = 0.2 E1 + 0.5 E2



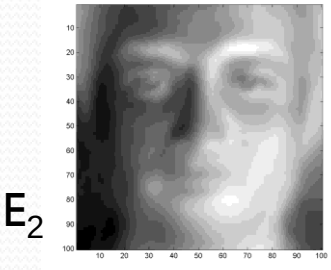


Image = a*E1 + b*E2 → a = Image.E1/|Image|

- Face detection with Eigen faces
- Step 0: Derive top 2 Eigen faces from training data
- Step 1: Represent all images in development set with these Eigen faces

Training data

 = 0.3 E1 - 0.6 E2

 = 0.5 E1 - 0.5 E2

 = 0.7 E1 - 0.1 E2

 = 0.6 E1 - 0.4 E2

 = 0.2 E1 + 0.4 E2

 = -0.8 E1 - 0.1 E2

 = 0.4 E1 - 0.9 E2

 = 0.2 E1 + 0.5 E2

ID	E1	E2.	Class
A	0.3	-0.6	+1
B	0.5	-0.5	+1
C	0.7	-0.1	+1
D	0.6	-0.4	+1
E	0.2	0.4	-1
F	-0.8	-0.1	-1
G	0.4	-0.9	-1
H	0.2	0.5	-1

Face = +1
Non-face = -1

The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - $\varepsilon_t = \text{Sum}\ \{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\ \alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

Initialization

 = 0.3 E1 - 0.6 E2

 = 0.5 E1 - 0.5 E2

 = 0.7 E1 - 0.1 E2

 = 0.6 E1 - 0.4 E2

 = 0.2 E1 + 0.4 E2

 = -0.8 E1 - 0.1 E2

 = 0.4 E1 - 0.9 E2

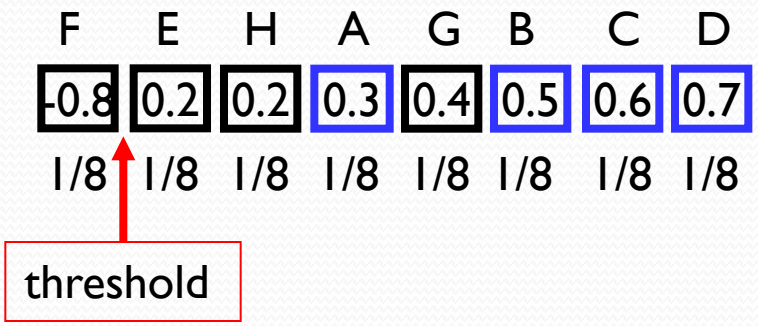
 = 0.2 E1 + 0.5 E2

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - ε_t = Sum $\{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\ \alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

The E1 "Stump"



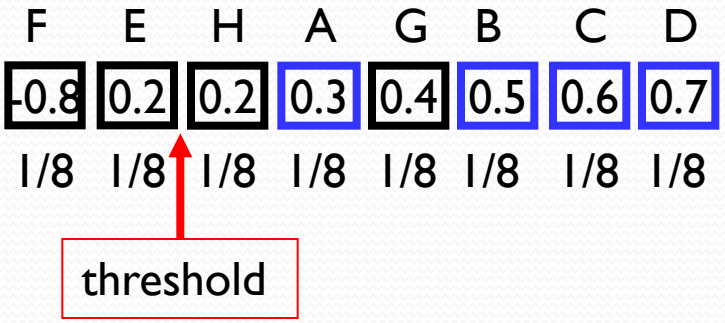
Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 3/8
Sign = -1, error = 5/8

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The E1 "Stump"



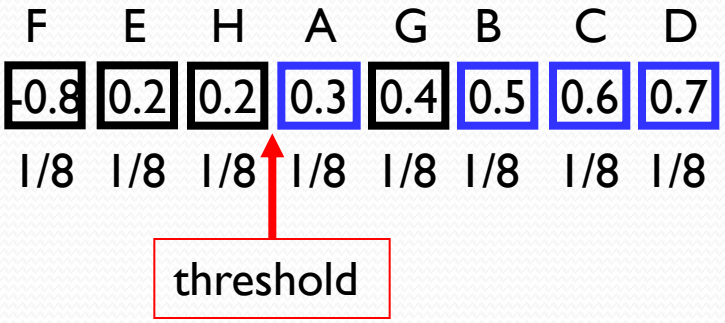
Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 2/8
Sign = -1, error = 6/8

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The E1 "Stump"



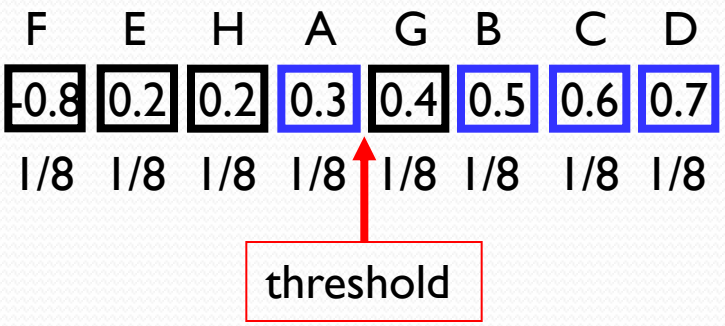
Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 1/8
Sign = -1, error = 7/8

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The E1 "Stump"



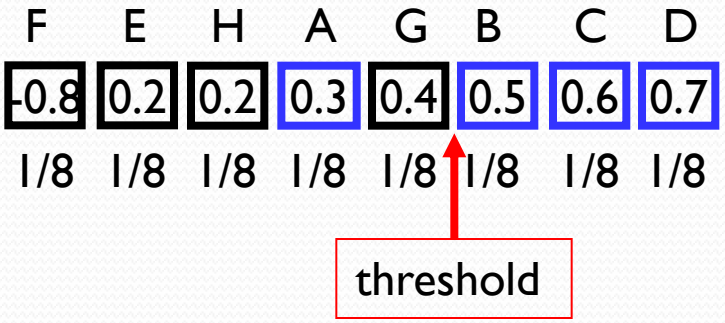
Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 2/8
Sign = -1, error = 6/8

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The E1 "Stump"



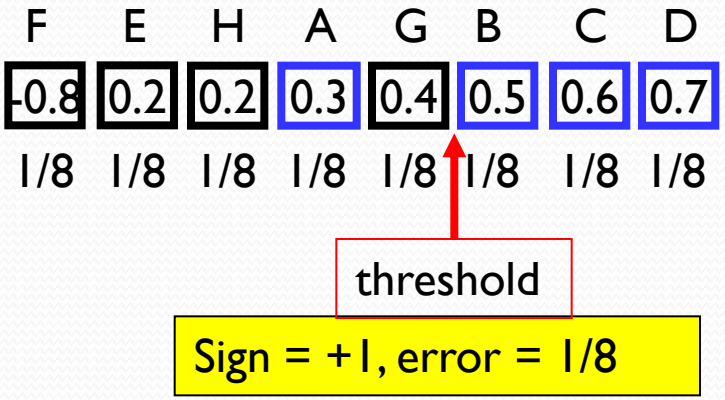
Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 1/8
Sign = -1, error = 7/8

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The Best E1 "Stump"

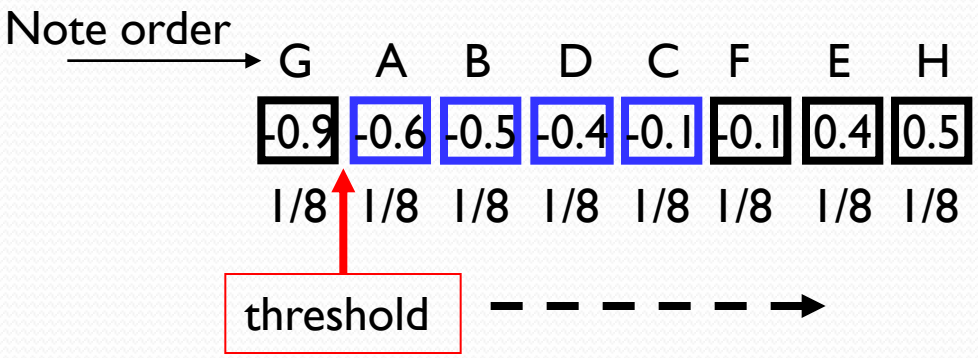


Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

Sign = +1
Threshold = 0.45

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The E2"Stump"



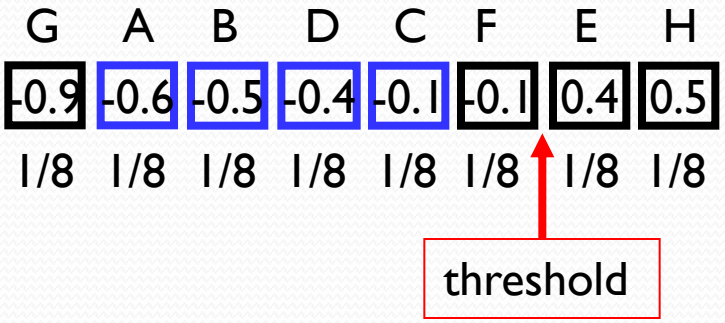
Classifier based on E2:
if (sign*wt(E2) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 3/8
Sign = -1, error = 5/8

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The Best E2"Stump"



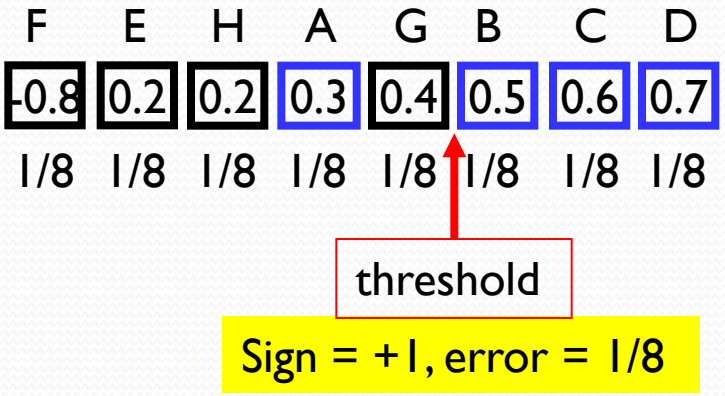
Sign = -1, error = 2/8

Classifier based on E2:
if (sign*wt(E2) > thresh) > 0)
face = true

sign = -1
Threshold = 0.15

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	-0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The Best "Stump"



Sign = +1, error = 1/8

The Best overall classifier based on a single feature is based on E1

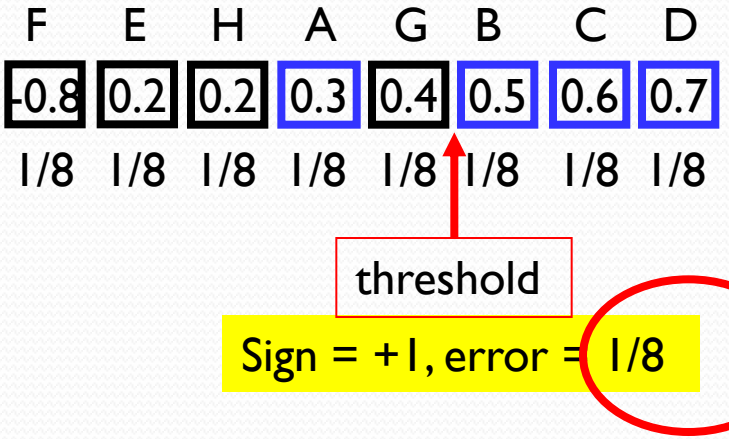
If (wt(E1) > 0.45) → Face

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - ε_t = Sum $\{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

The Best Error



The Error of the classifier is the sum of the weights of the misclassified instances

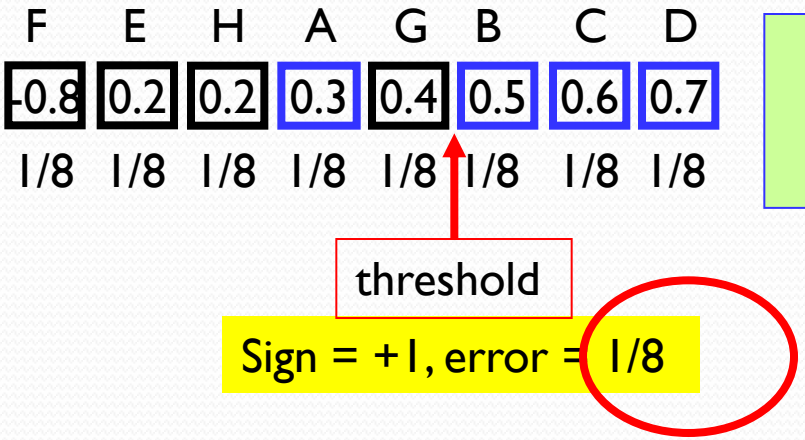
ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	1/8
B	0.5	-0.5	+1	1/8
C	0.7	-0.1	+1	1/8
D	0.6	-0.4	+1	1/8
E	0.2	0.4	-1	1/8
F	-0.8	0.1	-1	1/8
G	0.4	-0.9	-1	1/8
H	0.2	0.5	-1	1/8

NOTE: THE ERROR IS THE SUM OF THE WEIGHTS OF MISCLASSIFIED INSTANCES

The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - ε_t = Sum $\{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

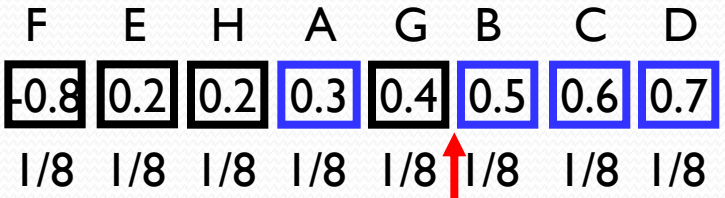
Computing Alpha



α = 0.5ln((1-1/8) / (1/8))

= 0.5 ln(7) = 0.97

The Boosted Classifier Thus Far



Sign = +1, error = 1/8

α = 0.5ln((1-1/8) / (1/8))

= 0.5 ln(7) = 0.97

h1(X) = wt(E1) > 0.45 ? +1 : -1

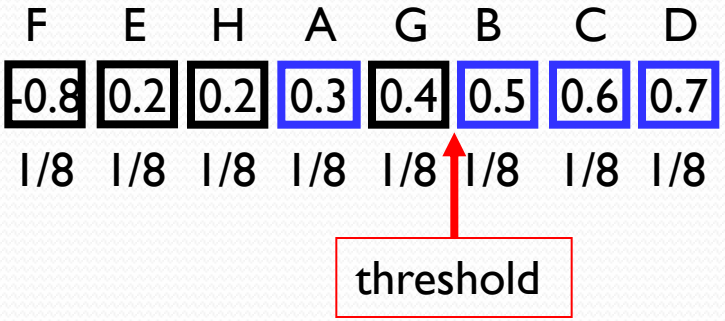
H(X) = sign(0.97 * h1(X))

It's the same as h1(x)

The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - ε_t = Sum $\{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\ \alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

The Best Error



$$D_{t+1}(x_i) = D_t(x_i)\ \exp(-\alpha_t\, y_i\, h_t\,(x_i))$$

$\exp(\alpha_t) = \exp(0.97) = 2.63$
$\exp(-\alpha_t) = \exp(-0.97) = 0.38$

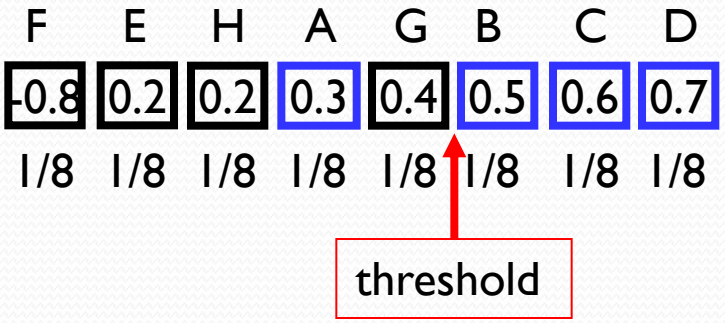
ID	E1	E2.	Class	Weight	Weight
A	0.3	-0.6	+1	1/8 * 2.63	0.33
B	0.5	-0.5	+1	1/8 * 0.38	0.05
C	0.7	-0.1	+1	1/8 * 0.38	0.05
D	0.6	-0.4	+1	1/8 * 0.38	0.05
E	0.2	0.4	-1	1/8 * 0.38	0.05
F	-0.8	0.1	-1	1/8 * 0.38	0.05
G	0.4	-0.9	-1	1/8 * 0.38	0.05
H	0.2	0.5	-1	1/8 * 0.38	0.05

Multiply the correctly classified instances by 0.38
Multiply incorrectly classified instances by 2.63

The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - ε_t = Sum $\{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\ \alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

The Best Error

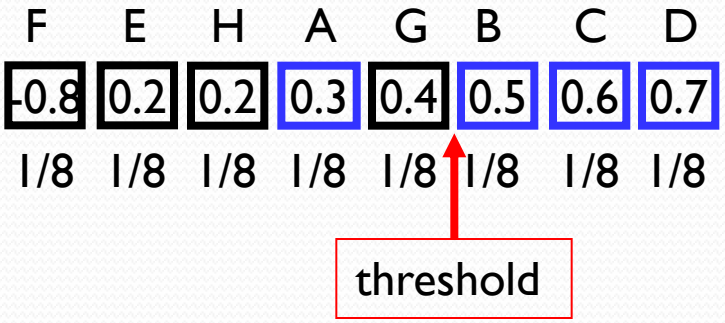


D' = D / sum(D)

ID	E1	E2.	Class	Weight	Weight	Weight
A	0.3	-0.6	+1	1/8 * 2.63	0.33	0.48
B	0.5	-0.5	+1	1/8 * 0.38	0.05	0.074
C	0.7	-0.1	+1	1/8 * 0.38	0.05	0.074
D	0.6	-0.4	+1	1/8 * 0.38	0.05	0.074
E	0.2	0.4	-1	1/8 * 0.38	0.05	0.074
F	-0.8	0.1	-1	1/8 * 0.38	0.05	0.074
G	0.4	-0.9	-1	1/8 * 0.38	0.05	0.074
H	0.2	0.5	-1	1/8 * 0.38	0.05	0.074

Multiply the correctly classified instances by 0.38
Multiply incorrectly classified instances by 2.63
Normalize to sum to 1.0

The Best Error



D' = D / sum(D)

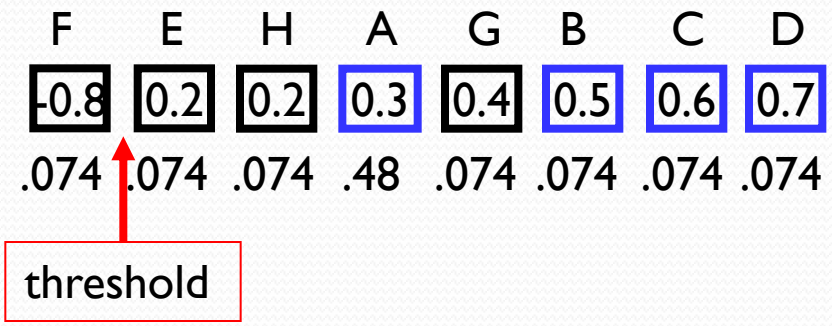
ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	0.48
B	0.5	-0.5	+1	0.074
C	0.7	-0.1	+1	0.074
D	0.6	-0.4	+1	0.074
E	0.2	0.4	-1	0.074
F	-0.8	0.1	-1	0.074
G	0.4	-0.9	-1	0.074
H	0.2	0.5	-1	0.074

Multiply the correctly classified instances by 0.38
Multiply incorrectly classified instances by 2.63
Normalize to sum to 1.0

The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - $\varepsilon_t = \text{Sum}\ \{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\ \alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

E1 classifier



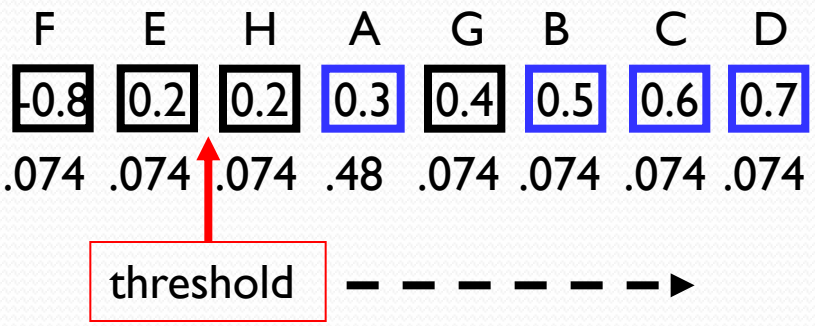
Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 0.222
Sign = -1, error = 0.778

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	0.48
B	0.5	-0.5	+1	0.074
C	0.7	-0.1	+1	0.074
D	0.6	-0.4	+1	0.074
E	0.2	0.4	-1	0.074
F	-0.8	0.1	-1	0.074
G	0.4	-0.9	-1	0.074
H	0.2	0.5	-1	0.074

E1 classifier



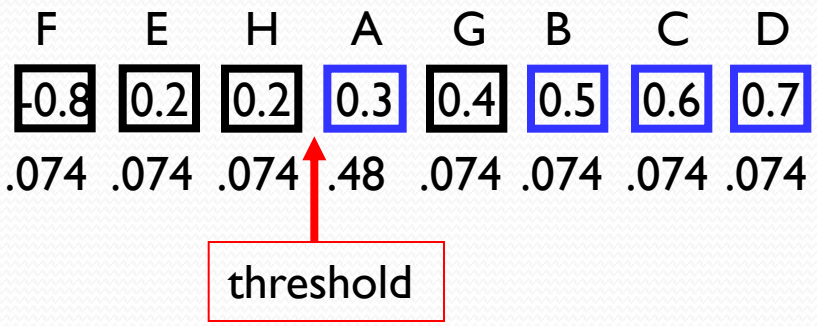
Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 0.148
Sign = -1, error = 0.852

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	0.48
B	0.5	-0.5	+1	0.074
C	0.7	-0.1	+1	0.074
D	0.6	-0.4	+1	0.074
E	0.2	0.4	-1	0.074
F	-0.8	0.1	-1	0.074
G	0.4	-0.9	-1	0.074
H	0.2	0.5	-1	0.074

The Best E1 classifier



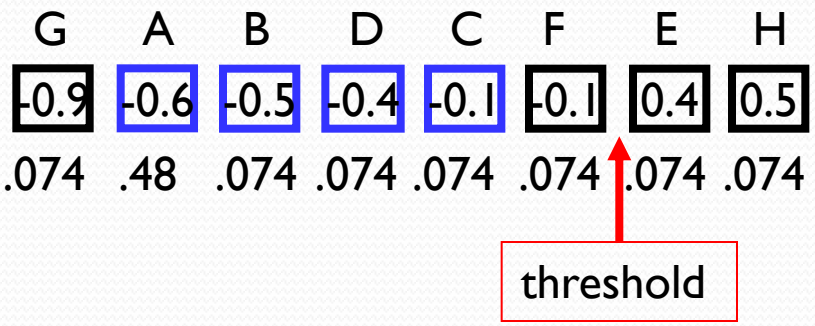
Classifier based on E1:
if (sign*wt(E1) > thresh) > 0)
face = true

sign = +1 or -1

Sign = +1, error = 0.074

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	0.48
B	0.5	-0.5	+1	0.074
C	0.7	-0.1	+1	0.074
D	0.6	-0.4	+1	0.074
E	0.2	0.4	-1	0.074
F	-0.8	0.1	-1	0.074
G	0.4	-0.9	-1	0.074
H	0.2	0.5	-1	0.074

The Best E2 classifier



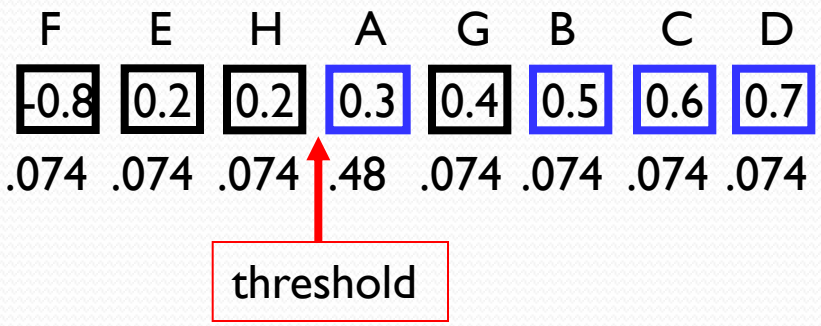
Classifier based on E2:
if (sign*wt(E2) > thresh) > 0)
face = true

sign = +1 or -1

Sign = -1, error = 0.148

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	0.48
B	0.5	-0.5	+1	0.074
C	0.7	-0.1	+1	0.074
D	0.6	-0.4	+1	0.074
E	0.2	0.4	-1	0.074
F	-0.8	-0.1	-1	0.074
G	0.4	-0.9	-1	0.074
H	0.2	0.5	-1	0.074

The Best Classifier



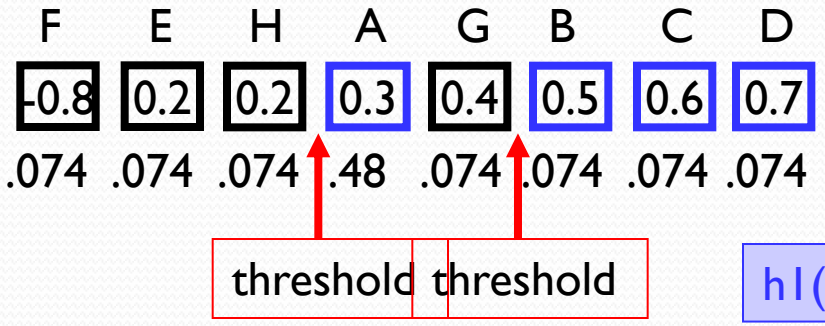
Classifier based on E1:
if (wt(E1) > 0.45) face = true

$\alpha = 0.5\ln((1-0.074) / 0.074)$
$= 1.26$

Sign = +1, error = 0.074

ID	E1	E2.	Class	Weight
A	0.3	-0.6	+1	0.48
B	0.5	-0.5	+1	0.074
C	0.7	-0.1	+1	0.074
D	0.6	-0.4	+1	0.074
E	0.2	0.4	-1	0.074
F	-0.8	0.1	-1	0.074
G	0.4	-0.9	-1	0.074
H	0.2	0.5	-1	0.074

The Boosted Classifier Thus Far

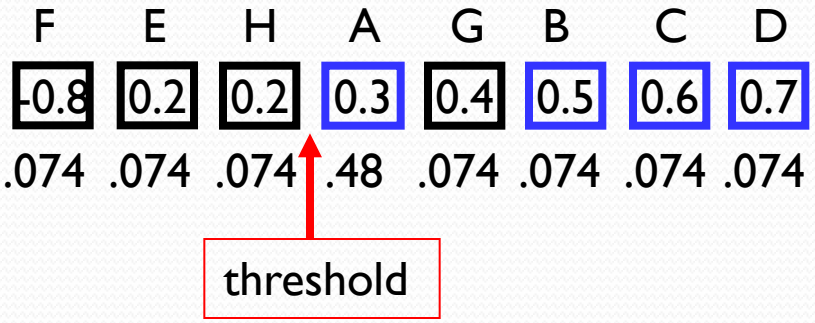


h1(X) = wt(E1) > 0.45 ? +1 : -1

h2(X) = wt(E1) > 0.25 ? +1 : -1

H(X) = sign(0.97 * h1(X) + 1.26 * h2(X))

Reweighting the Data



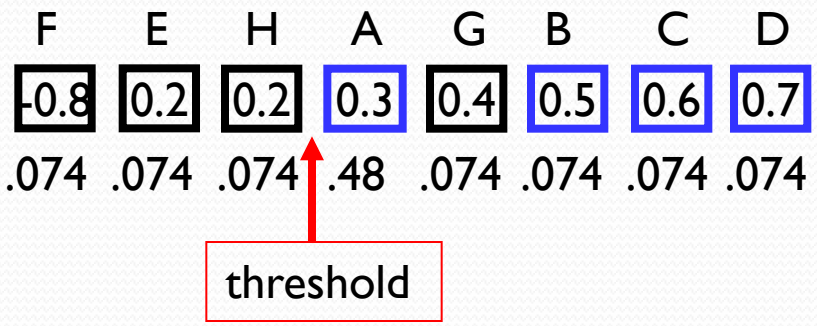
Exp(α) = exp(1.26) = 3.5
Exp(-α) = exp(-1.26) = 0.28

Sign = +1, error = 0.074

ID	E1	E2.	Class	Weight	
A	0.3	-0.6	+1	0.48*0.28	0.32
B	0.5	-0.5	+1	0.074*0.28	0.05
C	0.7	-0.1	+1	0.074*0.28	0.05
D	0.6	-0.4	+1	0.074*0.28	0.05
E	0.2	0.4	-1	0.074*0.28	0.05
F	-0.8	0.1	-1	0.074*0.28	0.05
G	0.4	-0.9	-1	0.074*3.5	0.38
H	0.2	0.5	-1	0.074*0.28	0.05

RENORMALIZE

Reweighting the Data



NOTE: THE WEIGHT OF "G" WHICH WAS MISCLASSIFIED BY THE SECOND CLASSIFIER IS NOW SUDDENLY HIGH

Sign = +1, error = 0.074

ID	E1	E2.	Class	Weight	
A	0.3	-0.6	+1	0.48*0.28	0.32
B	0.5	-0.5	+1	0.074*0.28	0.05
C	0.7	-0.1	+1	0.074*0.28	0.05
D	0.6	-0.4	+1	0.074*0.28	0.05
E	0.2	0.4	-1	0.074*0.28	0.05
F	-0.8	0.1	-1	0.074*0.28	0.05
G	0.4	-0.9	-1	0.074*3.5	0.38
H	0.2	0.5	-1	0.074*0.28	0.05

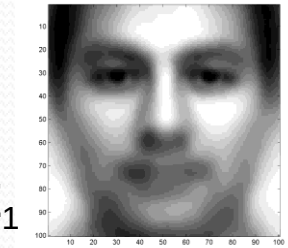
RENORMALIZE

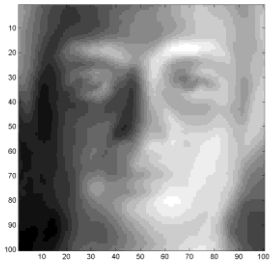
The AdaBoost Algorithm

- Initialize $D_1(x_i) = 1/N$
- For $t = 1, \ldots, T$
  - Train a weak classifier h_t using distribution D_t
  - Compute total error on training data
    - $\varepsilon_t = \text{Sum}\ \{D_t(x_i)\ ½(1 - y_i\ h_t(x_i))\}$
  - Set $\alpha_t = ½\ \ln((1 - \varepsilon_t)/\ \varepsilon_t)$
  - For $i = 1 \ldots N$
    - set $D_{t+1}(x_i) = D_t(x_i)\ \exp(-\ \alpha_t\ y_i\ h_t(x_i))$
  - Normalize D_{t+1} to make it a distribution
- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t\ \alpha_t\ h_t(x))$

ADA Boost

= 0.4 E1 - 0.4 E2

E_1 

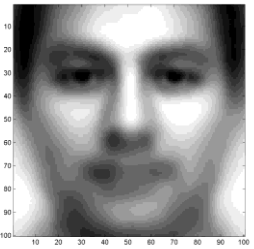
E_2 

- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t \alpha_t h_t(x))$

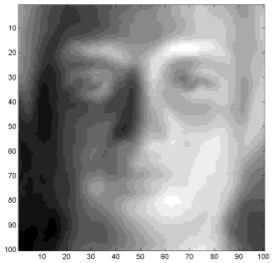
ADA Boost

= 0.4 E1 - 0.4 E2

E_1



E_2



- The final classifier is
  - $H(x) = \text{sign}(\Sigma_t \alpha_t h_t(x))$
- Not all classifiers need to use E1
- It is often a good idea to keep adding classifiers even after the training accuracy reaches 100%

Face Detection: History

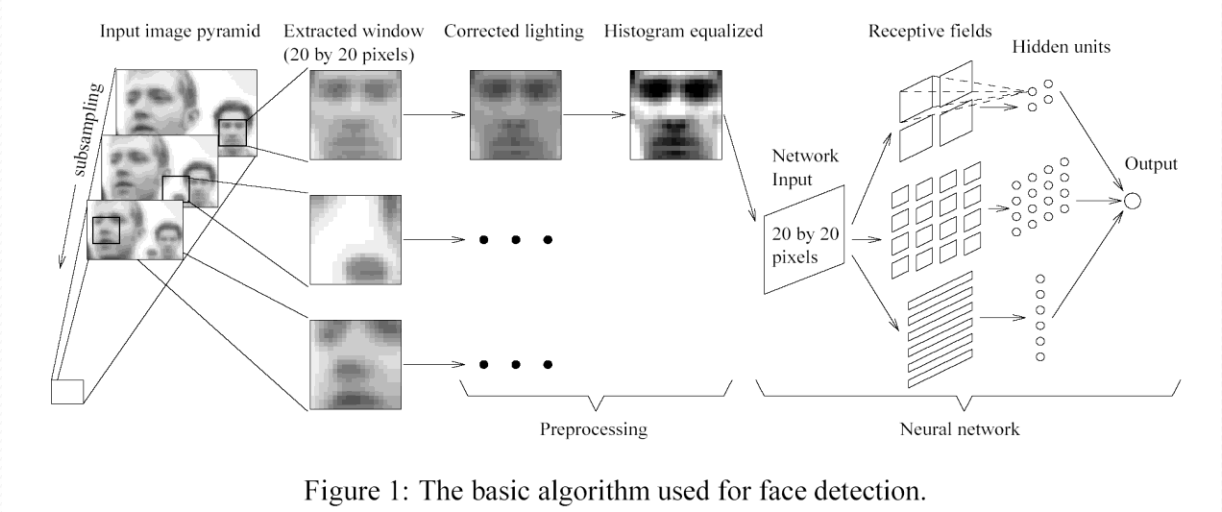


Figure 1: The basic algorithm used for face detection.

- Historically, tried complex models that tried to find eyes, nose, ears, etc…

Face Detection: History

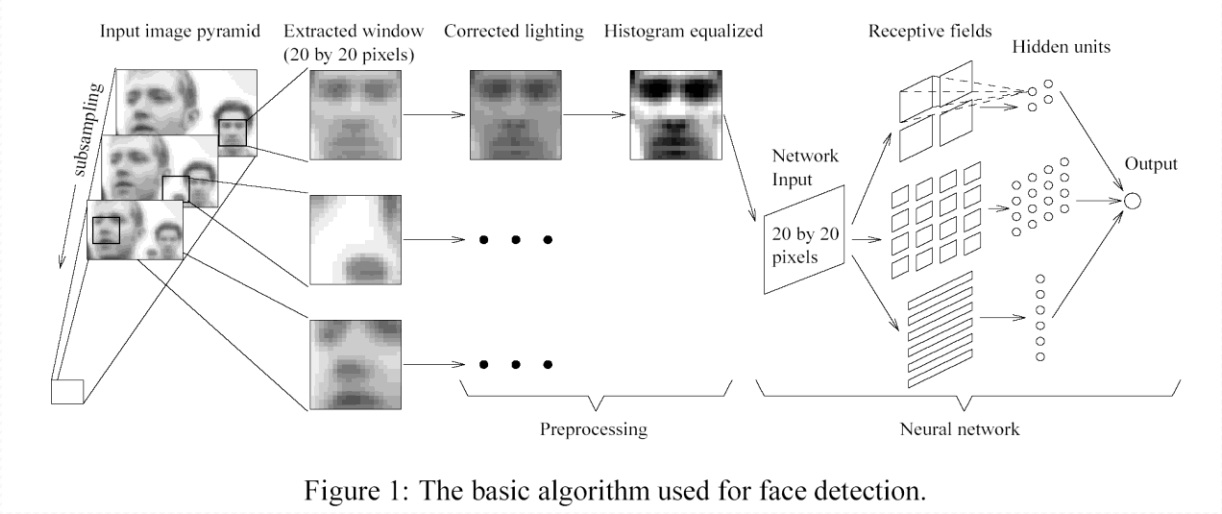


Figure 1: The basic algorithm used for face detection.

- Historically, tried complex models that tried to find eyes, nose, ears, etc…
- Viola-Jones burst onto the scene with boosted cascade classifiers

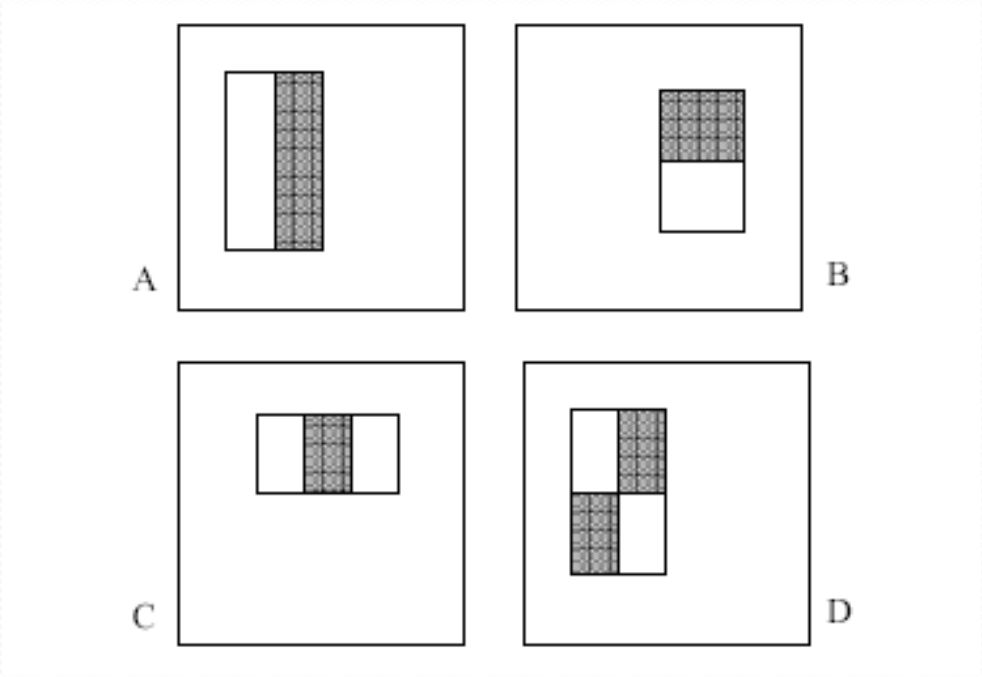
The problem of face detection

- 1. Defining Features
  - Should we be searching for noses, eyes, eyebrows etc.?
    - Nice, but expensive
  - Or something simpler
- 2. Selecting Features
  - Of all the possible features we can think of, which ones make sense
- 3. Classification: Combining evidence
  - How does one combine the evidence from the different features?

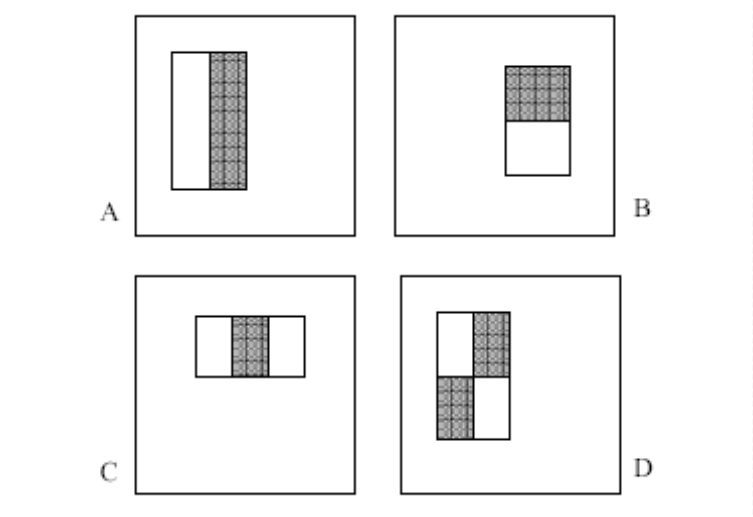
Viola Jones Method

- Very simple features
- Modified AdaBoost algorithm
- Cascading classifiers

Features



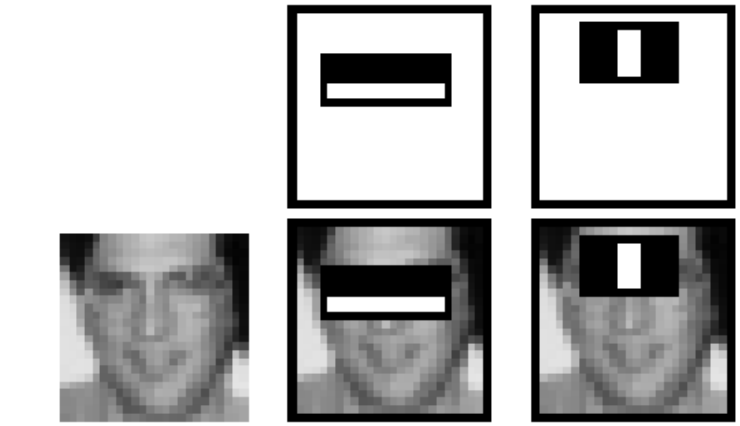
“Integral” Features



- Each checkerboard has the following parameters
  - Length
  - Width
  - Type (specifies type of the pattern)
- Viola and Jones only used the above 4 patterns.

Dot products

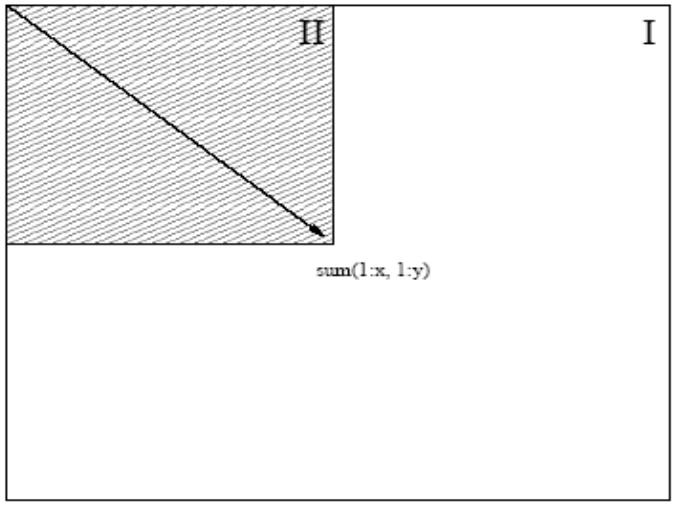
- Compute dot products between the checkerboard patterns and the images



Feature= (sum of pixels in light region) – (sum of pixels in dark region)

Integral images

- Summed area tables



- For each pixel store the sum of ALL pixels to the left of and above it.

Fast computation of Integral sums

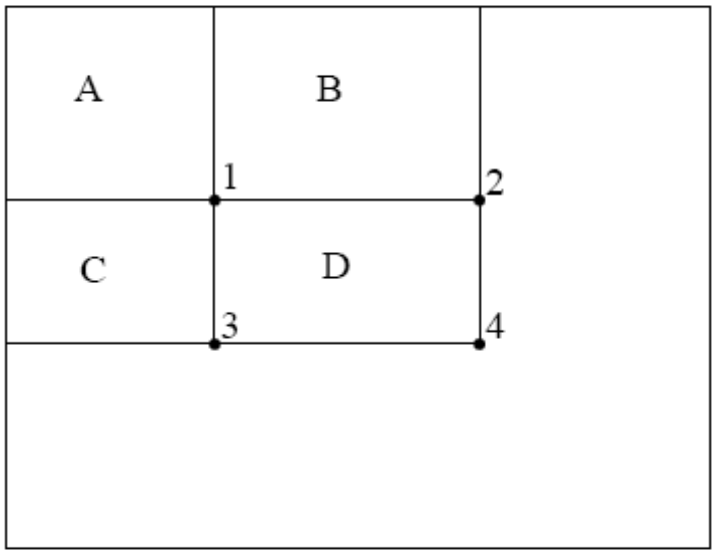
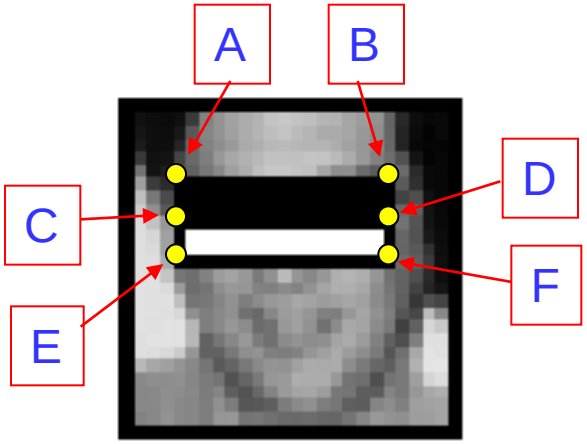


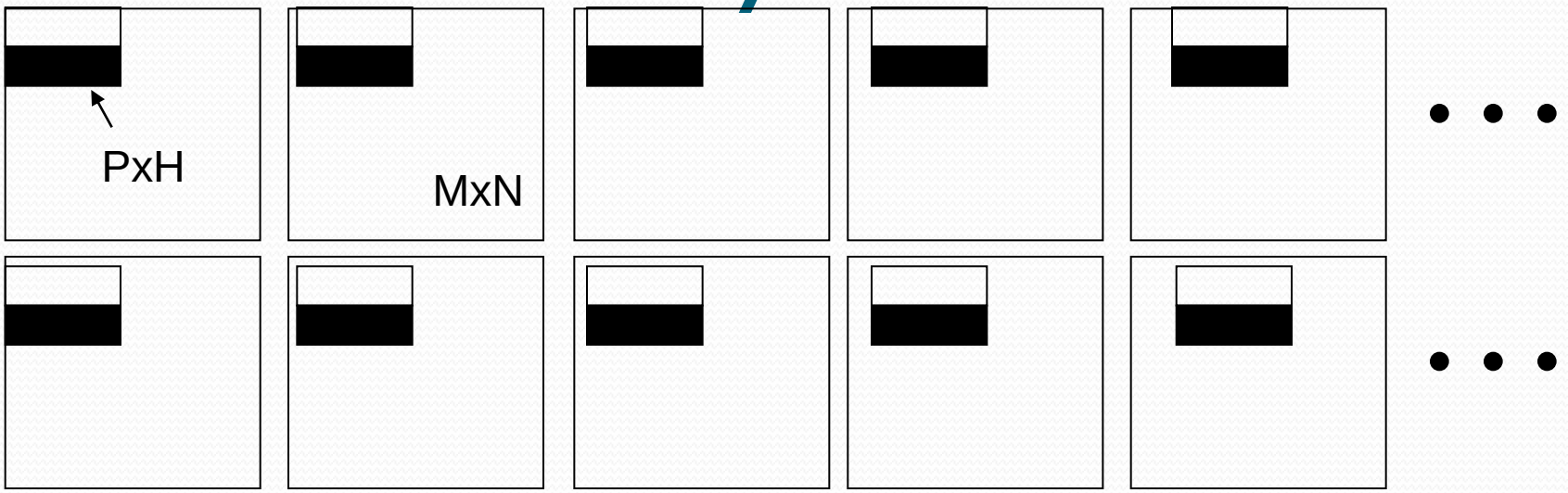
Figure 3: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.

A Fast Way to Compute the Feature



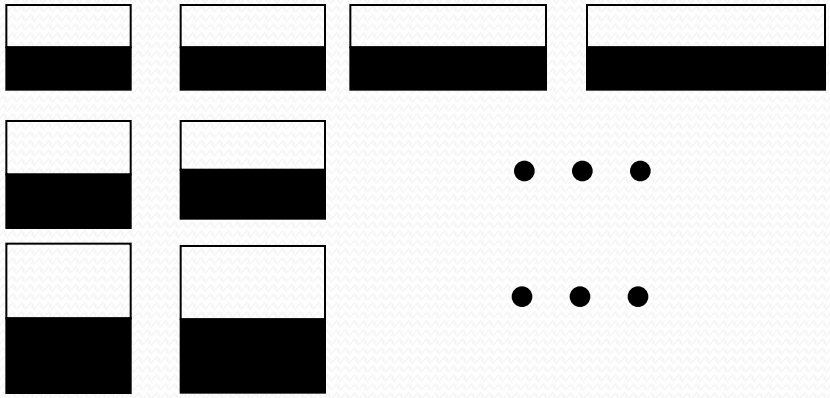
- Store pixel table for every pixel in the image
  - The sum of all pixel values to the left of and above the pixel
- Let A, B, C, D, E, F be the pixel table values at the locations shown
  - Total pixel value of black area = D + A – B – C
  - Total pixel value of white area = F + C – D – E
  - Feature value = (F + C – D – E) – (D + A – B – C)

How many features?



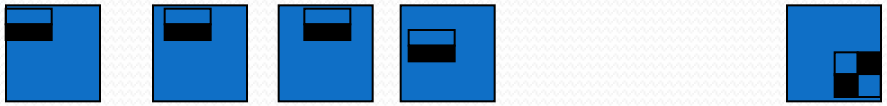
- Each checker board of width P and height H can start at
  - (0,0), (0,1),(0,2), … (0, N-P)
  - (1,0), (1,1),(1,2), … (1, N-P)
  - ..
  - (M-H,0), (M-H,1), (M-H,2), … (M-H, N-P)
- (M-H)*(N-P) possible starting locations
  - Each is a unique checker feature
    - E.g. at one location it may measure the forehead, at another the chin

How many features?



- Each feature can have many sizes
  - Width from (min) to (max) pixels
  - Height from (min ht) to (max ht) pixels
- At each size, there can be many starting locations
  - Total number of possible checkerboards of one type:
    No. of possible sizes x No. of possible locations
- There are four types of checkerboards
  - Total no. of possible checkerboards: VERY VERY LARGE!

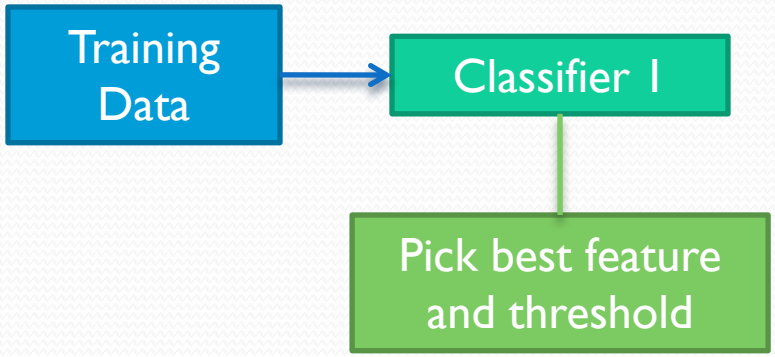
No. of features



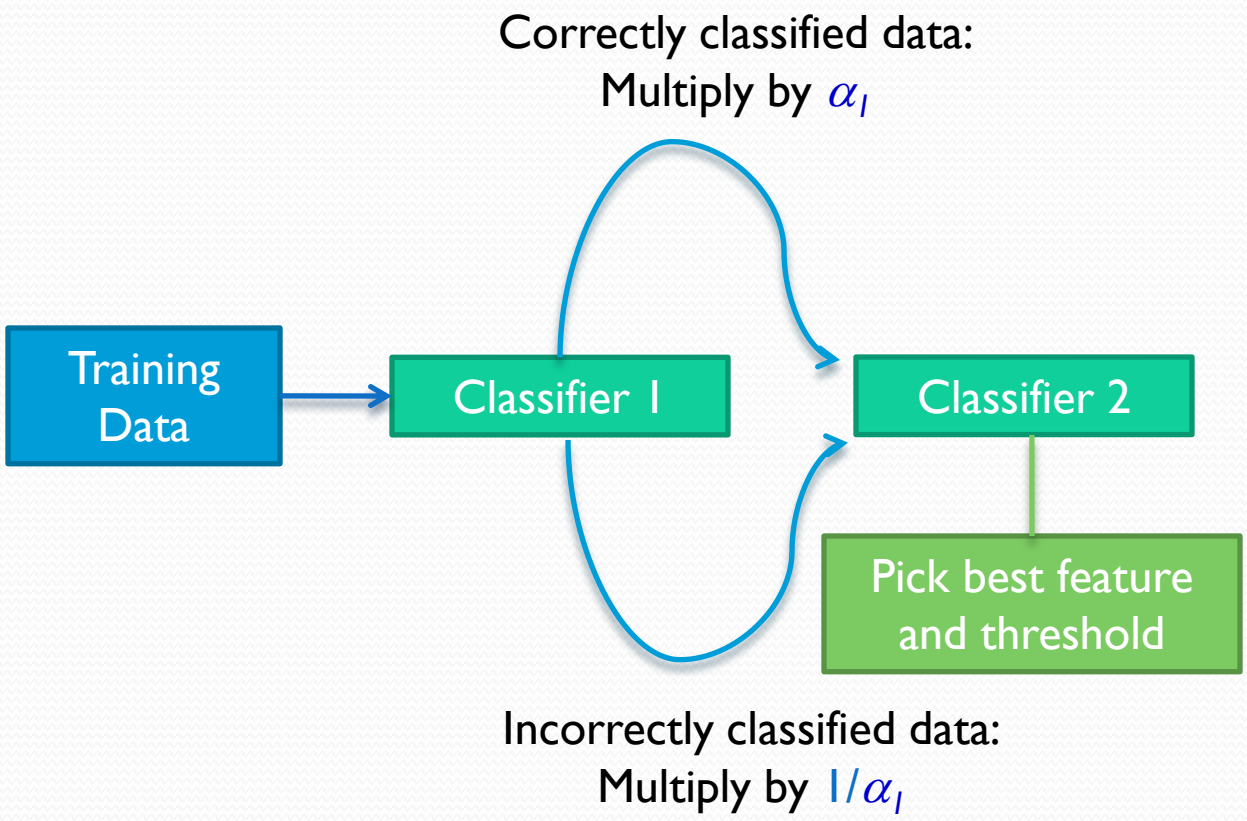
	F1	F2	F3	F4	…..	F180000
	7	9	2	-1	…..	12
	-11	3	19	17	…..	2

- Each possible checkerboard gives us one feature
- A total of up to 180000 features derived from a 24x24 image!
- Every 24x24 image is now represented by a set of 180000 numbers
  - This is the set of features we will use for classifying if it is a face or not!

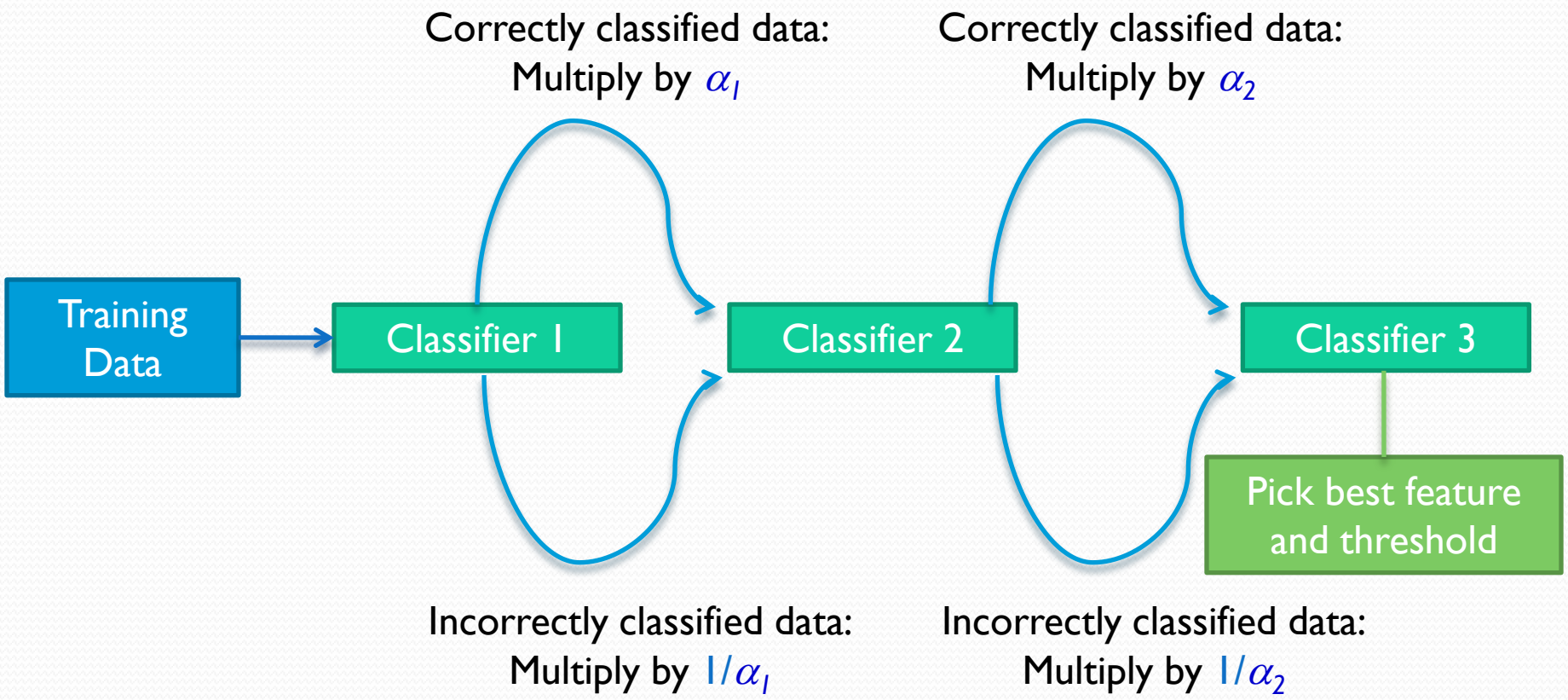
AdaBoost


Training Data
Classifier 1
Pick best feature and threshold

AdaBoost


Correctly classified data:
Multiply by α_1
Training Data
Classifier 1
Classifier 2
Pick best feature and threshold
Incorrectly classified data:
Multiply by $1/\alpha_1$

AdaBoost


Correctly classified data:
Multiply by α_1
Correctly classified data:
Multiply by α_2
Training Data
Classifier 1
Classifier 2
Classifier 3
Pick best feature and threshold
Incorrectly classified data:
Multiply by $1/\alpha_1$
Incorrectly classified data:
Multiply by $1/\alpha_2$

AdaBoost: The weak learner

> Threshold

then

Face detected !

Problems

- This only classifies images as face/non-face
- There can be multiple faces in an image
- Faces can be bigger than 24 x 24

Fixing the multiple face problem

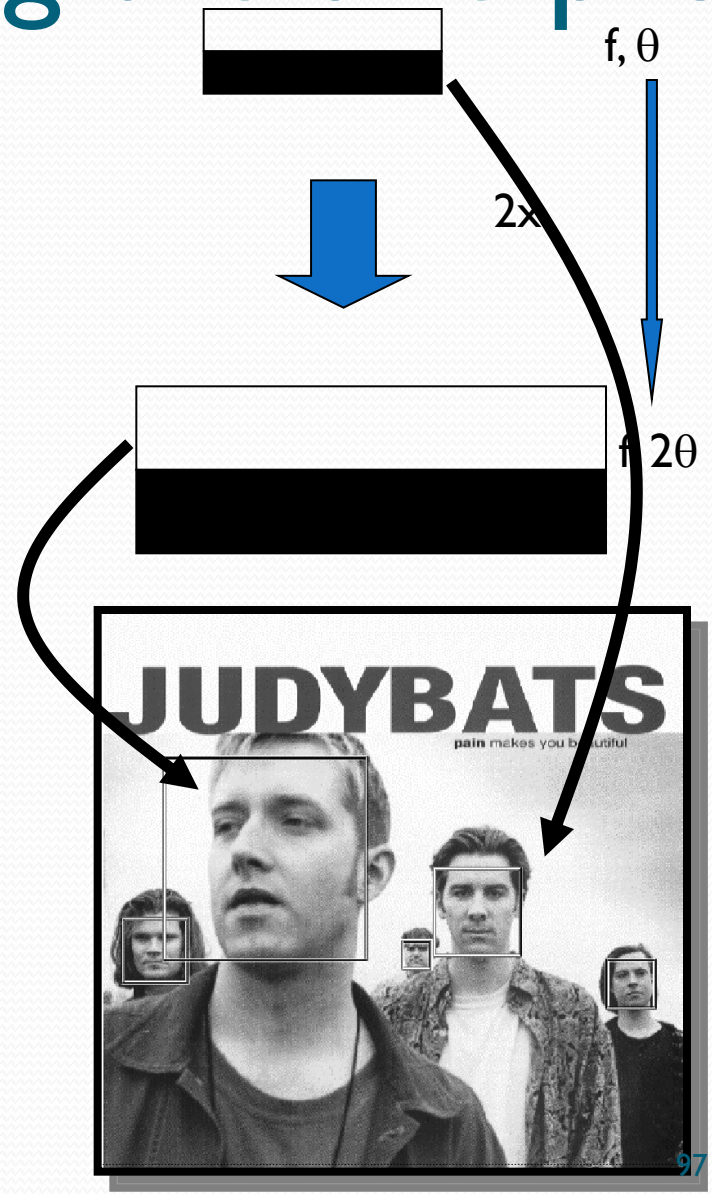
Fixing the multiple face problem

Fixing the multiple face problem

Fixing the multiple face problem

Fixing the multiple face problem

Fixing the size problem



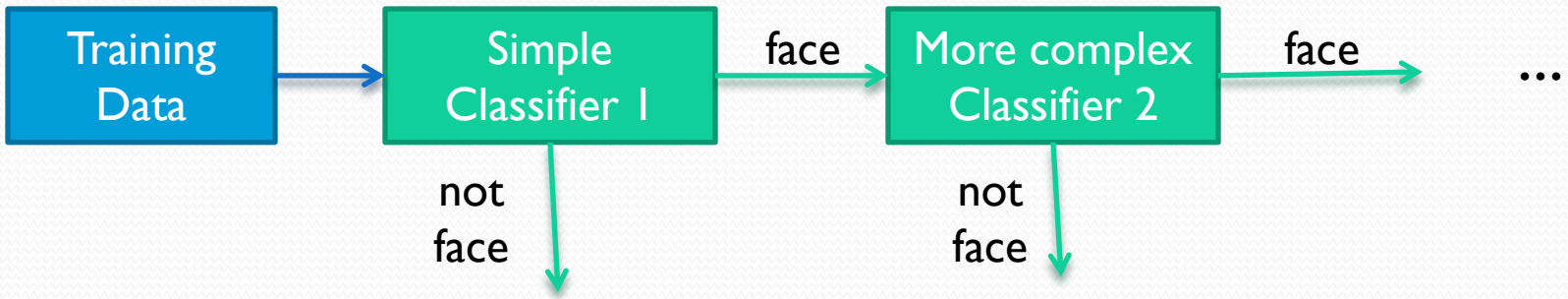
But this is still too slow ☹

Not all classifiers are equally fast

All animals are equal. Some animals are more equal than others

- George Orwell's Animal Farm

Basic Idea


Training Data
Simple Classifier 1
face
More complex Classifier 2
face
...
not face
not face

Modifying the classifier

- The normal AdaBoost trained classifier tries to optimize for accuracy
- We want it to err on the side of never missing a face

Modifying the classifier

- False rejection: Failing to detect a face
- False detection: Detecting a face where there is none

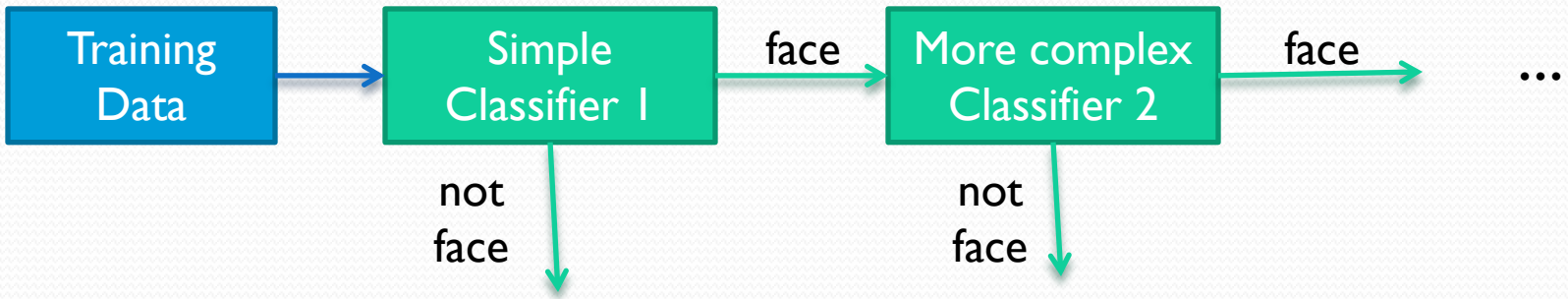
Modifying the classifier

- False rejection: Failing to detect a face
- False detection: Detecting a face where there is none

- Classifier:
  - Standard boosted classifier: $H(x) = \text{sign}(\Sigma_t \alpha_t h_t(x))$
  - Modified classifier $H(x) = \text{sign}(\Sigma_t \alpha_t h_t(x) + Y)$

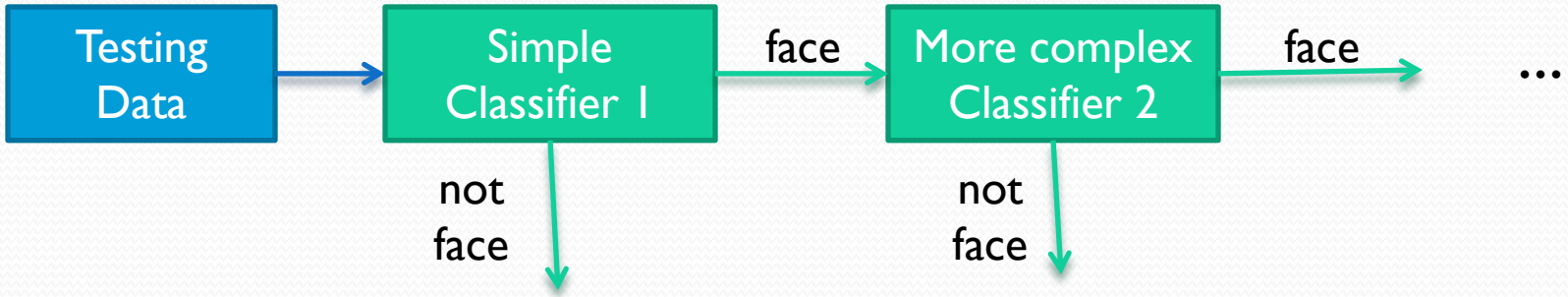
Biasing the classifier
towards False Detection

Basic Idea



Retrain the system using AdaBoost with the system this way

Testing

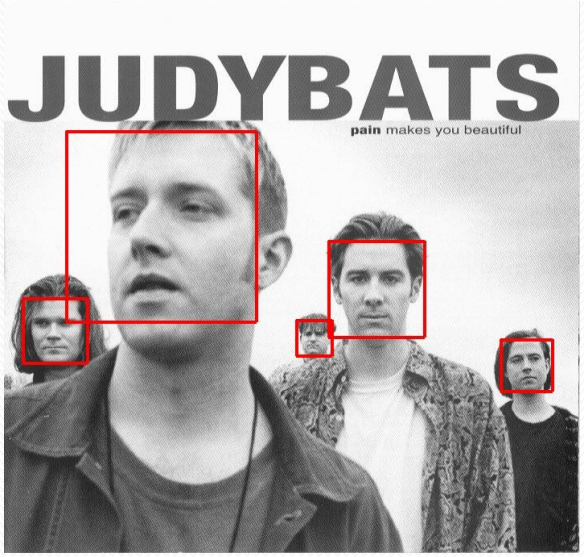


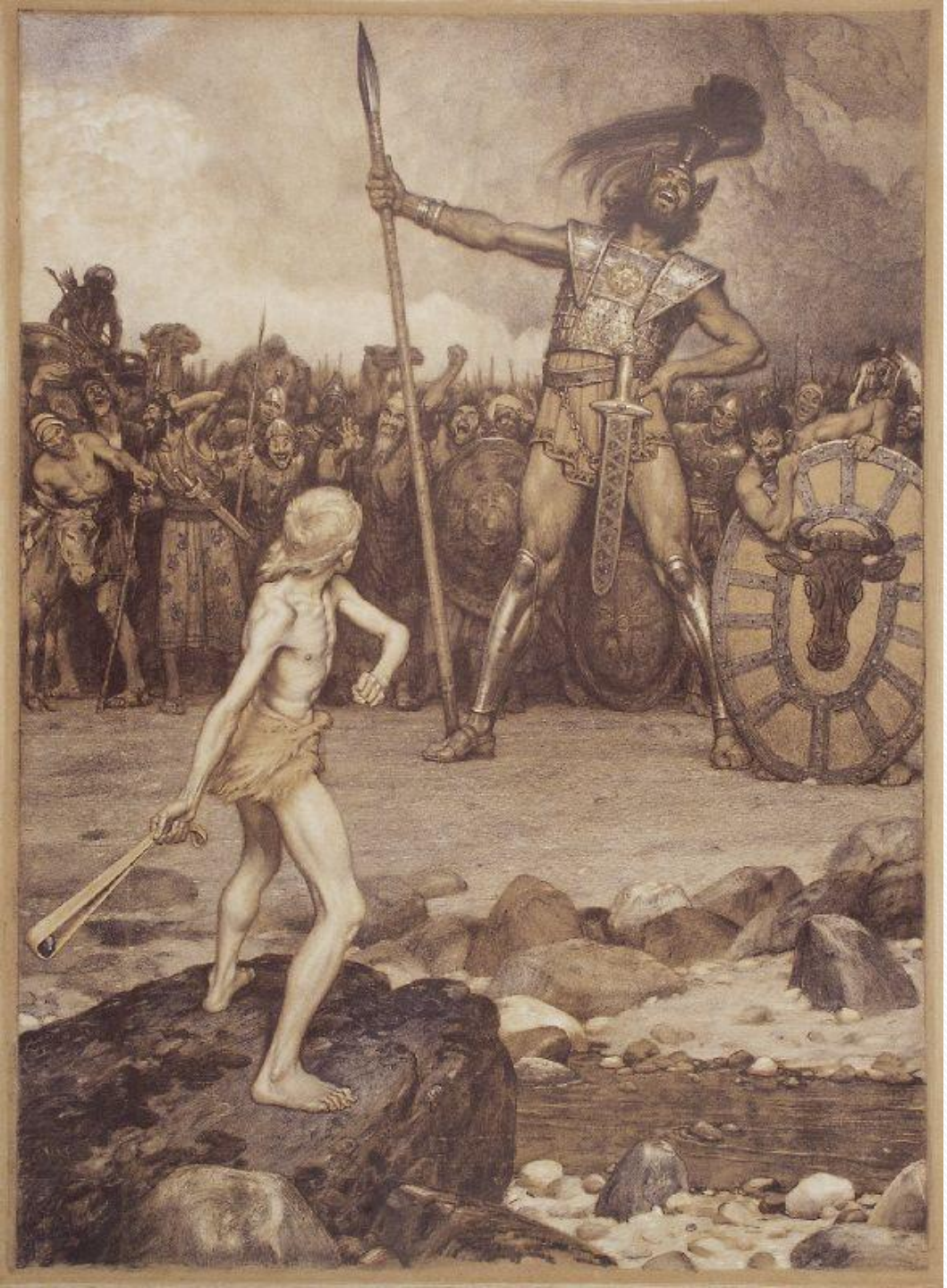
38 classifiers in sequence with over 6000 features

15x times faster than best systems at that time !!!

Sample results using the Viola-Jones Detector

- Notice detection at multiple scales





Sometimes, the weak can beat the strong

Sometimes, the weak can beat the strong