1. Objective

Boosting is a general method for improving the accuracy of any given learning algorithm. Specifically, one can use it to combine weak learners, each performing only slightly better than random guess, to form an arbitrarily good hypothesis. In this project, you are required to implement an AdaBoost and RealBoost algorithms for frontal human face detection.

2. Data

- **Training data:** Face and non-face images of the size of 16x16 pixels are given.
- **Testing data:** Two photos taken at the class are used for testing.
- **Hard negatives:** Three background images are taken without faces.

3. Tasks

Perform the following tasks. Report your results in the given order and write a concise interpretation for each result:

1. **Construct weak classifiers $\{h_j\}$:** Load the predefined set of Haar filters. Compute the features $\{f_j\}$ by applying each Haar filter to the integral images of the positive and negative populations. Determine the polarity s_j and threshold θ_j for each weak classifier h_j:

$$h_j(x) = \begin{cases} -s_j & \text{if } f_j < \theta_j, \\ s_j & \text{otherwise.} \end{cases}$$

Write a function which returns the weak classifier with lowest weighted error. Note, as the samples change their weights over time, the histograms and threshold θ will change.

2. **Implement AdaBoost:** Implement the AdaBoost algorithm to boost the weak classifiers. Construct the strong classifier $H(x)$ as an weighted ensemble of T weak classifiers:

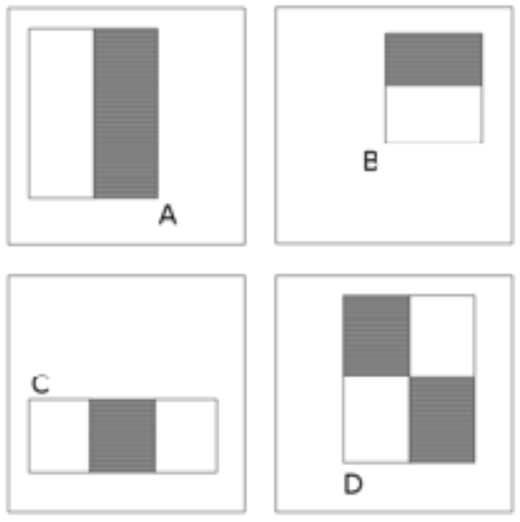
$$H(x) = sign(F(x)) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

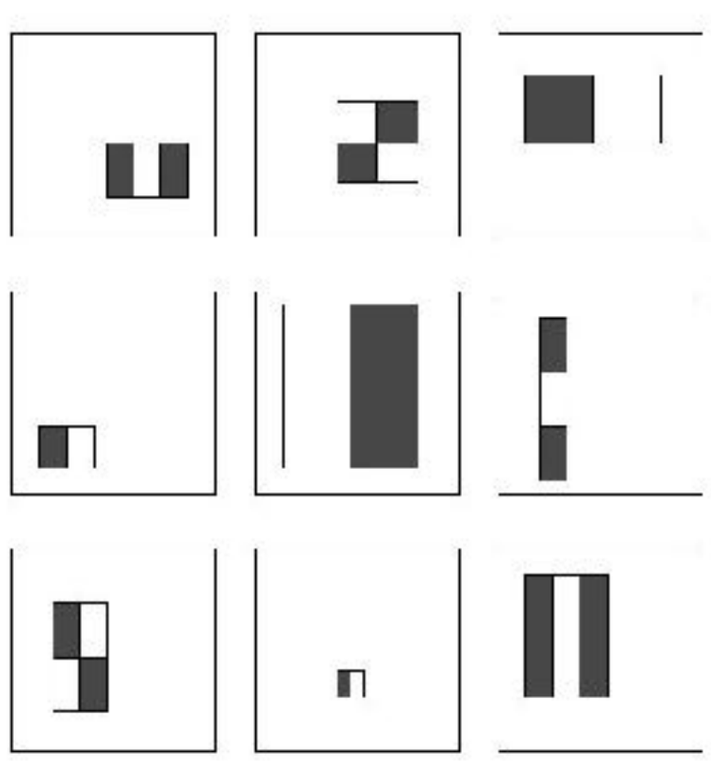
Two class photos are given. Note, you need to scale the image into a few scales so that the faces at the front and back are 16x16 pixels in one of the scaled image. Run your classifier on these images. Perform non-maximum suppression, i.e. when two positive detections overlap significantly, choose the one that has higher score. Perform hard negatives mining. You are given background images without faces. Run your strong classifier on these images. Any "faces" detected by your classifier are called "hard negatives". Add them to the negative population in the training set and re-train your model. Include the following in your report:

3. **Implement RealBoost:** Implement the RealBoost algorithm using the top $T = 10, 50, 100$ features you chose in step (c). Compute the histograms of negative and positive populations and the corresponding ROC curves. Include the following in your report:

Haar filter

- Haar-like feature by Viola-Jones inspired by Haar wavelets.
- Fast computation using integral images.
- Response = $\sum$(pixels in white area) - $\sum$(pixels in black area).

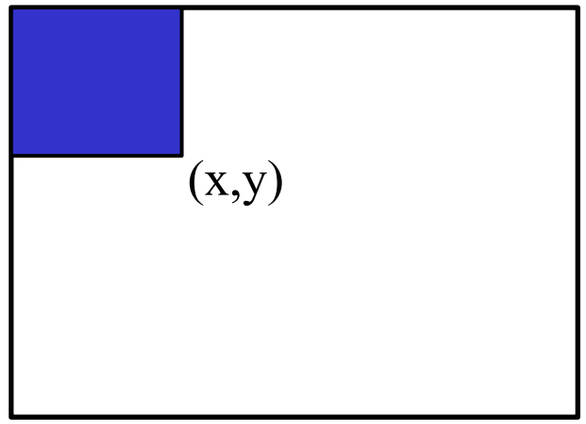




Integral images

- The *integral image* computes a value at each pixel (*x*,*y*) that is the sum of the pixel values i(x,y) above and to the left of (*x*,*y*), inclusive:

$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

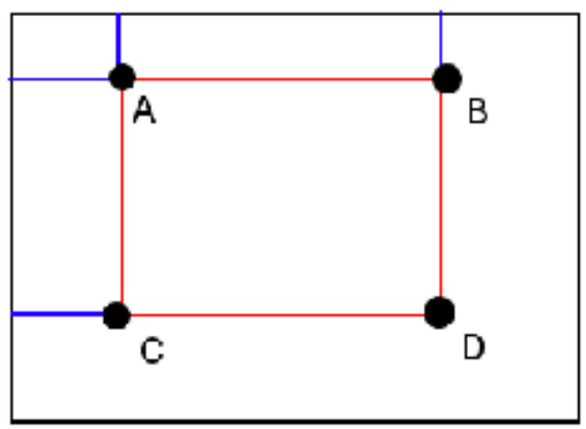


- This can quickly be computed in one pass through the image:

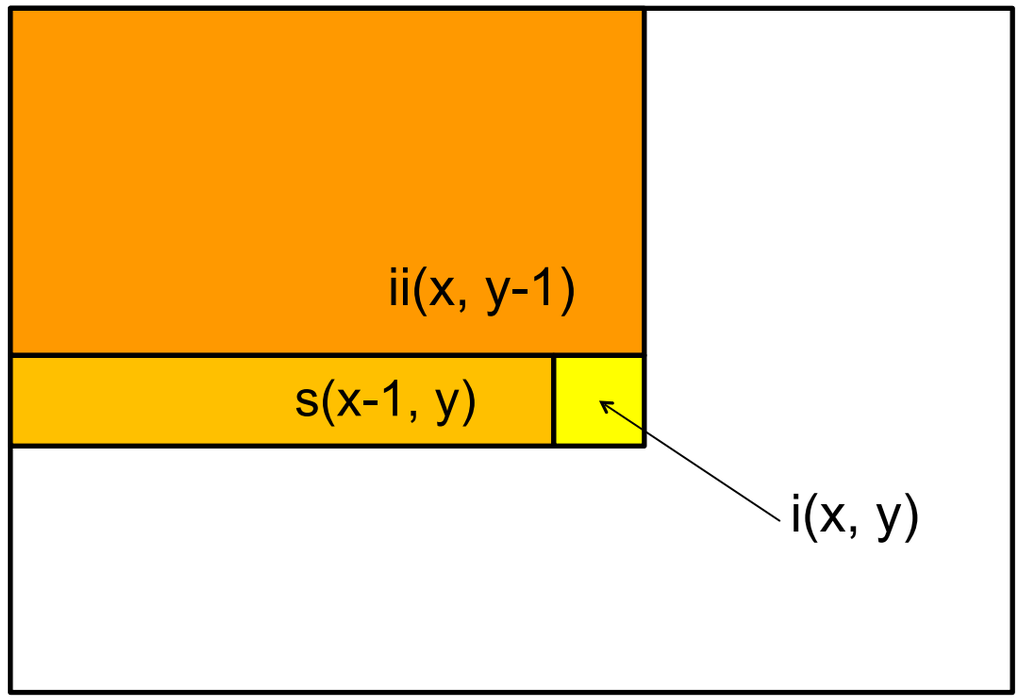
$$I(x,y) = i(x,y) + I(x,y-1) + I(x-1,y) - I(x-1,y-1)$$

- The sum of i(x,y) over the rectangle spanned by A, B, C and D is:

$$\sum_{\substack{x_0 < x \le x_1 \\ y_0 < y \le y_1}} i(x,y) = I(D) + I(A) - I(B) - I(C)$$



Sum = D - B - C + A

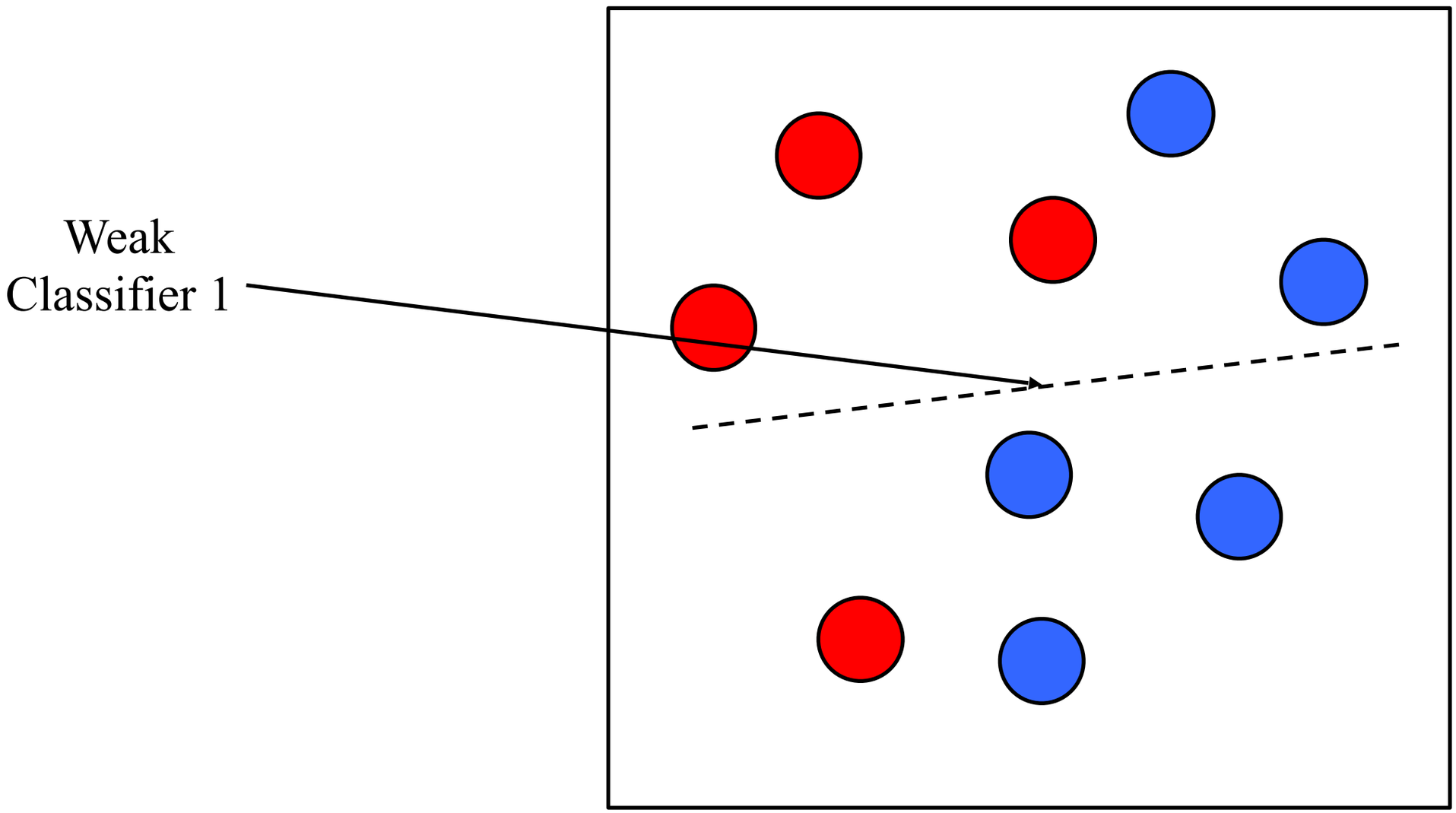


Cumulative row sum: $s(x, y) = s(x–1, y) + i(x, y)$

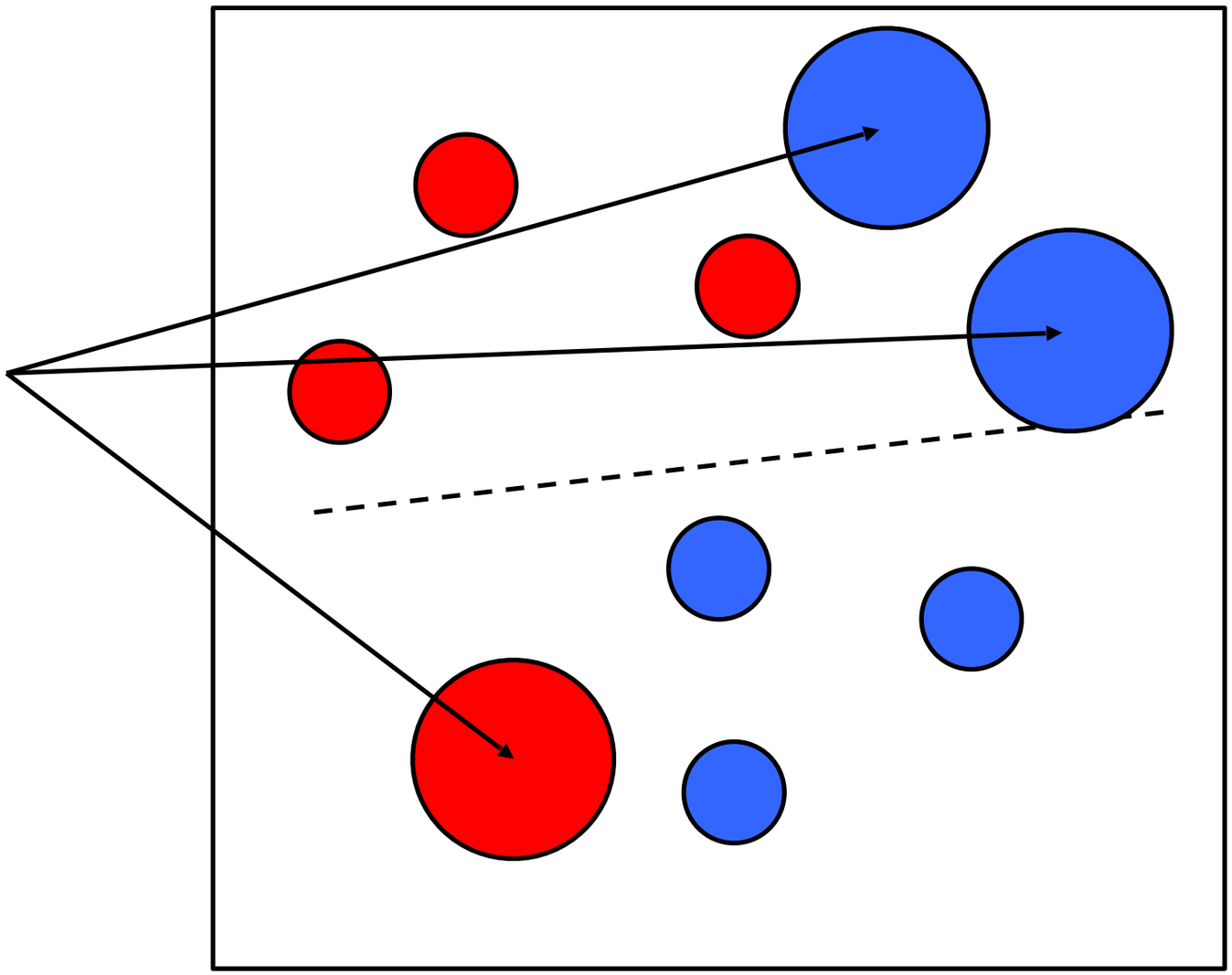
Integral image: $ii(x, y) = ii(x, y−1) + s(x, y)$

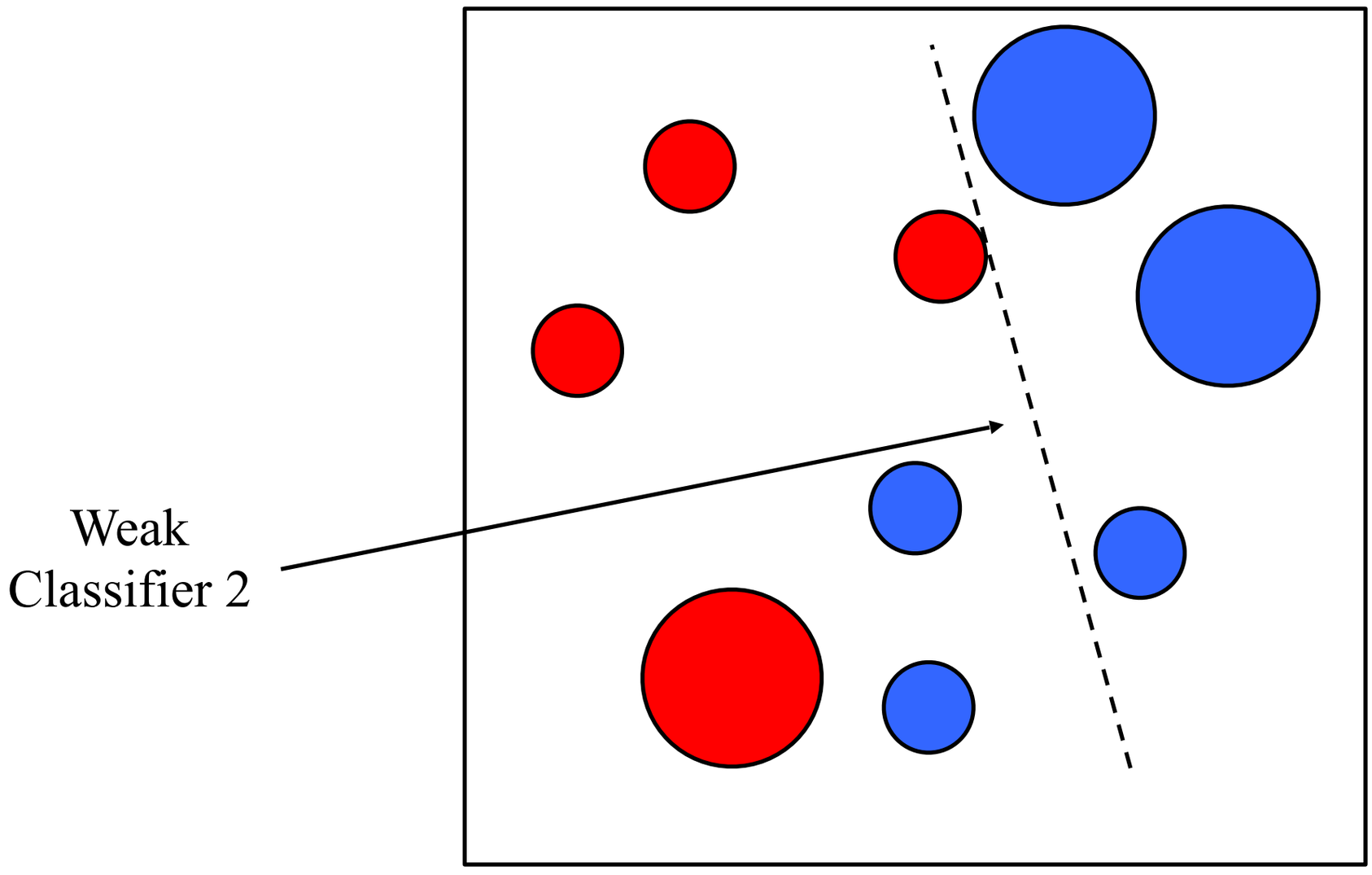
MATLAB: `ii = cumsum(cumsum(double(i)), 2);`

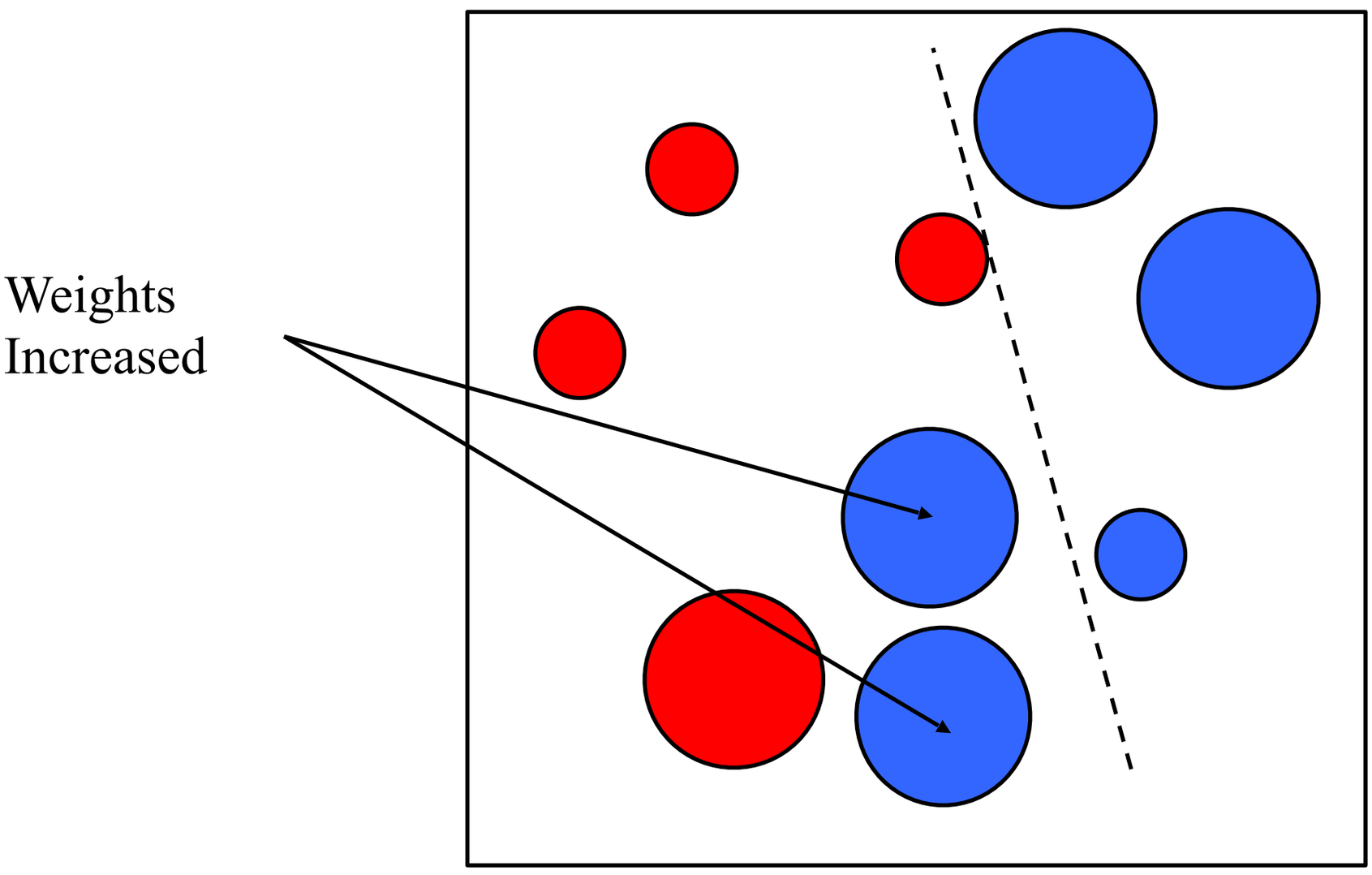
AdaBoost

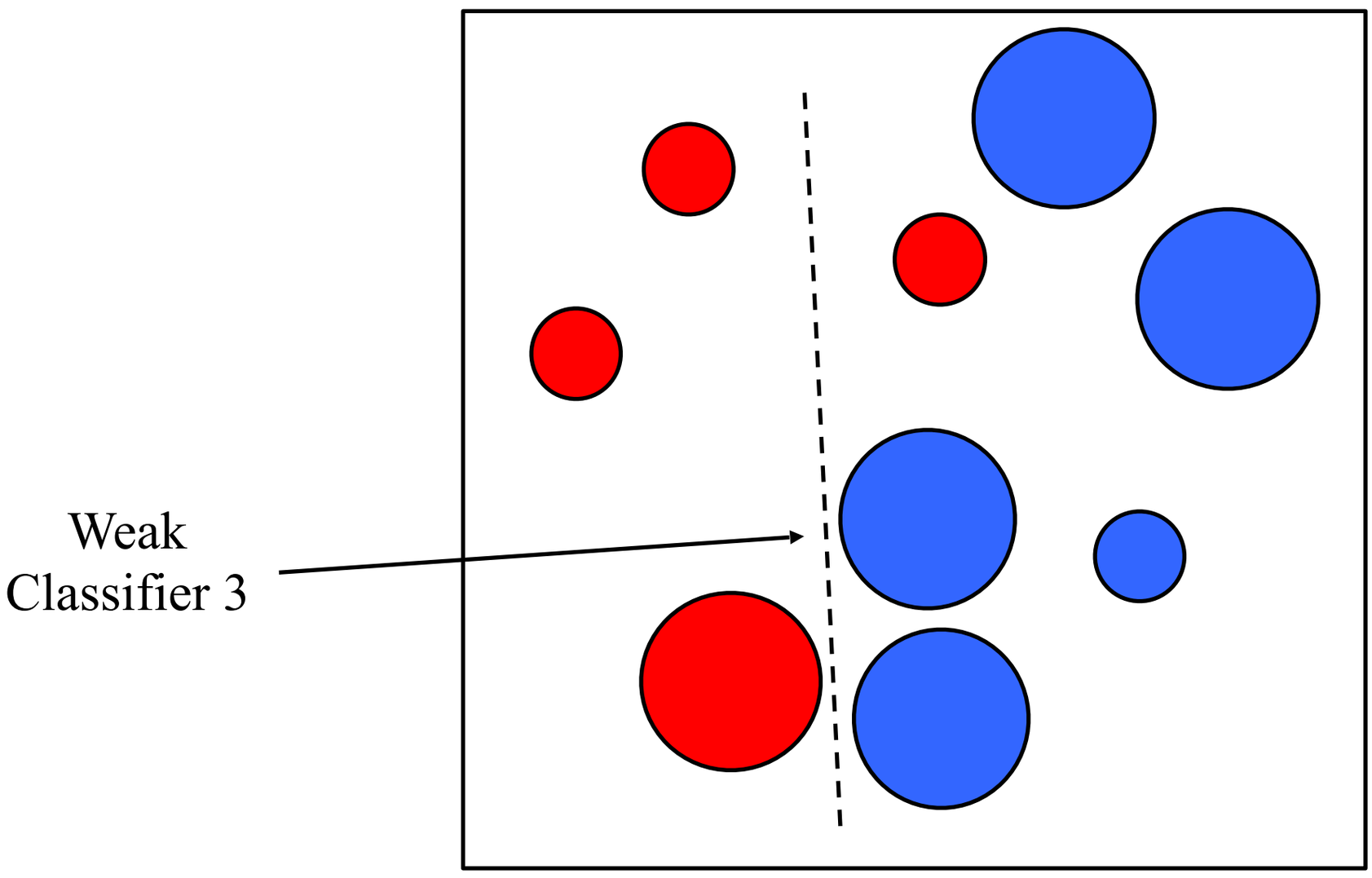

Weak
Classifier 1

Weights
Increased

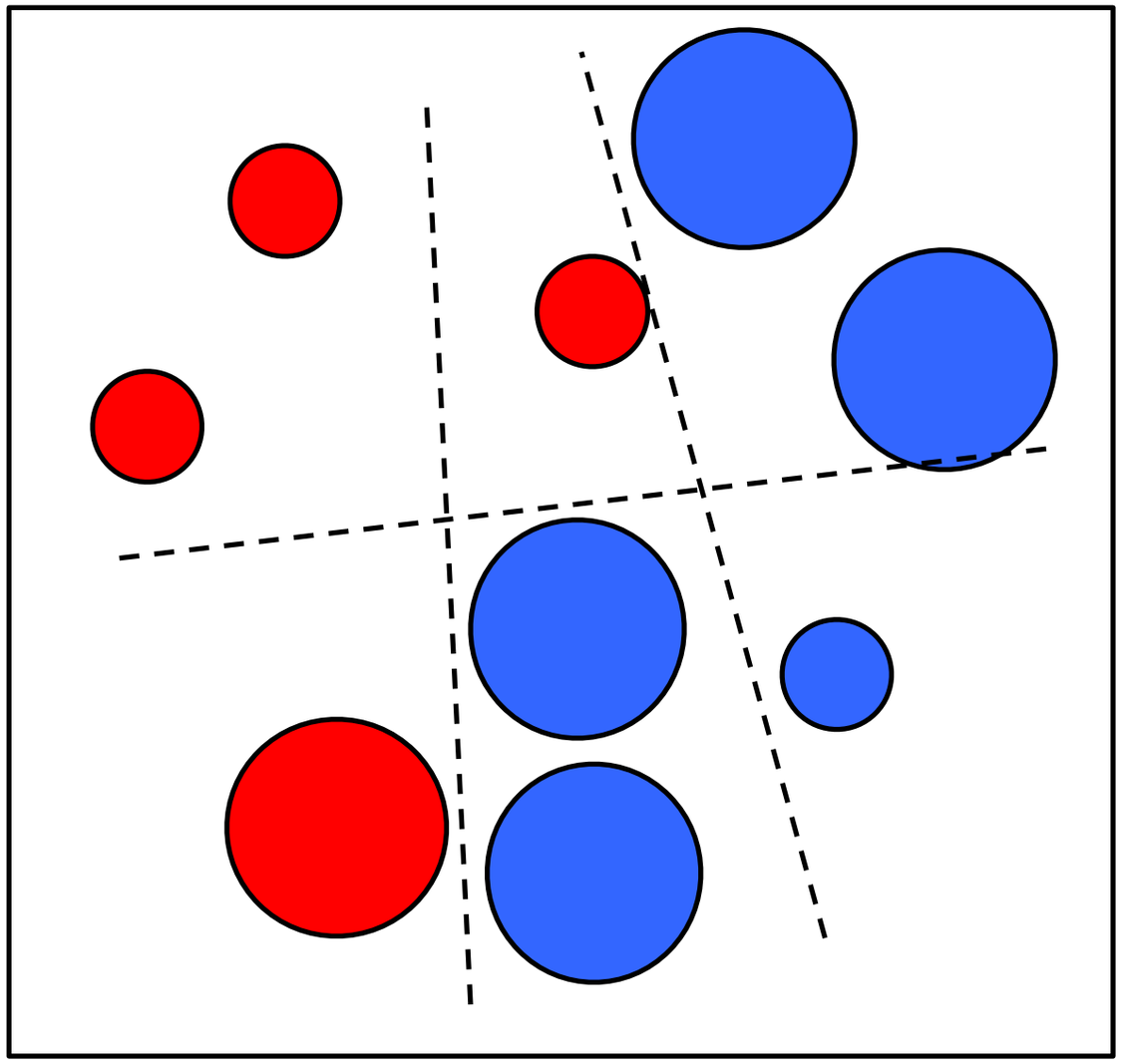



Weak
Classifier 2


Weights
Increased


Weak
Classifier 3

Strong classifier is
a combination of weak
classifiers



AdaBoost is a committee machine, which consists of a set of weak classifiers $h_k(x_i) \in \{+, -\}$, $t = 1, \ldots, T$. The final strong classifier is a perceptron based on the weak classifiers,

$$y_i = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x_i)\right), \tag{1}$$

where α_k can be interpreted as the weight of the vote of weak classifier k. The loss is in the exponential form:

$$\ell(\alpha) = \sum_{i=1}^{n} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right). \tag{2}$$

When training the strong classifier, we sequentially add members to the committee. Suppose the current committee has m classifiers, and we want to add a new member h_{new}. After adding a new member, the loss function becomes

$$\ell(\alpha_{new}, h_{new}) = \sum_{i=1}^{n} \exp\left(-y_i \sum_{t=1}^{m} \alpha_t h_t(x_i) + \alpha_{new} h_{new}(x_i)\right), \tag{3}$$

where we assume the current members and their weights of votes are fixed.

Take derivative

$$\frac{\partial \ell}{\partial \alpha_{new}} = \sum_{i=1}^{n} \exp\left(-y_i(\sum_{t=1}^{m} \alpha_t h_t(x_i) + \alpha_{new} h_{new}(x_i))\right) \cdot (-y_i h_{new}(x_i)). \quad (4)$$

When choosing ***a new member***, consider the current committee without adding a new member, $\alpha_{new} = 0$, then the above gradient can be written as

$$\left.\frac{\partial \ell}{\partial \alpha_{new}}\right|_{\alpha_{new}=0} = \sum_{i=1}^{n} \exp\left(-y_i(\sum_{t=1}^{m} \alpha_t h_t(x_i))\right) \cdot (-y_i h_{new}(x_i)) \quad (5)$$

$$= -\sum_{i=1}^{n} w_i y_i h_{new}(x_i) \quad (6)$$

where

$$w_i = \exp\left(-y_i \sum_{t=1}^{m} \alpha_t h_t(x_i)\right). \quad (7)$$

Then normalize $w_i \leftarrow w_i / \sum_{i=1}^{n} w_i$ to make it a distribution. Observe, this distribution focuses on those examples that are not well classified by the current committee. Thus, we want to choose a weak classifier h_{new} by maximizing $\sum_{i=1}^{n} w_i y_i h_{new}(x_i)$

$$h_{new} = \arg\max_{h_{new}} \sum_{i=1}^{n} w_i y_i h_{new}(x_i) \tag{8}$$

for the steepest drop in loss. Note, this implies that we want to choose the new weak classifier based on current distribution of the data (x_i, y_i, w_i) where the weights w_i keep changing.

To ***determine the voting weight*** α_{new}, we set the derivative to 0,

$$\frac{\partial \ell}{\partial \alpha_{new}} = 0, \tag{9}$$

$$\sum_{i=1}^{n} w_i \exp(-y_i \alpha_{new} h_{new}(x_i)) \cdot y_i h_{new}(x_i) = 0, \tag{10}$$

$$\sum_{i \in correct} w_i \exp(-\alpha_{new}) = \sum_{i \in wrong} w_i \exp(\alpha_{new}), \tag{11}$$

$$\sum_{i \in correct} w_i = \sum_{i \in wrong} w_i \exp(2\alpha_{new}), \tag{12}$$

$$\alpha_{new} = \frac{1}{2} \log \left(\frac{\sum_{i \in correct} w_i}{\sum_{i \in wrong} w_i} \right). \tag{13}$$

Note, in (11) we used the fact that if $i \in correct$, then $y_i h_{new}(x_i) = 1$. If we define the error rate as

$$\epsilon = \frac{\sum_{i \in wrong} w_i}{\sum_i w_i}, \tag{14}$$

then α_{new} is

$$\alpha_{new} = \frac{1}{2} \log \frac{1 - \epsilon}{\epsilon}. \tag{15}$$

Algorithm 1 AdaBoost learning

Input:

- Samples $\{x_i : i = 1, \ldots, n\}$,
- Desired outputs $\{y_i \in \{-1, +1\} : i = 1, \ldots, n\}$,
- Weak learners $H = \{h : x \to \{-1, +1\}\}$,
- Steps T.

Output:

- Ensemble $\{h_t : t = 1, \ldots, T\}$,
- Voting weights $\{\alpha_t : t = 1, \ldots, T\}$.

Steps:

1: Let $\{w_{1,i} = \frac{1}{n} : i = 1, \ldots, n\}$.

2: **for** t in $1, \ldots, T$ **do**

3: Compute weighted error:

$$\epsilon_t(h) = \sum_{i=1}^{n} w_{t,i} 1(h(x_i) \neq y_i) \quad \forall h \in H.$$

$$\theta_j, s_j = \arg\min_{\theta, s} \epsilon_t(h).$$

$$h_j(x) = \begin{cases} -s_j & \text{if } f_j < \theta_j, \\ s_j & \text{otherwise.} \end{cases}$$

4: Choose weak learner:

$$h_t = \arg\min_{h \in H} \epsilon_t(h).$$

5: Assign voting weight:

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)}.$$

6: Update weights:

$$w_{i,t+1} = w_{i,t} \exp\left(-y_i \alpha_t h_t(x_i)\right) \quad \forall i \in 1, \ldots, n.$$

7: Renormalize weights:

$$w_{i,t+1} = \frac{w_{i,t+1}}{\sum_{i=1}^{n} w_{i,t+1}}.$$

Note, weights keep changing.

8: **end for**

Computation

Strategy 1: C

- Implement AdaBoost loop and weak filters in **C and MEX**.

- Develop in Xcode, VS, … then run in MATLAB.

Strategy 2: parpool()

```
delete(gcp('nocreate'));
parpool(8); % number of CPU-cores

parfor j = 1:n
    [theta(j),…,error(j)] = weak(f(:,j),w,…);
end
```

Strategy 2: parpool()

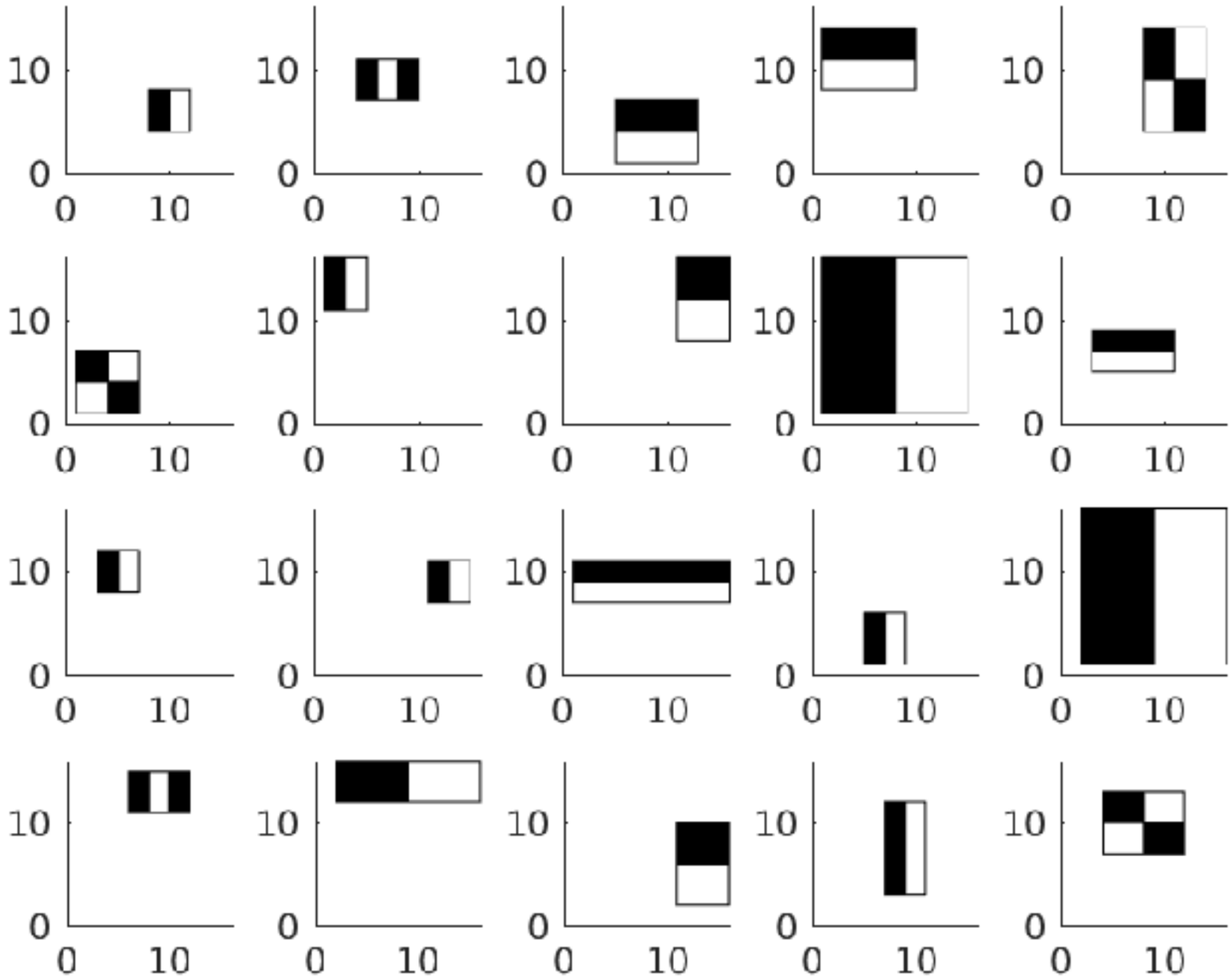
- N_pos = 11838, N_neg = 25356
- 9168 filters
- 1 CPU-core: 270s per iteration -> **8 hours**
- 8 CPU-cores: 35s per iteration -> **1 hour**
- 64 CPU-cores: Amazon **EC2** ‘m4.16xlarge’

Strategy 3: Otherwise.

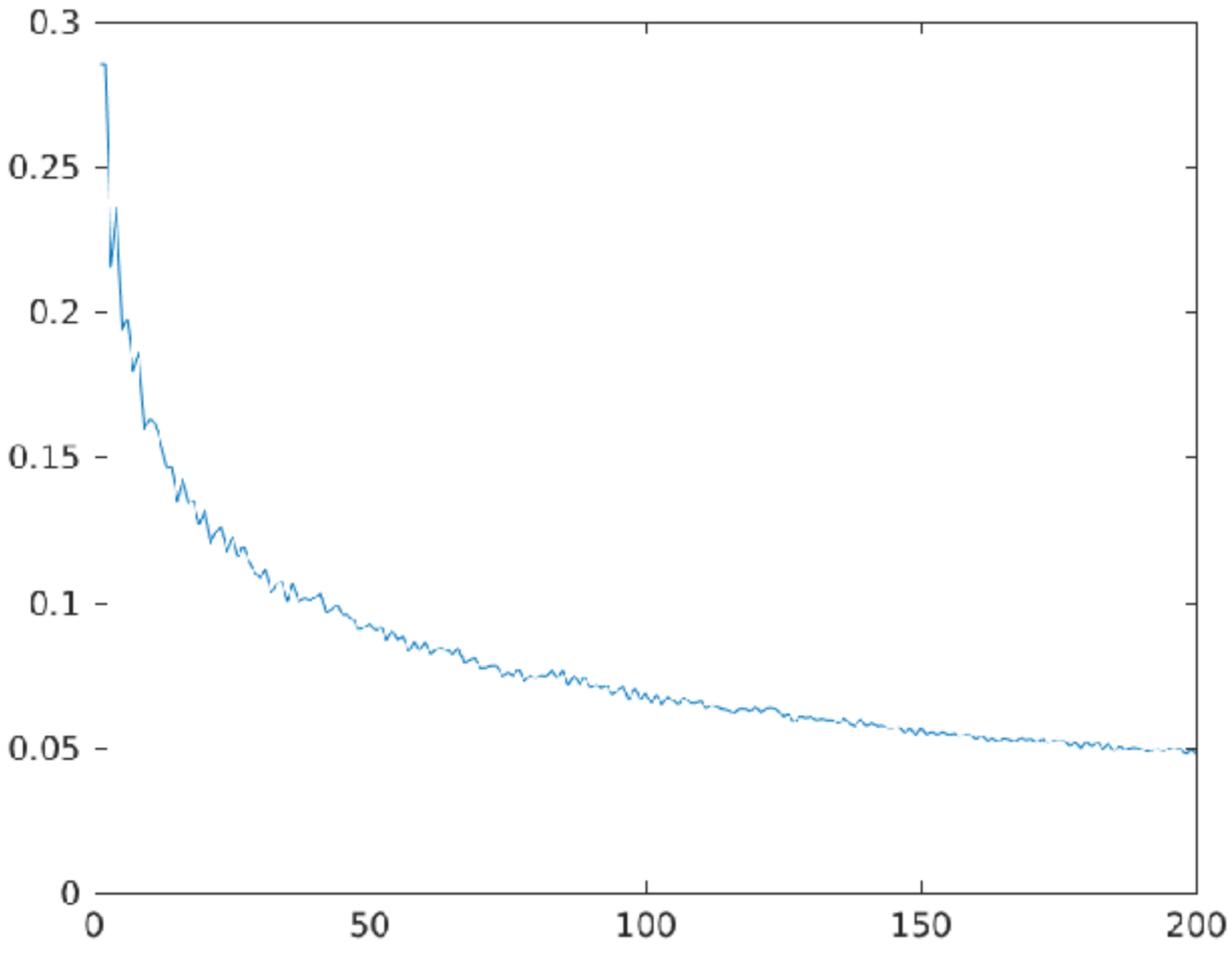
- Start very early.
- Reduce number of filters.
- Reduce number of images.

Report

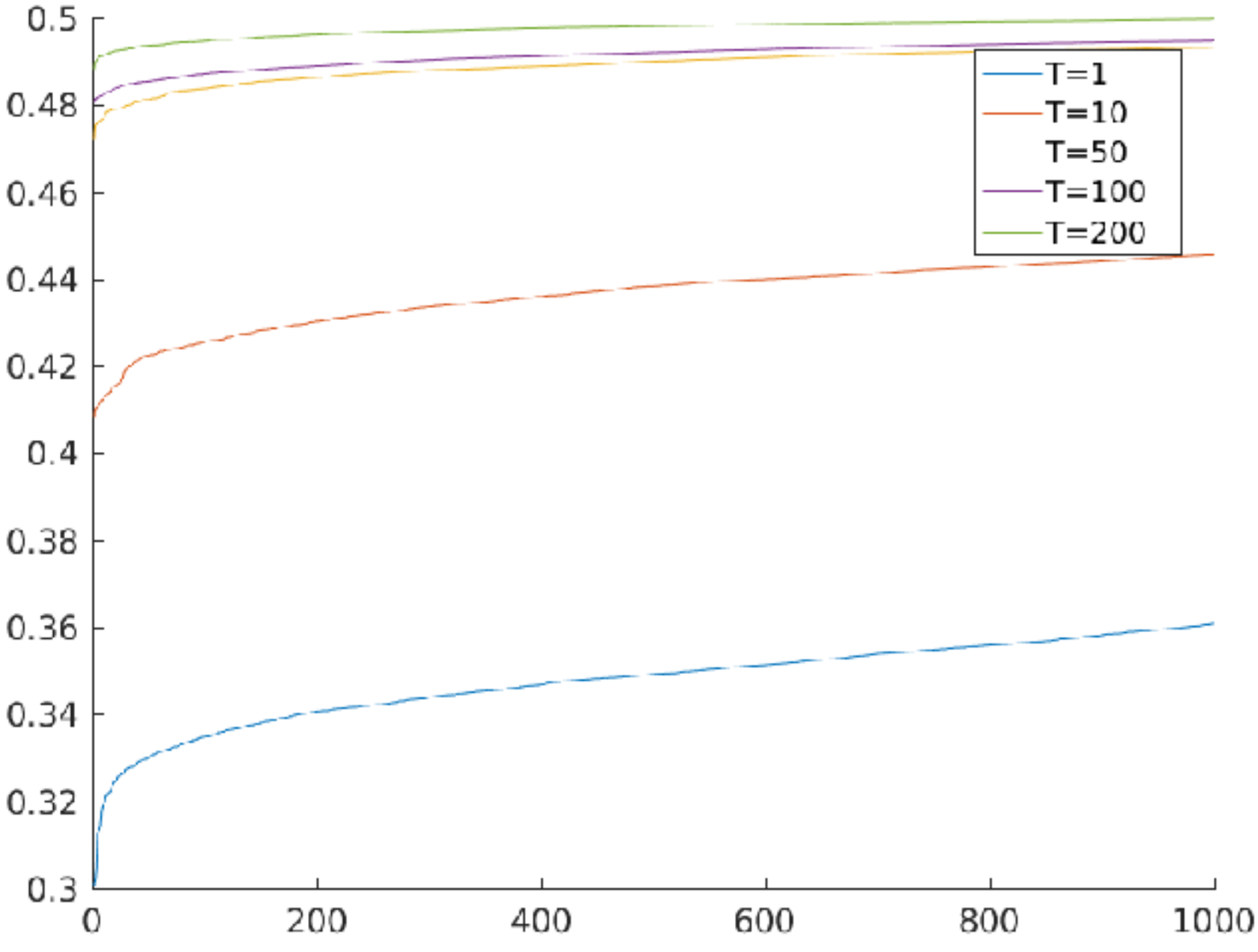
(a) Haar filters



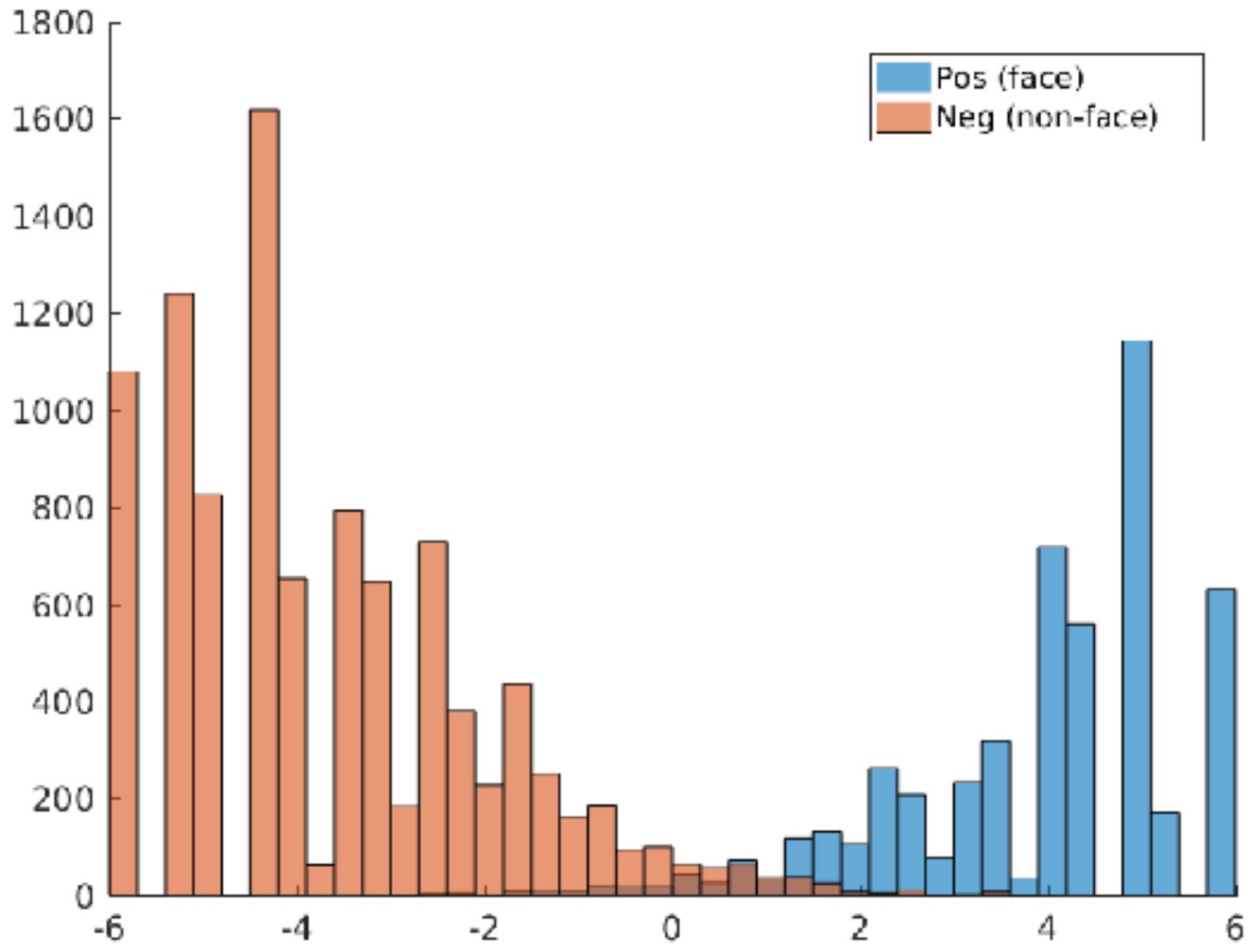
(b) Training error



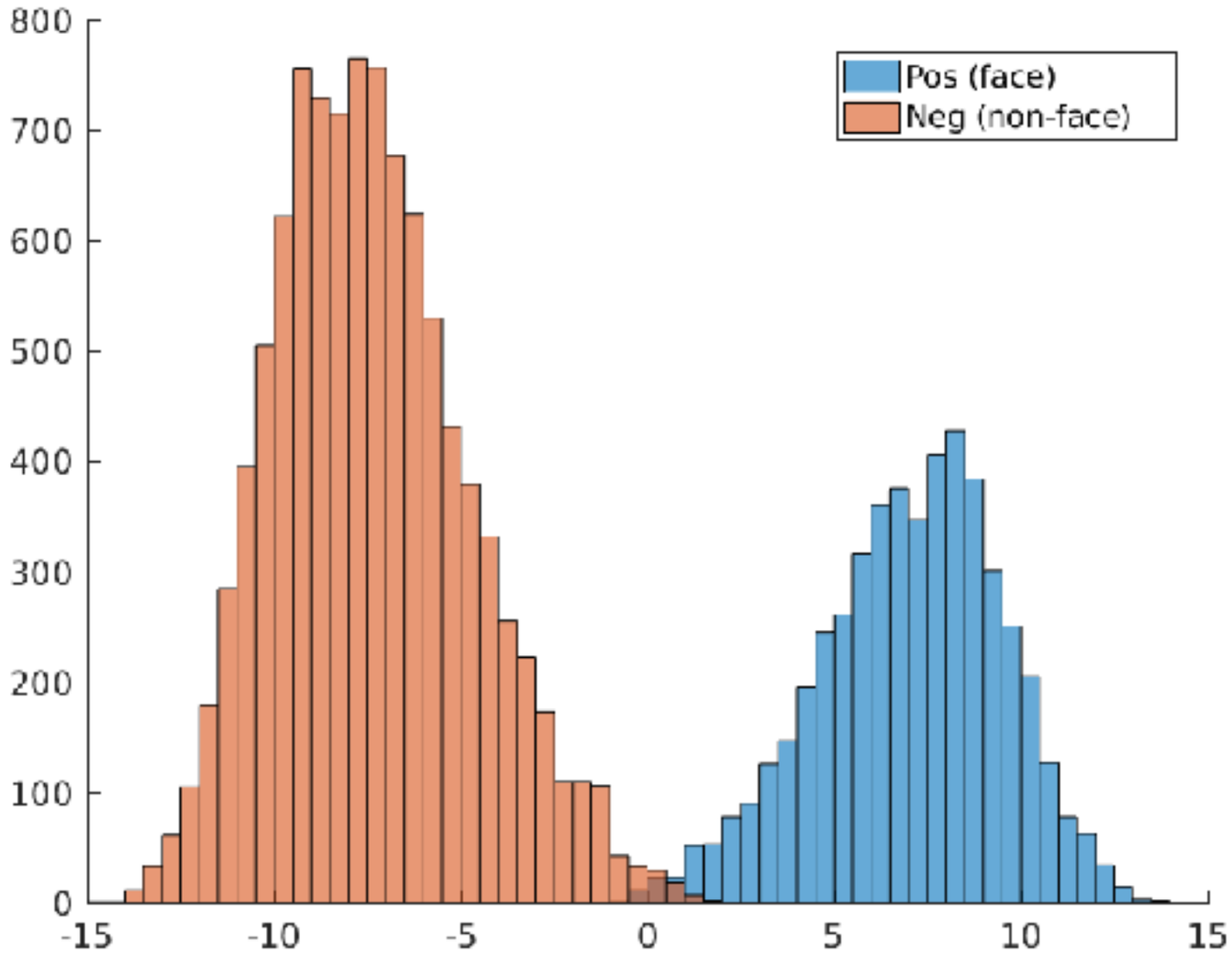
(c) Weak error


0.5
0.48
0.46
0.44
0.42
0.4
0.38
0.36
0.34
0.32
0.3
0
200
400
600
800
1000
T=1
T=10
T=50
T=100
T=200

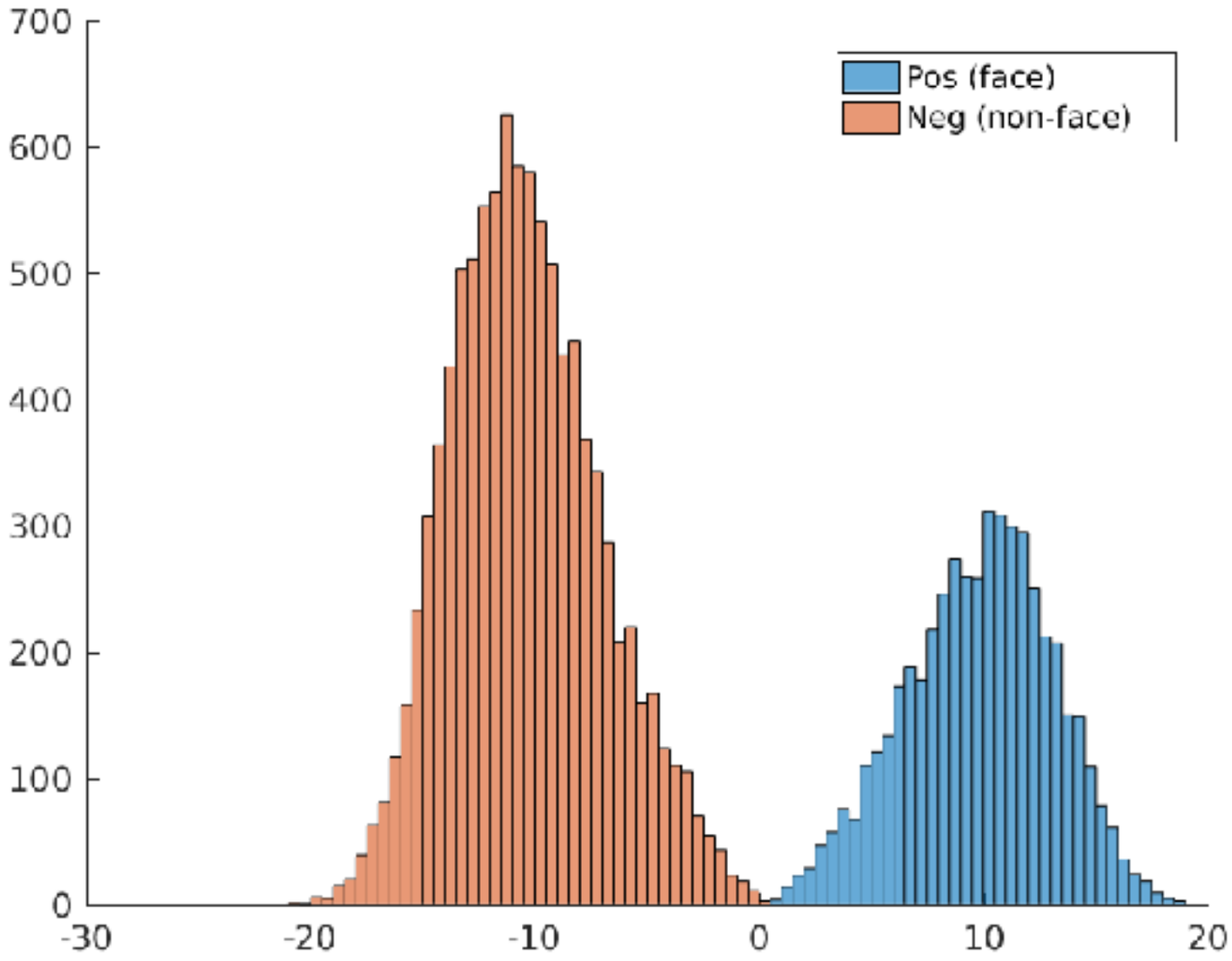
(d) Histogram T=10



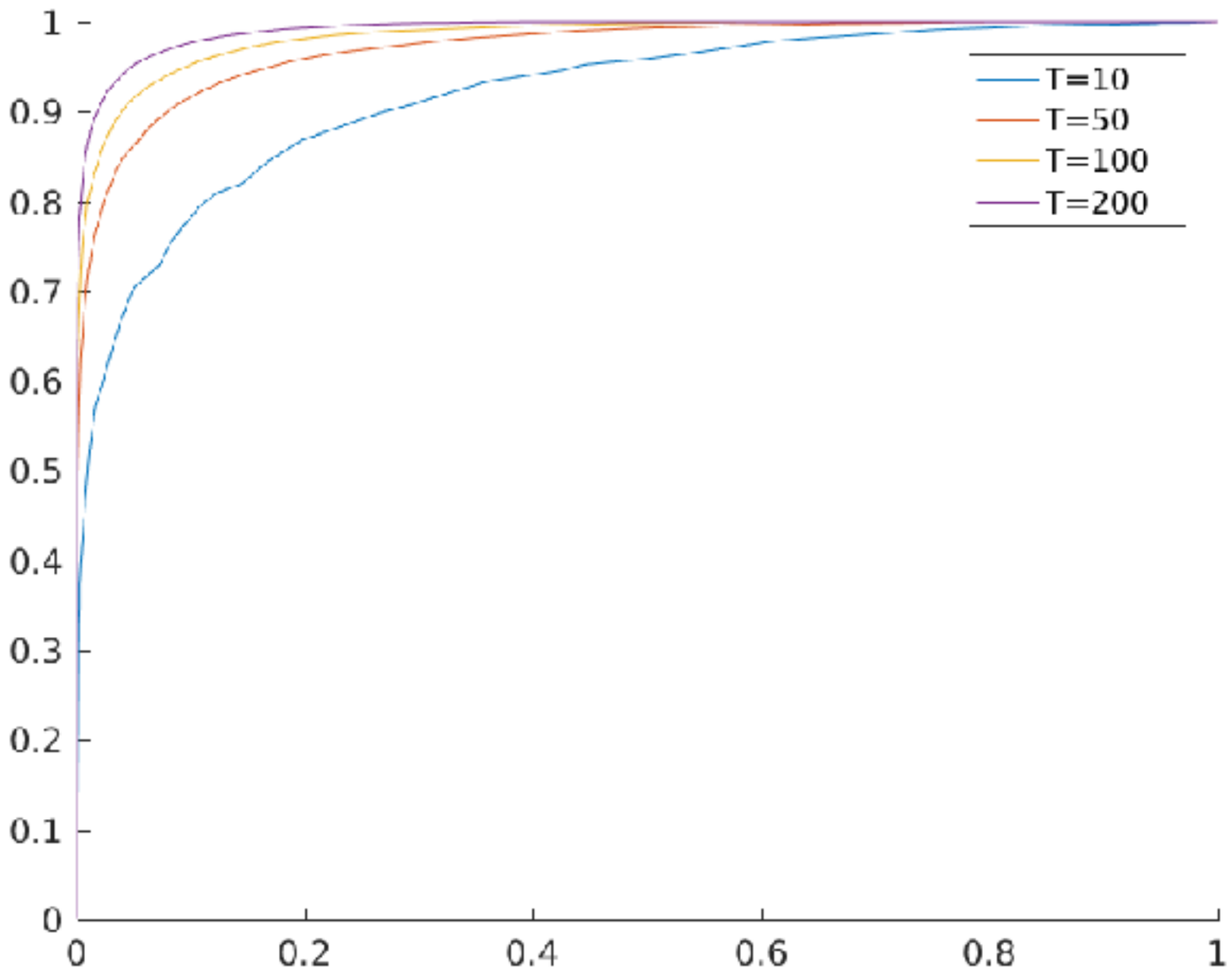
(d) Histogram T=50



(d) Histogram T=100



(e) ROC



(f) Detection