# Ensemble Learning Models

Yanyan Lan

lanyanyan@ict.ac.cn

# What is Ensemble Learning?

# A spam email filter

- Design an email filter that can distinguish spam from non-spam.
- How can we approach this problem with ML?
  1. Gathering as many example as possible of both spam and non-spam emails.
  2. Train a classifier using these examples and their labels.
  3. Take the learned classifier, or prediction rule, and use it to filter your mail.
  4. The goal is to train a classifier that makes the most accurate predictions possible on new test examples.
- But, building a highly accurate classifier is a difficult task. (You still get spam, right?!)


# A spam email filter

We could probably come up with many quick rules of thumb. These could be only moderately accurate. Can you think of an example for this situation?

- An example could be “if the subject line contains ‘buy now’ then classify as spam.”

- This certainly doesn’t cover all spams, but it will be significantly better than random guess.

**Weak Learner:**

The learner performance is slightly higher than random guess. For example, 50% accuracy in binary classification task.

# Ensemble Learning

**Weak Learner:** label examples better than random guess.

**Strong Learner:** arbitrarily well-correlated with the true classification.

- Can a set of weak learners create a single strong learner (Kearns and Valiant 1988, 1989)?

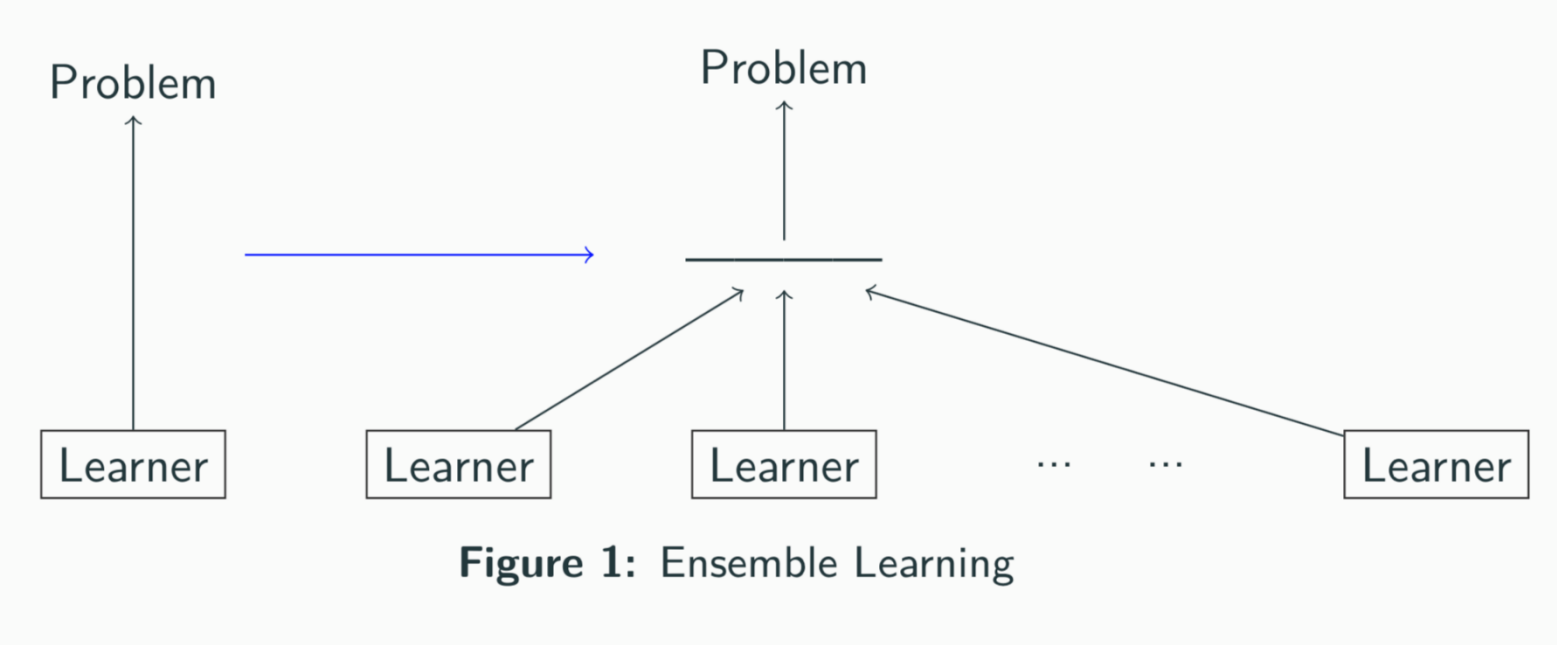
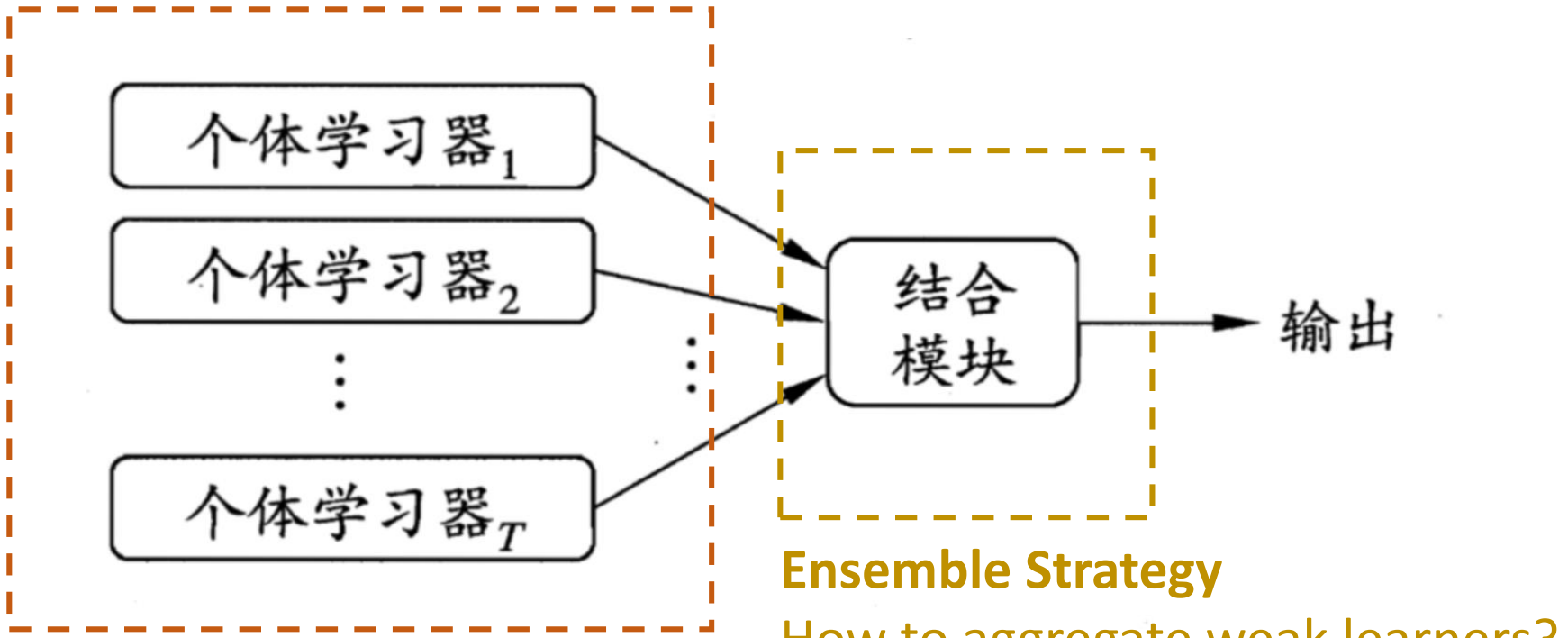



**Figure 1:** Ensemble Learning

# Ensemble Learning




**Ensemble Method**

How to get weak learners?

**Ensemble Strategy**

How to aggregate weak learners?

# Is Ensemble Learning Better?

# Is Ensemble Learning Better?

- Kearns and Valiant proposed a question "Whether the notions of **strong** and **weak** learnability are equivalent"

- Robert Schapire gave an affirmative answer in a 1990 paper to the question of [Kearns and Valiant 1989].

Robert Schapire

# Is Ensemble Learning Better?

- Many experiments show that Ensemble Learning is much better. For example, Hansen and Salamon, TPAMI90.

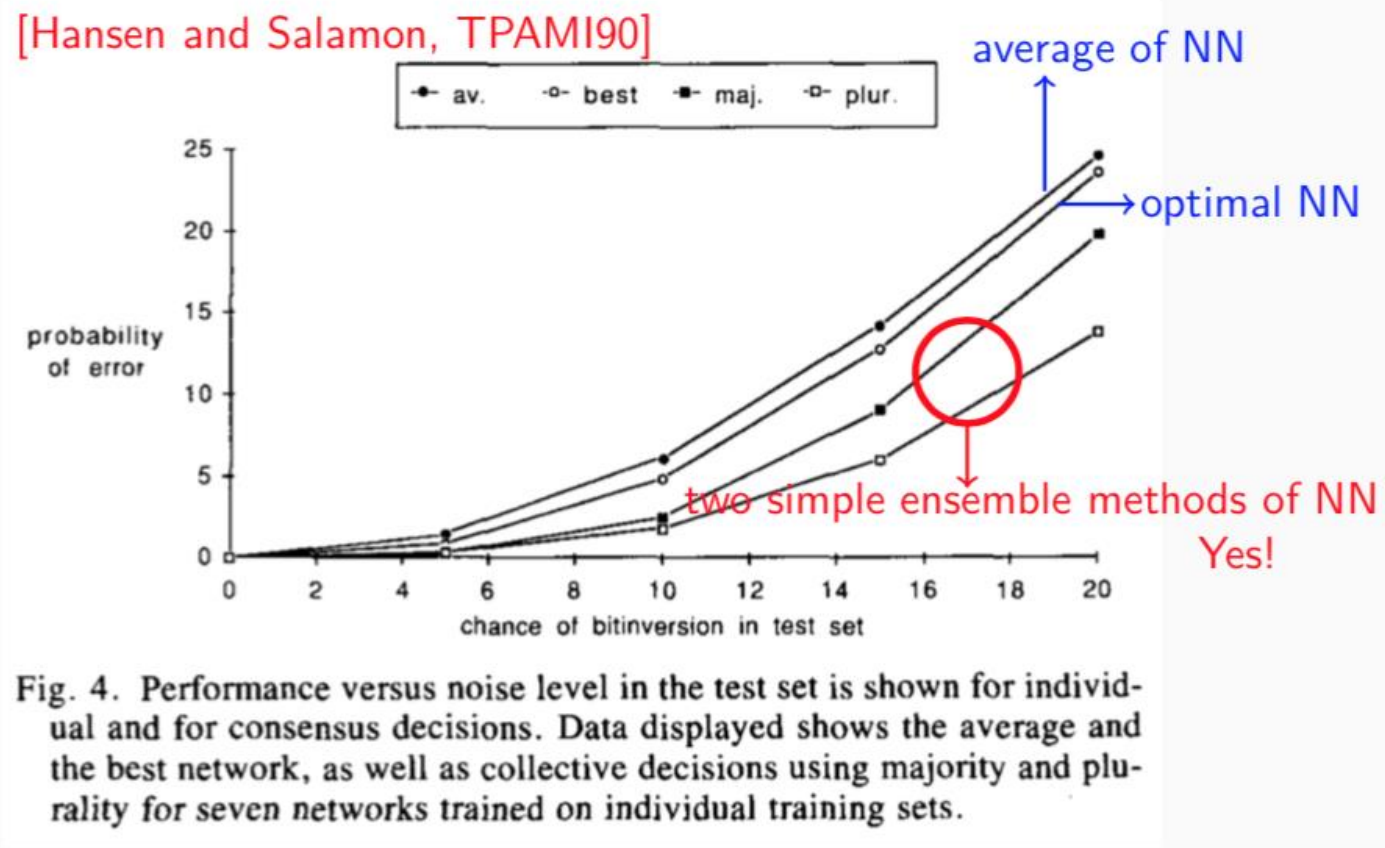



Fig. 4. Performance versus noise level in the test set is shown for individual and for consensus decisions. Data displayed shows the average and the best network, as well as collective decisions using majority and plurality for seven networks trained on individual training sets.

**Figure 2:** Ensemble vs. Single of the Best

# What single learners are needed?

# What single learners are needed?

**Table 1:** single learners vs. Ensemble learner

| Learner | test1 | test2 | test3 |
|---|---|---|---|
| h1 | √ | √ | × |
| h2 | × | √ | √ |
| h3 | √ | × | √ |
| Ensemble | √ | √ | √ |

(a) better

| Learner | test1 | test2 | test3 |
|---|---|---|---|
| h1 | √ | √ | × |
| h2 | √ | √ | × |
| h3 | √ | √ | × |
| Ensemble | √ | √ | × |

(b) no sense

| Learner | test1 | test2 | test3 |
|---|---|---|---|
| h1 | √ | × | × |
| h2 | × | √ | × |
| h3 | × | × | √ |
| Ensemble | × | × | × |

(c) worse

- Single learners should be different and better than random.
- Is it right, theoretically?

# What single learners are needed?

- Label $y \in \{1, -1\}$
- True classifier: $f$
- T base classifiers: independent and error of each one is $\epsilon$, i.e.,

$$P\big(h_i(x) \neq f(x)\big) = \epsilon.$$

- Ensemble classifier:

$$H(x) = sign\Big(\sum_{i=1}^{T} h_i(x)\Big).$$

# What single learners are needed?

- Error of Ensemble Learner:

$$P\big(H(x) \neq f(x)\big) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k}$$

$$\leq \exp\left(-\frac{1}{2} T (1-2\epsilon)^2\right)$$

  - Error:

$$\epsilon \downarrow, \mathrm{P}(\mathrm{H}(\mathrm{x}) \neq f(\mathrm{x})) \downarrow$$

  - Number:

$$\mathrm{T} \uparrow, \mathrm{P}(\mathrm{H}(\mathrm{x}) \neq f(\mathrm{x})) \downarrow$$

- good($\epsilon \downarrow$) v.s. independence

# Boosting

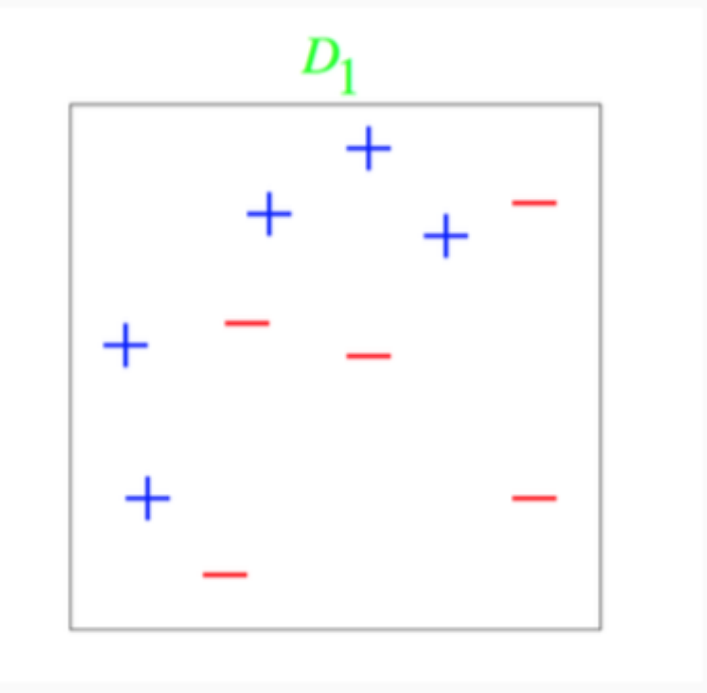
# Boosting

- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote
- On each iteration t:
  - Weight each training example by how incorrectly it was classified
  - Learn a weak hypothesis $h_t$
  - A strength for this hypothesis $\alpha_t$
- Final classifier:

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

# A Toy Example

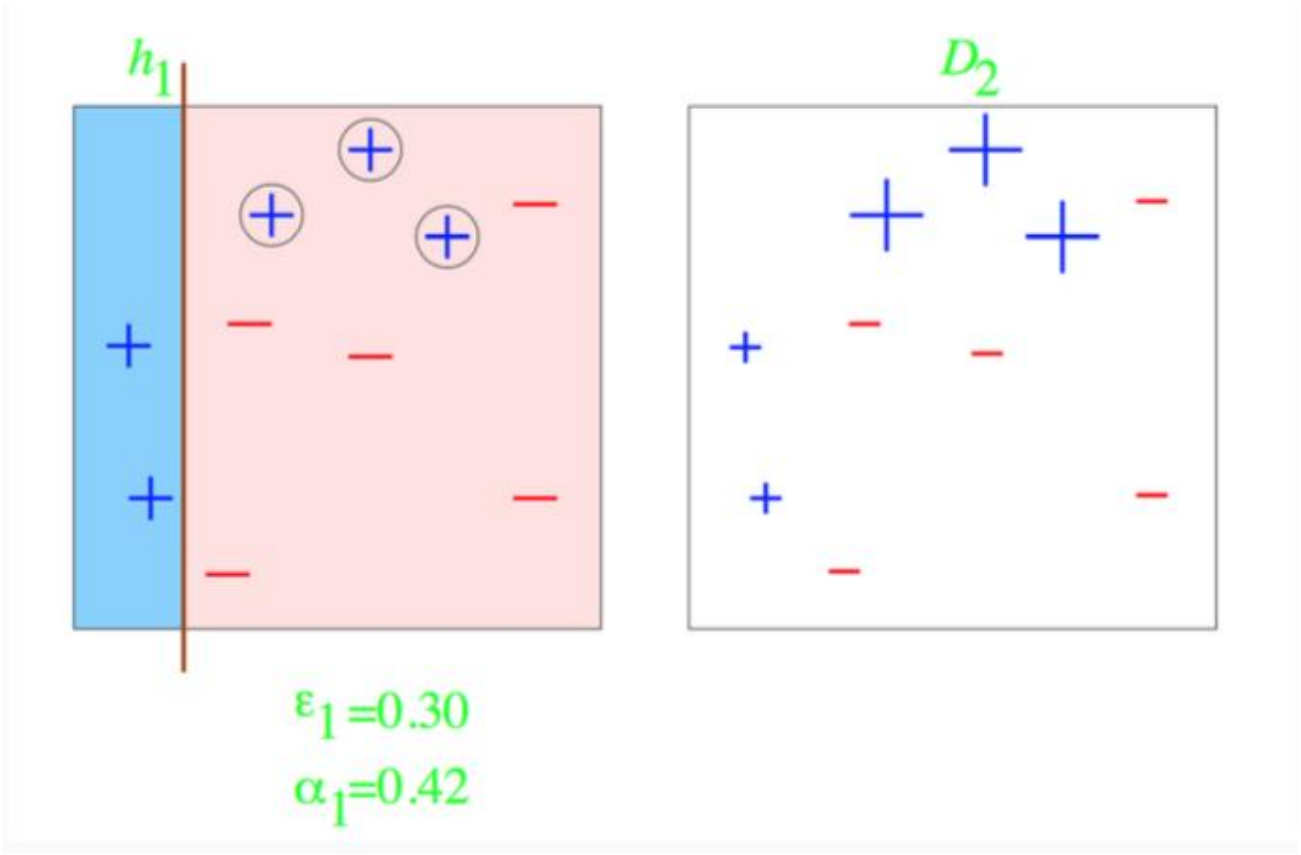
- Toy Example:




- weak classifiers: vertical or horizontal half-planes

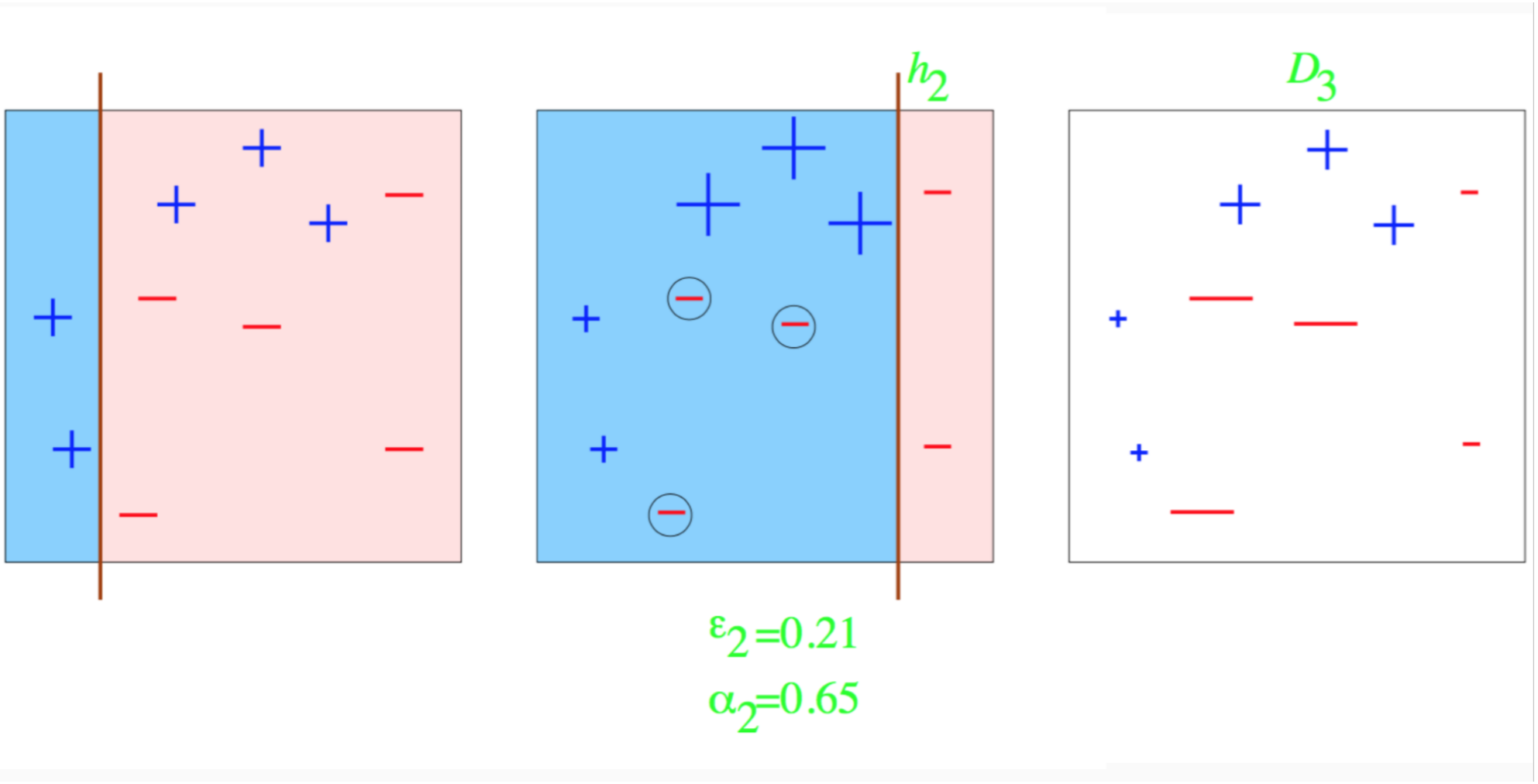
# A Toy Example

- Round 1:



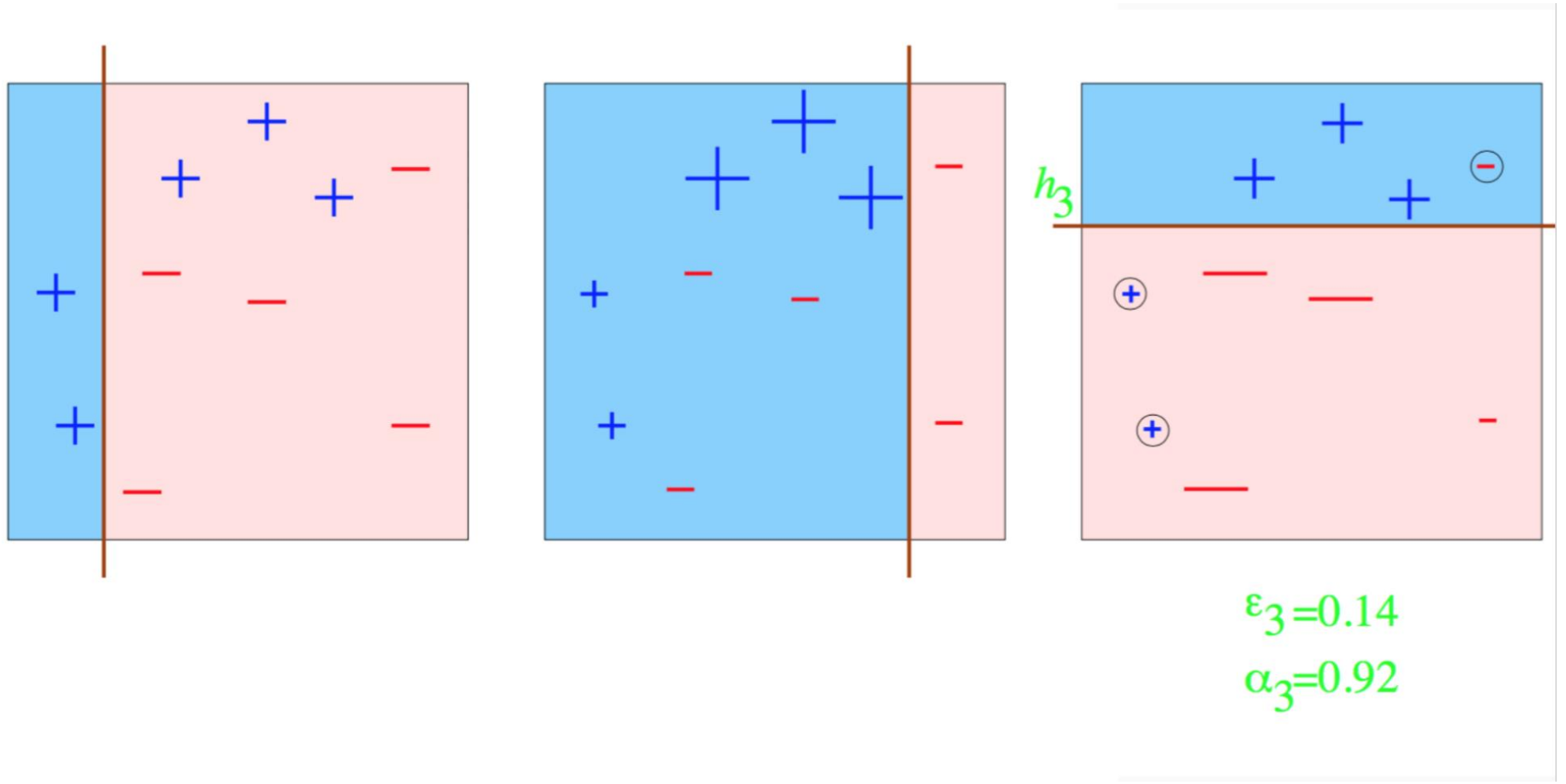
# A Toy Example

- Round 2:



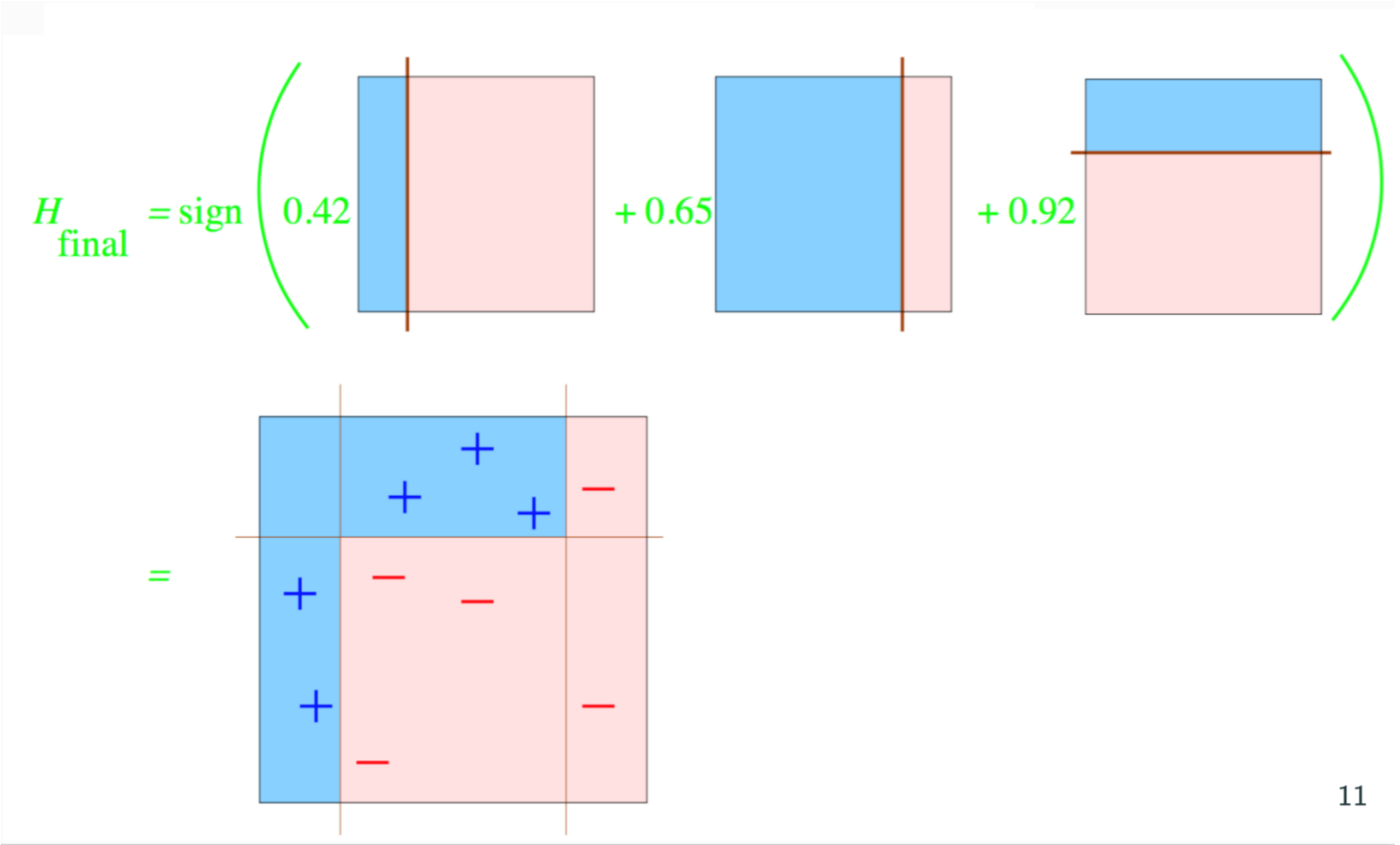
# A Toy Example

- Round 3:



# A Toy Example

- Final Classifier:



# Many Variations

- AdaBoost.M1, AdaBoost.MR,

  FilterBoost, GentleBoost,

  **GradientBoost**, MadaBoost,

  LogitBoost, LPBoost,

  MultiBoost, RealBoost,

  RobustBoost, …

# Key Questions in Boosting

How should the distribution be chosen each round?

How should the weak rules be combined into a single rule?

How many weak classifiers should we learn?

What is the criterion that we are optimizing?

# AdaBoost

- AdaBoost[Freund and Schapire, 1997] is a typical Boosting algorithm.
- Input:
  - $N$ examples $\{(x^1, y^1), \ldots, (x^N, y^N)\}$
  - A weak base learner $h = h(x; \theta)$
- Final classifier is the sign of

$$H(x) = \alpha_1 h(x; \theta) + \cdots + \alpha_m h(x; \theta_m)$$

- Exponential Loss:

$$\ell_{exp}(H) = \sum_{i=1}^{N} \exp(-y^i H(x^i))$$

- How to weak learner weight $\alpha_t$?

# AdaBoost

- AdaBoost[Freund and Schapire, 1997] is a typical Boosting algorithm.
- Input:
  - $N$ examples $\{(x^1, y^1), \ldots, (x^N, y^N)\}$
  - A weak base learner $h = h(x; \theta)$
- Final classifier is the sign of

$$H(x) = \alpha_1 h(x; \theta) + \cdots + \alpha_m h(x; \theta_m)$$

- Exponential Loss:

$$\ell_{exp}(H) = \sum_{i=1}^{N} \exp(-y^i H(x^i))$$

- How to weak learner weight $\alpha_t$?

  minimize the exponential loss $\ell_{exp}(H)$

# AdaBoost

- AdaBoost[Freund and Schapire, 1997] is a typical Boosting algorithm.
- Input:
  - $N$ examples $\{(x^1, y^1), \ldots, (x^N, y^N)\}$
  - A weak base learner $h = h(x; \theta)$
- Final classifier is the sign of

$$H(x) = \alpha_1 h(x; \theta) + \cdots + \alpha_m h(x; \theta_m)$$

- Exponential Loss:

$$\ell_{exp}(H) = \sum_{i=1}^{N} \exp(-y^i H(x^i))$$

- How to weak learner weight $\alpha_t$?

  minimize the exponential loss $\ell_{exp}(H)$

- How to update the distribution in each round?

# AdaBoost

- AdaBoost[Freund and Schapire, 1997] is a typical Boosting algorithm.
- Input:
  - $N$ examples $\{(x^1, y^1), \ldots, (x^N, y^N)\}$
  - A weak base learner $h = h(x; \theta)$
- Final classifier is the sign of

$$H(x) = \alpha_1 h(x; \theta) + \cdots + \alpha_m h(x; \theta_m)$$

- Exponential Loss:

$$\ell_{exp}(H) = \sum_{i=1}^{N} \exp(-y^i H(x^i))$$

- How to weak learner weight $\alpha_t$?

  minimize the exponential loss $\ell_{exp}(H)$
- How to update the distribution in each round?

  derived from exponential loss

# Weight of Weak Learner: $\alpha_t$

1. $\frac{\partial \ell_{exp}(H|D)}{\partial H(x)}$:

$$\frac{\partial \ell_{exp}(H|D)}{\partial H(x)} = -e^{-H(x)}P(f(x)=1|x) + e^{H(x)}P(f(x)=-1|x) = 0$$

2. $H(x)$:

$$H(x) = \frac{1}{2} ln \frac{P(f(x)=1|x)}{P(f(x)=-1|x)}$$

3. $sign(H(x))$:

$$sign\big(H(x)\big) = sign\left(\frac{1}{2} ln \frac{P(f(x)=1|x)}{P(f(x)=-1|x)}\right)$$

$$= \begin{cases} 1, & if\ P(f(x)=1|x) > P(f(x)=-1|x) \\ -1, & if\ P(f(x)=1|x) < P(f(x)=-1|x) \end{cases}$$

$$= \arg \max_{y \in \{-1,1\}} P(f(x)=y|x)$$

# Weight of Weak Learner: $\alpha_t$

- $\alpha_t$ minimizes $\ell_{exp}(\alpha_t h_t|\mathcal{D}_t)$
- $\ell_{exp}(\alpha_t h_t|\mathcal{D}_t)$:

$$\begin{aligned}\ell_{exp}(\alpha_t h_t|\mathcal{D}_t) &= \mathbb{E}_{x\sim\mathcal{D}_t}\left[e^{-f(x)\alpha_t h_t(x)}\right] \\ &= \mathbb{E}_{x\sim\mathcal{D}_t}\left[e^{-\alpha_t}\mathbb{I}\left(f(x)=h_t(x)\right)+e^{\alpha_t}\mathbb{I}\left(f(x)\neq h_t(x)\right)\right] \\ &= e^{-\alpha_t}P_{x\sim\mathcal{D}_t}\left(f(x)=h_t(x)\right)+e^{\alpha_t}P_{x\sim\mathcal{D}_t}\left(f(x)\neq h_t(x)\right) \\ &= e^{-\alpha_t}(1-\epsilon_t)+e^{\alpha_t}\epsilon_t\end{aligned}$$

where $\epsilon_t = P_{x\sim\mathcal{D}_t}(f(x)\neq h_t(x))$.

- $\frac{\partial \ell_{exp}(\alpha_t h_t|\mathcal{D}_t)}{\partial \alpha_t}$:

$$\frac{\partial \ell_{exp}(\alpha_t h_t|\mathcal{D}_t)}{\partial \alpha_t} = -e^{-\alpha_t}(1-\epsilon_t)+e^{\alpha_t}\epsilon_t = 0$$

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

# Update the distribution:

- $h_{t+1}$ can minimize the loss $\ell_{exp}(H_t + h_{t+1}|\mathcal{D})$
- $\ell_{exp}(H_t + h_{t+1}|\mathcal{D})$:

$$\begin{aligned}
\ell_{exp}(H_t + h_{t+1}|\mathcal{D}) &= \mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)(H_t(x)+h_{t+1}(x))}\right] \\
&= \mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\mathrm{e}^{-f(x)h_{t+1}(x)}\right] \\
&\simeq \mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\left(1 - f(x)h_{t+1}(x) + \frac{f^2(x)h_{t+1}^2(x)}{2}\right)\right] \\
&= \mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}(1 - f(x)h_{t+1}(x) + \frac{1}{2})\right]
\end{aligned}$$

- $h_{t+1}$:

$$\begin{aligned}
h_{t+1} &= \arg\min_h \ell_{exp}(H_t + h|\mathcal{D}) \\
&= \arg\max_h \mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}f(x)h(x)\right] \\
&= \arg\max_h \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{e^{-f(x)H_t(x)}}{\mathbb{E}_{x\sim\mathcal{D}}[e^{-f(x)H_t(x)}]}f(x)h(x)\right]
\end{aligned}$$

- $h_{t+1}$:

$$h_{t+1} = \arg\max_h \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{e^{-f(x)H_t(x)}}{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\right]} f(x)h(x)\right]$$

- Let $\mathcal{D}_{t+1}(x) = \frac{\mathcal{D}(x)e^{-f(x)H_t(x)}}{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\right]}$

$$\mathcal{D}_{t+1}(x) = \frac{\mathcal{D}(x)e^{-f(x)H_t(x)}}{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\right]}$$

$$= \frac{\mathcal{D}(x)e^{-f(x)H_{t-1}(x)}e^{-f(x)\alpha_t h_t(x)}}{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\right]}$$

$$= \frac{\mathcal{D}(x)e^{-f(x)H_{t-1}(x)}e^{-f(x)\alpha_t h_t(x)}\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_{t-1}(x)}\right]}{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_{t-1}(x)}\right]\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\right]}$$

$$= \mathcal{D}_t(x)e^{-f(x)\alpha_t h_t(x)}\frac{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_{t-1}(x)}\right]}{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\right]}$$

where $\frac{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_{t-1}(x)}\right]}{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_t(x)}\right]}$ is the normalization factor.

- Iteration equation:

$$\mathcal{D}_{t+1}(x) \propto \mathcal{D}_t(x)e^{-\alpha_t f(x)h_t(x)}$$

# AdaBoost

**Input**: Trainset $D = (x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)$;

Algorithm to obtain the base learner $\mathcal{L}$;

Turns of training $T$;

**Process**:

1. $\mathcal{D}_1(x) = 1/m$
2. **for** $t = 1,2, \dots, T$ **do**
3. $h_t = \mathcal{L}(\mathrm{D}, \mathcal{D}_t)$;
4. $\epsilon_t = P_{x \sim \mathcal{D}_t}(h_t(x) \neq f(x))$
5. **if** $\epsilon_t > 0.5$ **then Break**
6. $\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \geq 0$
7. $\mathcal{D}_{t+1} = \frac{\mathcal{D}_t}{Z_t} \times \begin{cases} \exp(-\alpha_t) \leq 1, \ if \ h_t(x) = f(x) \\ \exp(\alpha_t) \geq 1, \ if \ h_t(x) \neq f(x) \end{cases} = \frac{\mathcal{D}_t(x) \exp(-\alpha_t f(x) h_t(x))}{Z_t}$
8. **end for**

**Output:** $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

# AdaBoost Example

- 给定如表所示的训练数据
- 假设弱分类器由$x < v$ 或 $x > v$产生，其阈值$v$使该分类器在训练数据集上分类误差率最低
- 试用AdaBoost算法学习一个强分类器

| 序号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $y$ | 1 | 1 | 1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 |

# AdaBoost Example

**解**：初始化数据权值分布

$$D_1 = (w_{11}, w_{12}, \ldots, w_{110})$$

$$w_{1i} = 0.1, i = 1,2, \ldots, 10$$

| 序号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $y$ | 1 | 1 | 1 | −1 | −1 | −1 | 1 | 1 | 1 | −1 |

# AdaBoost Example

- 对m =1，

(a) 在权值分布为$D_1$ 的训练数据上，阈值$v$取2.5时分类误差率最低，故基本分类器为

$$G_1(x)=\begin{cases}1, & x<2.5\\ -1, & x>2.5\end{cases}$$

(b) $G_1(x)$ 在训练数据集上的误差率 $e_1=P(G_1(x_i)\neq y_i)=0.3$

(c) 计算$G_1(x)$ 的系数：　　$\alpha_1=\frac{1}{2}\log\frac{1-e_1}{e_1}=0.4236$

(d) 更新训练数据的权值分布：

$$D_2=(w_{21},\cdots,w_{2i},\cdots,w_{210})$$

$$w_{2i}=\frac{w_{1i}}{Z_1}\exp(-\alpha_1 y_i G_1(x_i)),\quad i=1,2,\cdots,10$$

$$D_2=(0.0715,0.0715,0.0715,0.0715,0.0715,0.0715,$$
$$0.1666,0.1666,0.1666,0.0715)$$

$$f_1(x)=0.4236G_1(x)$$

分类器sign[$f_1(x)$]在训练数据集上有3个误分类点.

# AdaBoost Example

- 对m =2，

(a) 在权值分布为$D_2$ 的训练数据上，阈值$v$是8.5时分类误差率最低，基本分类器为

$$G_2(x)=\begin{cases}1, & x<8.5\\ -1, & x>8.5\end{cases}$$

(b) $G_2(x)$ 在训练数据集上的误差率 $e_2=0.2143$

(c) 计算 $\alpha_2=0.6496$

(d) 更新训练数据的权值分布：

$$D_3=(0.0455, 0.0455, 0.0455, 0.1667, 0.1667, 0.1667,$$
$$0.1060, 0.1060, 0.1060, 0.0455)$$
$$f_2(x)=0.4236G_1(x)+0.6496G_2(x)$$

分类器sign[$f_2(x)$]在训练数据集上有3个误分类点.

# AdaBoost Example

- 对m =3，

(a) 在权值分布为$D_3$ 的训练数据上，阈值$v$是5.5时分类误差率最低，基本分类器为

$$G_3(x)=\begin{cases}1, & x>5.5\\ -1, & x<5.5\end{cases}$$

(b) $G_3(x)$ 在训练数据集上的误差率 $e_3=0.1820$

(c) 计算 $\alpha_3=0.7514$

(d) 更新训练数据的权值分布：

$$D_4=(0.125,0.125,0.125,0.102,0.102,0.102,0.065,0.065,0.065,0.125)$$

于是得到：

$$f_3(x)=0.4236G_1(x)+0.6496G_2(x)+0.7514G_3(x)$$

分类器sign[$f_3(x)$]在训练数据集上误分类点个数为0.

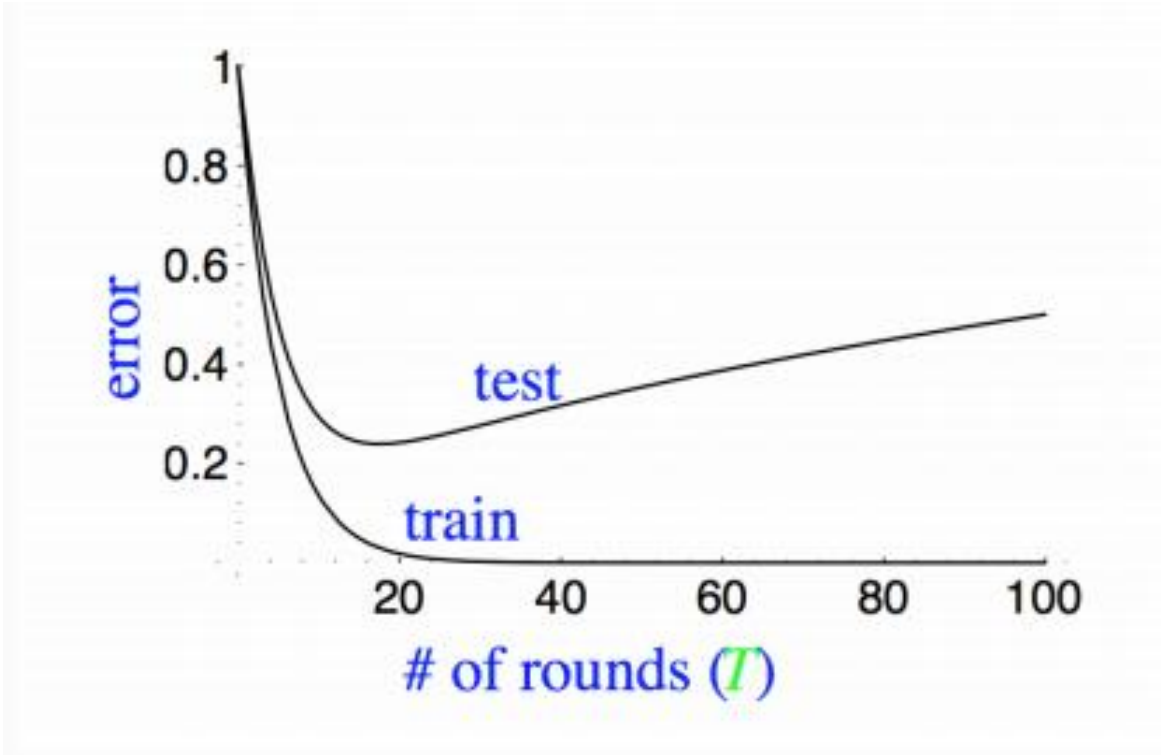
于是最终分类器为

$$G(x)=\text{sign}[f_3(x)]=\text{sign}[0.4236G_1(x)+0.6496G_2(x)+0.7514G_3(x)]$$

# How Will Test Error Behave?

# Test Error

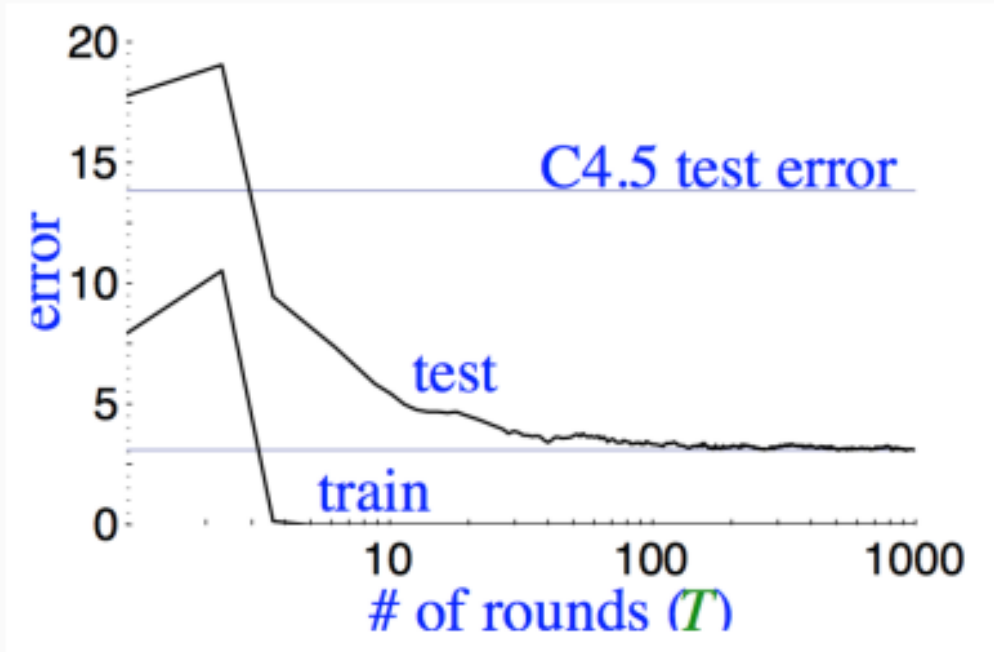
- Expect (a first guess)
  - Training error to continue to drop(or reach 0)
  - Test error increases when $H_{final}$ becomes “too complex”
    - Occam’s razor
    - Overfitting: hard to know when to stop training




Modified from the tutorial of Freund & Schapire.

# Test Error

- Actually
  - 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 |

**Occam's razor wrongly predicts "simpler" rule is better?**

Modified from the tutorial of Freund & Schapire.

# Test Error - Margin Theory Explanation

- Margin Theory Explanation

Based on the concept of margin, Schapire et al. [1998] proved that, given any threshold $\theta > 0$ of margin over the training data $D$, with probability at least $1-\delta$, the generalization error of the ensemble $\epsilon_{\mathcal{D}} = P_{\boldsymbol{x}\sim\mathcal{D}}(f(\boldsymbol{x}) \neq H(\boldsymbol{x}))$ is bounded by

$$\epsilon_{\mathcal{D}} \leq P_{\boldsymbol{x}\sim D}(f(\boldsymbol{x})H(\boldsymbol{x}) \leq \theta) + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2} + \ln\frac{1}{\delta}}\right)$$

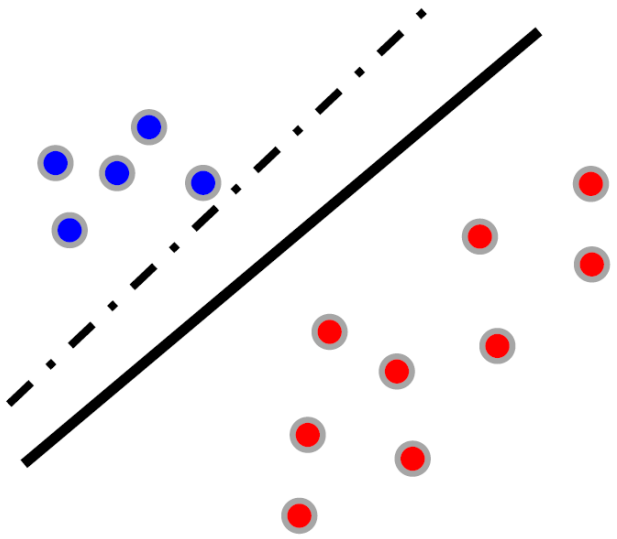
$$\leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}} + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2} + \ln\frac{1}{\delta}}\right)$$

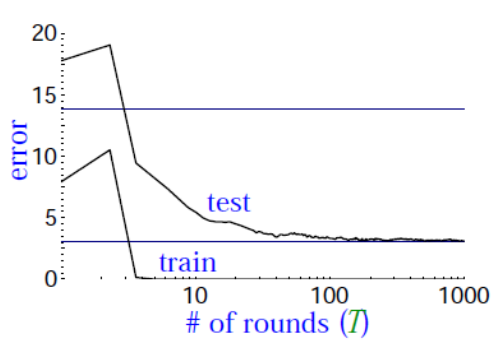
- This bound implies that, when other variables are fixed, **the larger the margin over the training data, the smaller the generalization error**!

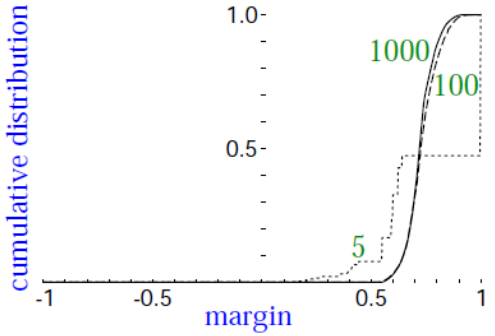
Modified from the tutorial of Freund & Schapire.

# Test Error - Margin Theory Explanation

- Why AdaBoost tends to be resistant to overfitting? the margin theory answers:
  - It can increase the **ensemble margin** even after the training error reaches zero!






| | # rounds | | |
|---|---|---|---|
| | 5 | 100 | 1000 |
| train error | 0.0 | 0.0 | 0.0 |
| test error | 8.4 | 3.3 | 3.1 |
| % margins $\leq 0.5$ | 7.7 | 0.0 | 0.0 |
| minimum margin | 0.14 | 0.52 | 0.55 |

Modified from the tutorial of Freund & Schapire.

# Practical Advantages of AdaBoost

- Very fast
- Simple and easy to program
- Few parameters to tune (except T)
- Flexible — can combine with (m)any learning algorithm
- Little prior knowledge needed about weak learner
- Provably effective, provided can consistently find rough rules of thumb
  - → shift in mind set — goal now is merely to find classifiers barely better than random guessing
- versatile
  - can use with data that is textual, numeric, discrete, etc.
  - has been extended to learning problems well beyond binary classification

# Disadvantages of AdaBoost

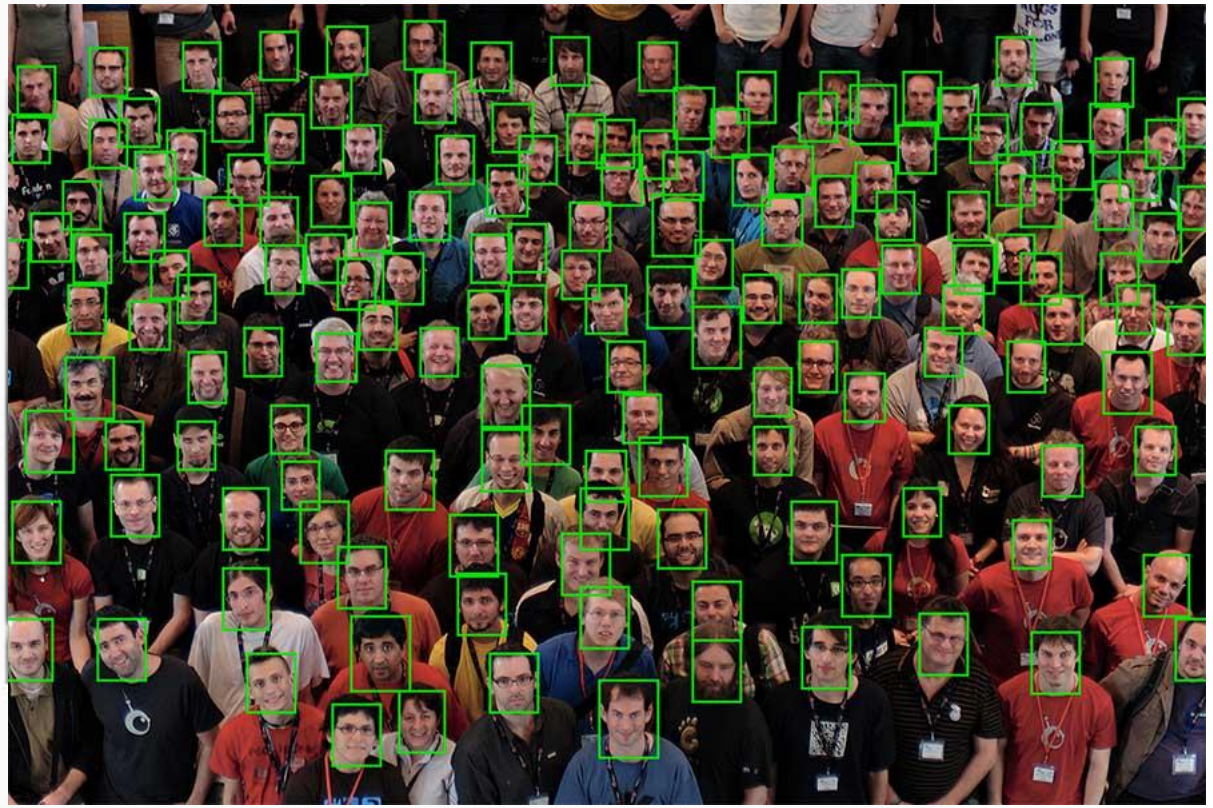
- performance of AdaBoost depends on data and weak learner
- consistent with theory, AdaBoost can fail if
  - weak classifiers too complex
    - overfitting
  - weak classifiers too weak ($\gamma_t \to 0$ too quickly)
    - underfitting
    - low margins → overfitting

empirically, AdaBoost seems especially susceptible to uniform noise

# Application: Face Detection

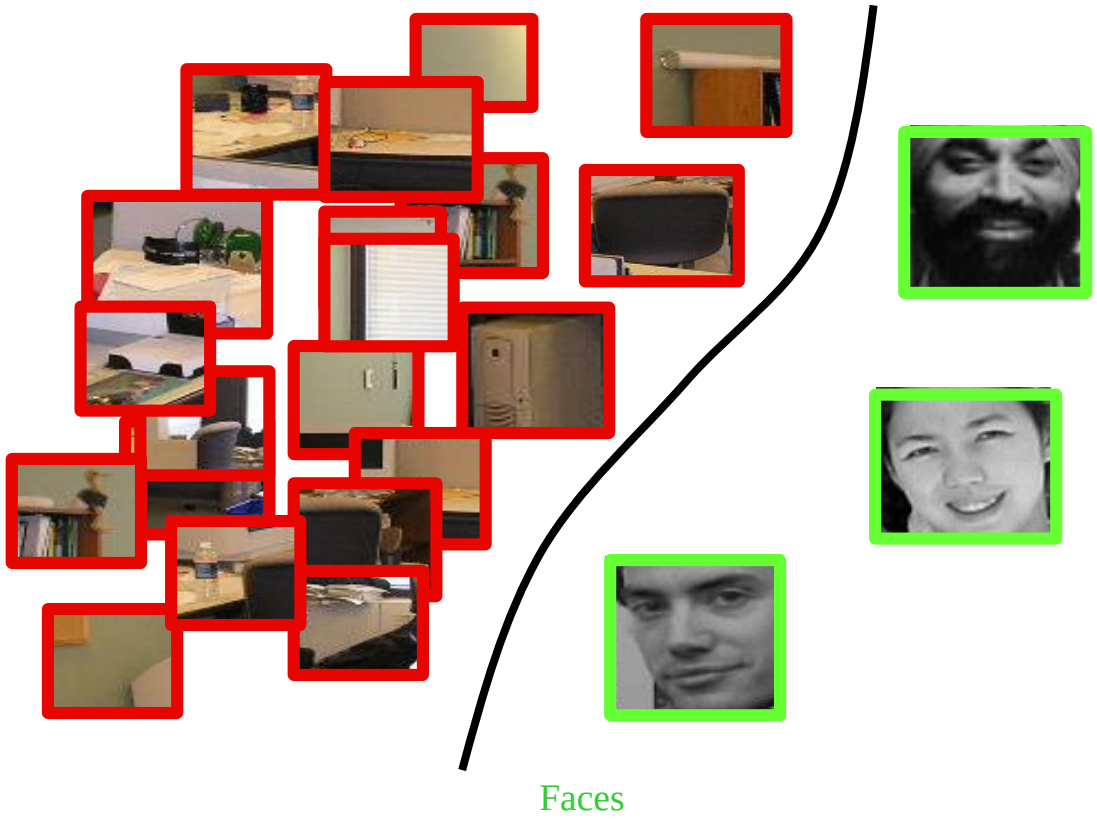

Training set

Adaboost



# Application: Face Detection

- In some feature space




## 1. hypothesize:

try all possible rectangle locations, sizes

## 2. test:

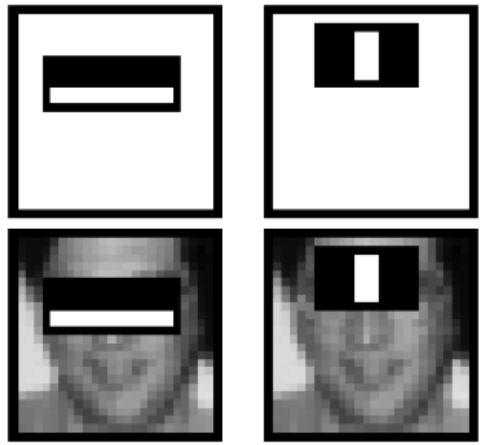
classify if rectangle contains a face (and only the face)

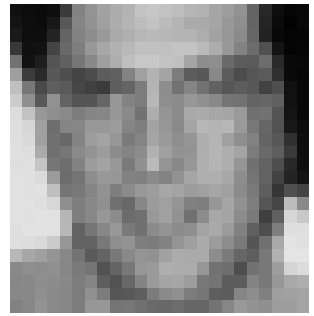
Note: 1000's more false windows then true ones.

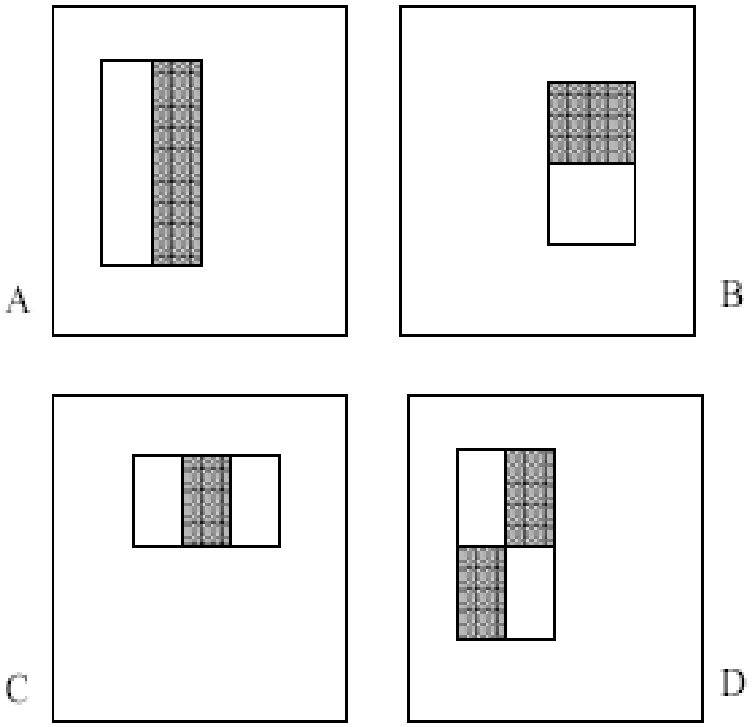
# Application: Face Detection

- Rectangle filters denoted by $f$
- Only 4 Types of “Rectangle filters” (Similar to Haar wavelets Papageorgiou, et al. )

g(x) =
sum(WhiteArea) - sum(BlackArea)







# Application: Face Detection

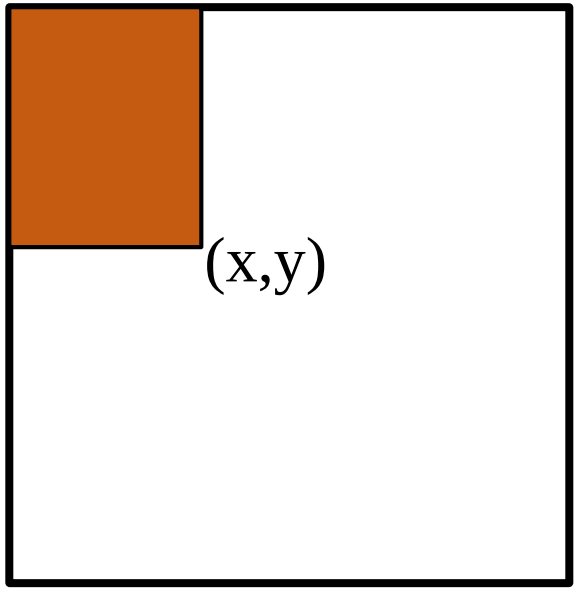
Why rectangle features ?

Too Slow ! $O(n^4)$

# Integral Image

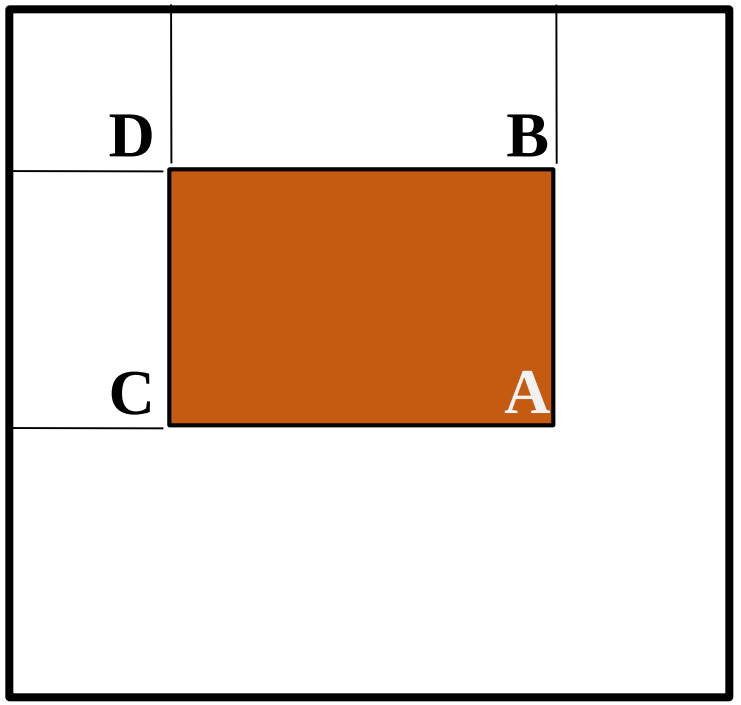
- The *integral image* computes a value at each pixel (*x*,*y*) that is the sum of the pixel values above and to the left of (*x*,*y*), inclusive.

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



# Integral Image

- Let A,B,C,D be the values of the integral image at the corners of a rectangle




- Then the sum of original image values within the rectangle can be computed:

  sum = A – B – C + D

- Only 3 additions are required for any size of rectangle!
  - This is now used in many areas of computer vision

# Application: Face Detection

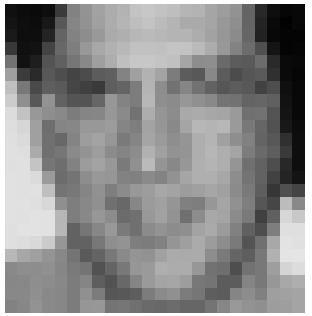
$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

Features vector

Weight

Weak classifier



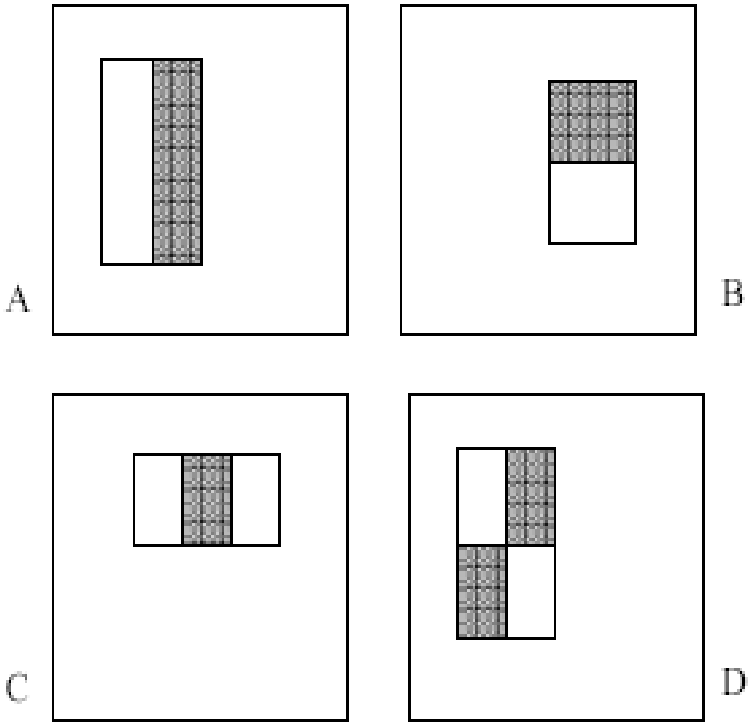
Binary Features: $f_i(x) = \begin{cases} 1 & \text{if } g_i(x) > \theta_i \\ -1 & \text{otherwise} \end{cases}$

Need to:

(1) Select Features i=1..n,

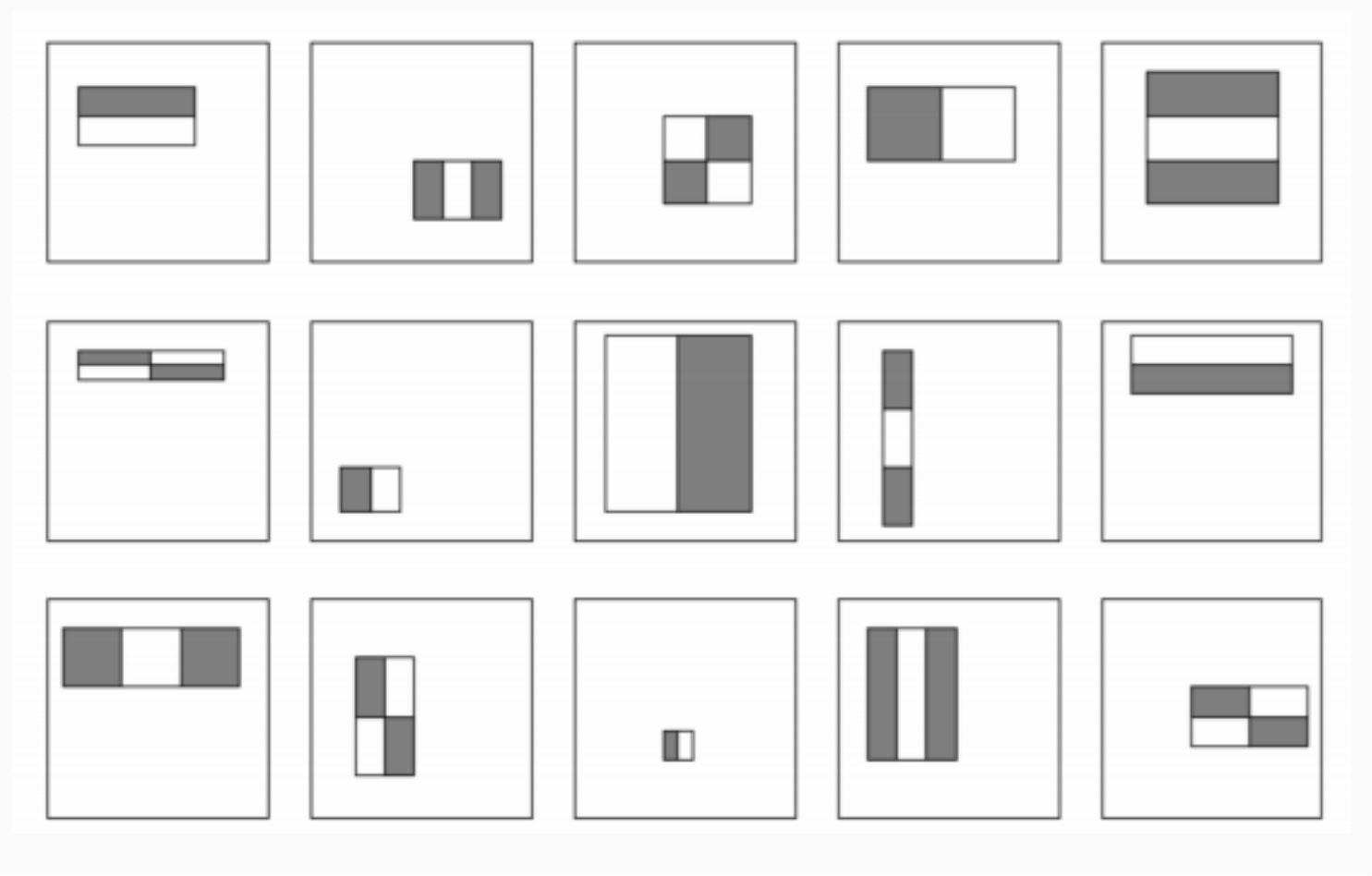
(2) Learn thresholds $\theta_i$,

(3) Learn weights $\alpha_i$



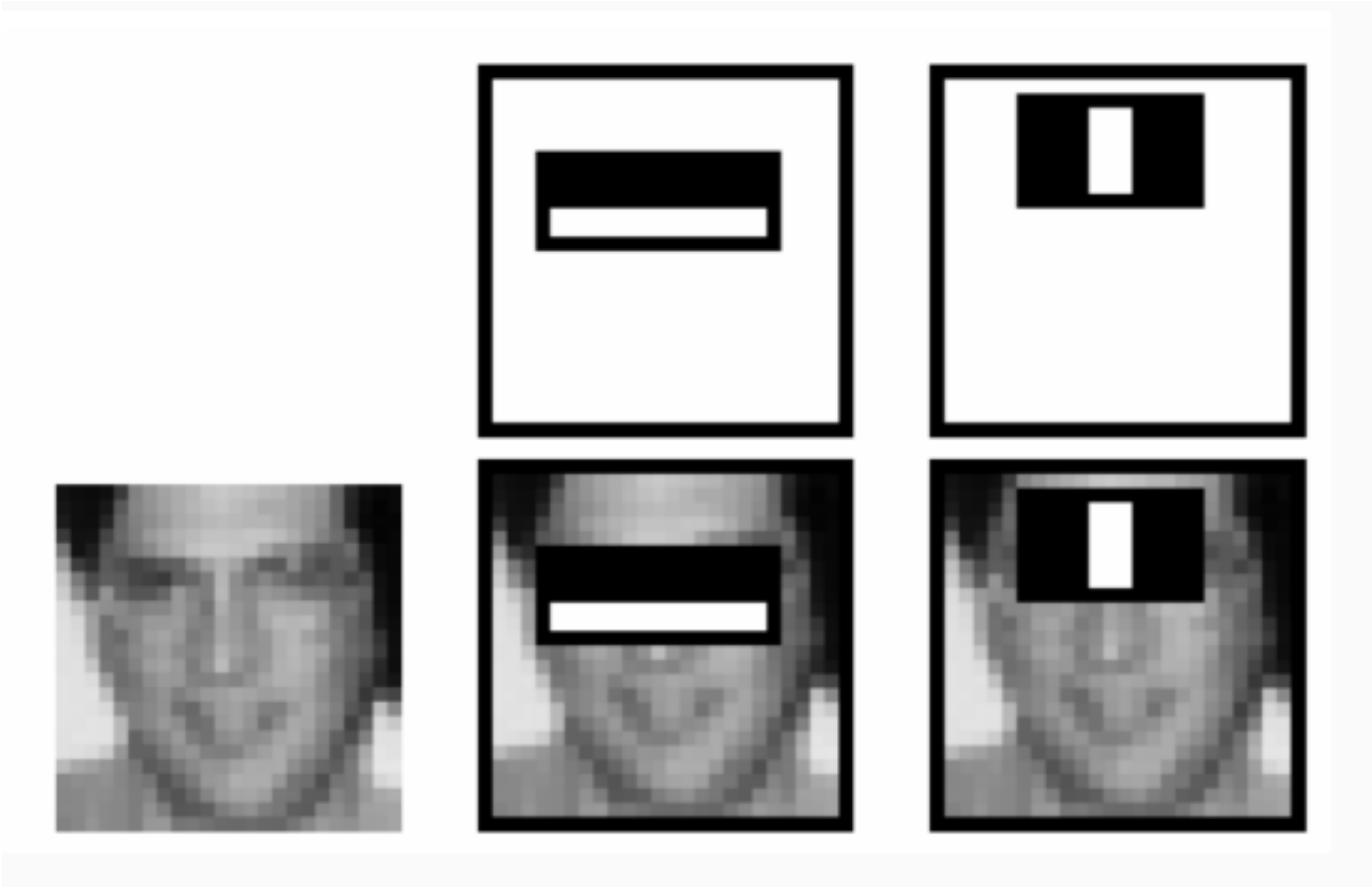
# Feature Selection

The number of possible rectangle features with 24x24 region is 180,000!



# Feature Selection

- Top two features selected by AdaBoost



# Face Detection Performance

- A classifier with 200 rectangle features was learned using AdaBoost
- detection rate of 95% with a false positive rate of 1 in 14084

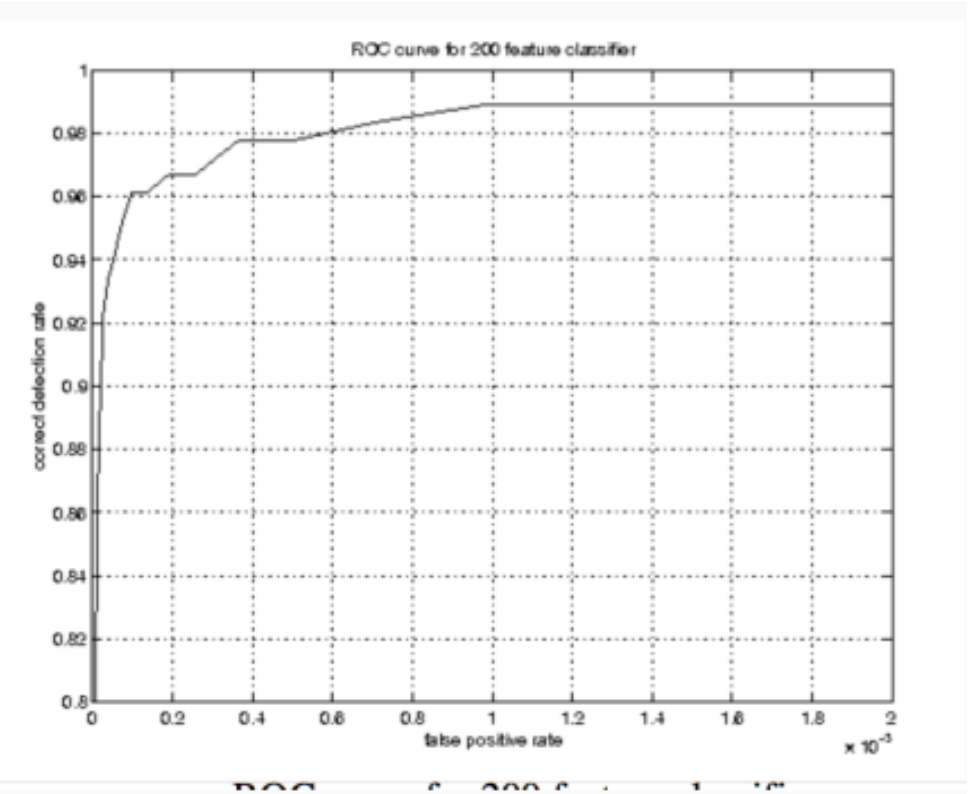
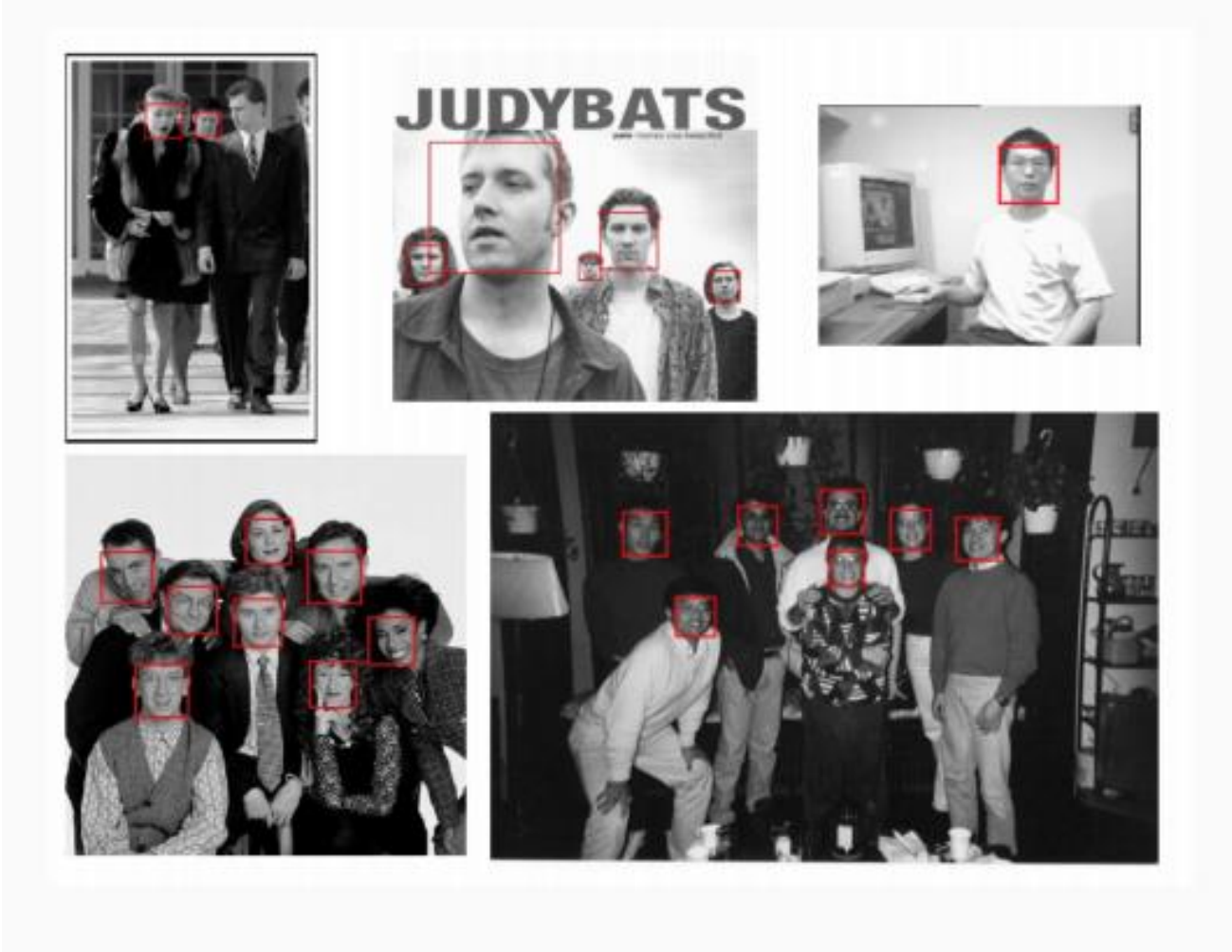



**Figure 3:** ROC curve for 200 feature classifier

# Face Detection Performance

- Output of Face Detection on Test Images



# Schedule

- Ensemble II (11.12)
- Clustering (11.19)
- Dimensional Reduction (11.26)
- Semi-Supervised Learning (12.3)
- Presentation (12.10)
- Presentation (12.17)
- Graphical Model (12.24)
- Reinforcement Learning (12.31)
- Exam (1.7)

# Thanks!