## CS442/542b: Artificial Intelligence II
## Prof. Olga Veksler

# Lecture 5: Machine Learning

# Boosting

## Boosting: motivation

- It is usually hard to design an accurate classifier which generalizes well
- However it is usually easy to find many "rule of thumb" or "*weak*" classifiers
  - A classifier is weak if it is only slightly better than random guessing
  - Weak classifier example: if an email has word "money" classify it as spam
    - This classifier is likely to be better than random guessing
- Can we combine several weak classifiers to produce an accurate classifier?
  - Question people have been working on since 1980's
  - Ada-Boost (1996) the first practical boosting algorithm

## *Ada Boost*

- Assume we have 2-class classification problem, with labels +1 and -1
  - $y_i \in \{-1,1\}$
- Ada boost will produce a discriminant function:

$$g(x) = \sum_{t=1}^{T} \alpha_t h_t(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots \alpha_T h_T(x)$$

- where $\boldsymbol{h}_t(\boldsymbol{x})$ is a "weak" classifier, for example:

$$h_t(x) = \begin{cases} -1 & \text{if email has word "money"} \\ 1 & \text{if email doesn't have "money} \end{cases}$$

- As usual, the final classifier is the sign of the discriminant function $f_{final}(x) = sign[g(x)]$

## *Idea Behind Ada Boost*

- Algorithm is iterative
- Maintains distribution of weights over the training examples
  - Examples that have not been classified correctly at previous iterations get larger weights
- Initially all weights are equal
- At successive iterations, the weight of misclassified examples is increased, forcing the weak learner to focus on the hard examples in the training set

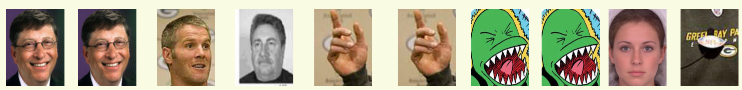
## Idea Behind Ada Boost

- Examples of high weight are going to be shown more often
- face/nonface classification example:

***Round 1***

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1/7 | 1/7 | 1/7 | 1/7 | 1/7 | 1/7 | 1/7 |
| **Best weak classifier:** | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ |
| **Change Weights:** | 1/16 | 1/4 | 1/16 | 1/16 | 1/4 | 1/16 | 1/4 |

***Round 2***



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ |
| **Change Weights:** | 1/8 | | 1/32 | 11/32 | | 1/2 | 1/8 | | 1/32 | 1/32 |

## Idea Behind Ada Boost

***Round 3***

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |

- we choose the best classifier at round 3
- we assume there is always a weak classifier better than random (better than 50% error)
-  image is half of the data given to the classifier
- a better than random classifier will **have to** classify this image correctly

## More Comments on Ada Boost

- Ada boost is very simple to implement, provided you have an implementation of a "weak learner"
- Will work as long as the "basic" classifier $\boldsymbol{h}_t(\boldsymbol{x})$ is at least slightly better than random
  - will work if the error rate of $\boldsymbol{h}_t(\boldsymbol{x})$ is less than 0.5
  - 0.5 is the error rate of a random guessing classifier for a 2-class problem
- Can be applied to boost any classifier, not necessarily weak
  - but there may be no benefits in boosting a "strong" classifier

## Ada Boost for 2 Classes

**Initialization step:** for each example $\boldsymbol{x}$, set

$$D(\mathbf{x}) = \frac{1}{N}, \text{ where } N \text{ is the number of examples}$$

**Iteration step** (for $t = 1 \ldots T$):

1. Find best weak classifier $h_t(\mathbf{x})$ using weights $D(\mathbf{x})$
2. Compute the error rate $\varepsilon_t$ as

$$\varepsilon_t = \sum_{i=1}^{N} D(\mathbf{x}_i) \cdot \underbrace{I[y_i \neq h_t(\mathbf{x}_i)]}_{= \begin{cases} 1 & \text{if } y_i \neq h_t(x_i) \\ 0 & \text{otherwise} \end{cases}}$$

3. assign weight $\alpha_t$ to classifier $h_t$ in the final hypothesis

$$\alpha_t = \log((1 - \varepsilon_t)/\varepsilon_t)$$

4. For each $\mathbf{x}_i$, $D(\mathbf{x}_i) = D(\mathbf{x}_i) \cdot \exp(\alpha_t \cdot I[y_i \neq h_t(\mathbf{x}_i)])$
5. Normalize $D(\mathbf{x}_i)$ so that $\sum_{i=1}^{N} D(\mathbf{x}_i) = 1$

$$\boxed{f_{final}(\mathbf{x}) = \text{sign} [ \sum \alpha_t h_t(\mathbf{x}) ]}$$

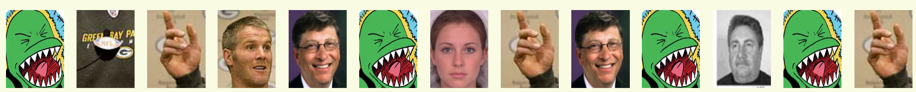
## Ada Boost: step by step

1. Find best weak classifier $h_t(\mathbf{x})$ using weights $\mathbf{D}(\mathbf{x})$

- Some classifiers accept weighted samples, but most don't
- If the classifier does not take weighted samples, this step is done by sampling from the training samples according to the distribution $D(\mathbf{x})$

**1/16 1/4 1/16 1/16 1/4 1/16 1/4**

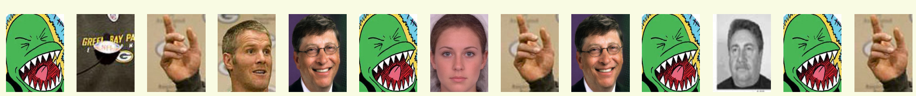
- Draw **k** samples, each $\mathbf{x}$ with probability equal to $D(\mathbf{x})$:



## Ada Boost: step by step

1. Find best weak classifier $h_t(\mathbf{x})$ using weights $\mathbf{D}(\mathbf{x})$

- Give to the classifier the following re-sampled examples:



- To find the best weak classifier, go through ALL weak classifiers, and find the on that works best (gives smallest error) on the collection above

| | $h_1(\mathbf{x})$ | $h_2(\mathbf{x})$ | $h_3(\mathbf{x})$ | ............... | $h_m(\mathbf{x})$ |
|---|---|---|---|---|---|
| errors: | 0.46 | 0.36 | 0.16 | | 0.43 |

*the best classifier $h_t(x)$ at iteration t*

## Ada Boost: step by step

2. Compute $\varepsilon_t$ the error rate as

$$\varepsilon_t = \sum D(x_i) \cdot I[y_i \neq h_t(x_i)]$$

- where $I[y_i \neq h_t(x_i)] = \begin{cases} 1 & \text{if} \quad y_i \neq h_t(x_i) \\ 0 & \text{otherwise} \end{cases}$

| | | | | | | |
|---|---|---|---|---|---|---|
| 1/16 | 1/4 | 1/16 | 1/16 | 1/4 | 1/16 | 1/4 |
| ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ |

$$\varepsilon_t = \frac{1}{4} + \frac{1}{16} = \frac{5}{16}$$

- $\varepsilon_t$ is simply the weight of all misclassified examples added
  - notice that error rate is computed over original examples, not the re-sampled examples
- If a weak classifier is better than random, then $\varepsilon_t < ½$

## Ada Boost: step by step

3. assign weight $\alpha_t$ to classifier $h_t$ in the final hypothesis

$$\alpha_t = \log\left((1 - \varepsilon_t)/\varepsilon_t\right)$$

Example from previous slide:

$$\varepsilon_t = \frac{5}{16} \Rightarrow \alpha_t = \log \frac{1 - \frac{5}{16}}{\frac{5}{16}} = \log \frac{11}{5} \approx 0.8$$

- Recall that $\varepsilon_t < ½$
- Thus $(1 - \varepsilon_t)/\varepsilon_t > 1 \Rightarrow \alpha_t > 0$
- The smaller is $\varepsilon_t$, the larger is $\alpha_t$, and thus the more importance (weight) classifier $h_t(x)$ gets in the final classifier

$$f_{final}(x) = \text{sign}\left[\sum \alpha_t h_t(x)\right]$$

## Ada Boost: step by step

4. For each $\boldsymbol{x_i}$, $\boldsymbol{D(x_i)} = \boldsymbol{D(x_i)} \cdot \exp[\alpha_t \cdot \boldsymbol{I}(\boldsymbol{y_i} \neq \boldsymbol{h_t(x_i)})]$

Example from previous slide: $\alpha_t = 0.8$

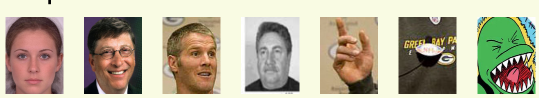
| | | | | | | |
|---|---|---|---|---|---|---|
| 1/16 | 1/4 | 1/16 | 1/16 | 1/4 | 1/16 | 1/4 |
| ✓ | ✓ | ✓ | ✖ | ✖ | ✓ | ✓ |
| ⇓ | ⇓ | ⇓ | ⇓ | ⇓ | ⇓ | ⇓ |
| 1/16 | 1/4 | 1/16 | (1/16) exp(0.8) | (1/4) exp(0.8) | 1/16 | 1/4 |

- Weight of misclassified examples is increased and the new $\boldsymbol{D}(\boldsymbol{x_i})$'s are normalized to be a distribution again

## Ada Boost: step by step

5. Normalize $\boldsymbol{D(x_i)}$ so that $\sum \boldsymbol{D(x_i)} = 1$

Example from previous slide:



| | | | | | | |
|---|---|---|---|---|---|---|
| 1/16 | 1/4 | 1/16 | 0.14 | 0.56 | 1/16 | 1/4 |

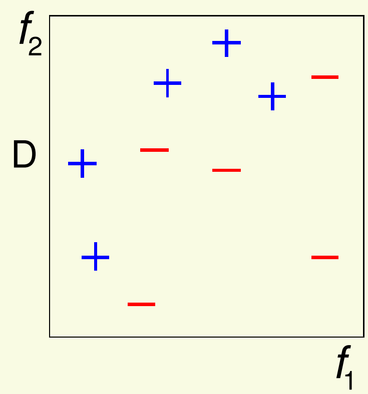
After normalization:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.05 | 0.18 | 0.05 | 0.10 | 0.40 | 0.05 | 0.18 |

- In matlab, if D is a vector storing weights, D = D./sum(D)

# *AdaBoost  Example*

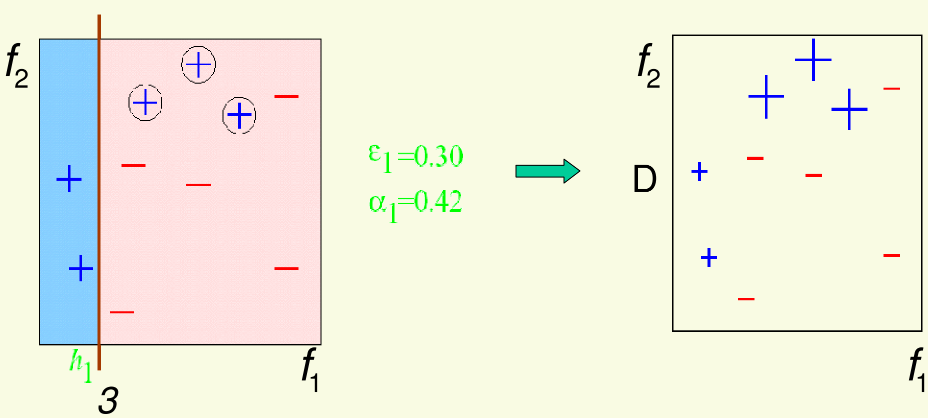
from "**A Tutorial on Boosting" by** Yoav Freund and Rob Schapire




Original Training set : equal weights to all training samples

# *AdaBoost Example*

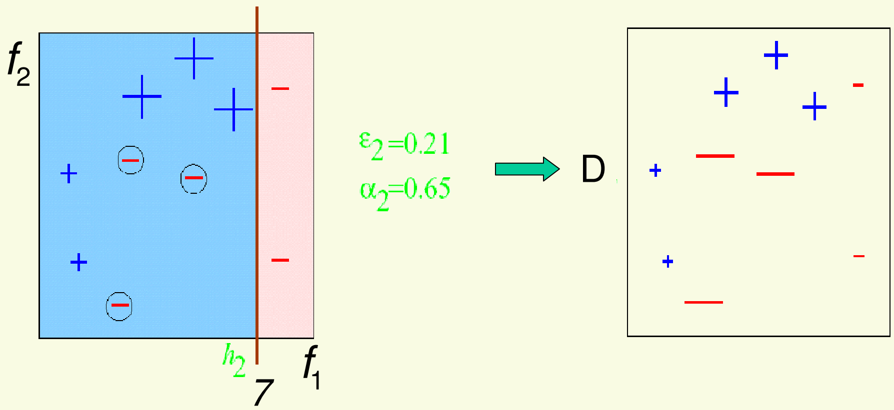
ROUND 1




$$h_1(x) = sign(f_1 - 3)$$

## AdaBoost Example

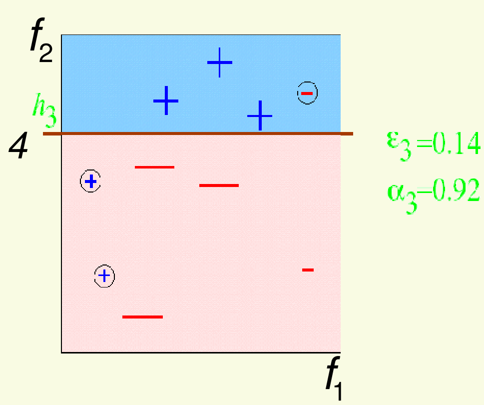
ROUND 2




$$h_2(x) = sign(f_1 - 7)$$

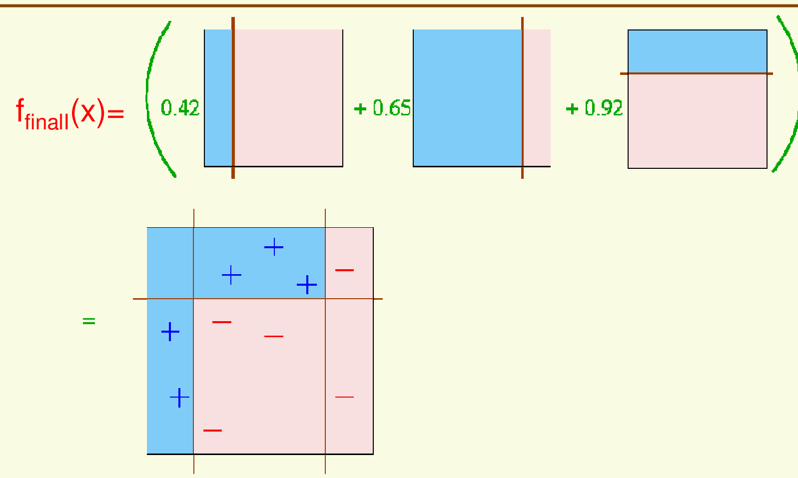
## AdaBoost Example

ROUND 3




$$h_3(x) = sign(f_2 - 4)$$

## AdaBoost Example




$$f_{final}(x) = sign\begin{bmatrix} 0.42 \cdot sign(f_1 - 3) + 0.65 \cdot sign(f_1 - 7) + \\ + 0.92 \cdot sign(f_2 - 4) \end{bmatrix}$$

## AdaBoost Comments

- It can be shown that the training error drops exponentially fast, if each weak classifier is slightly better than random

$$\boldsymbol{Err_{train} \le \exp\left(-2\sum_t \gamma_t^2\right)}$$

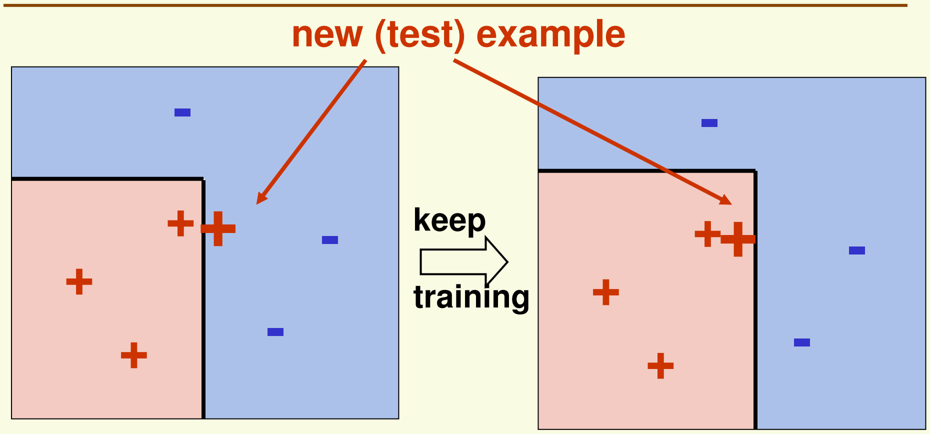
- Here $\gamma_t = \varepsilon_t - 1/2$, where is classification error at round $t$ (weak classifier $f_t$ )
- Example: let errors for the first four rounds be, 0.3, 0.14, 0.06, 0.03, 0.01 respectively. Then

$$Err_{train} \le exp\left[-2\left(0.2^2 + 0.36^2 + 0.44^2 + 0.47^2 + 0.49^2\right)\right] \approx 0.19$$

## AdaBoost Comments

- But we are really interested in the generalization properties of $f_{FINAL}(x)$, not the training error
- AdaBoost was shown to have excellent generalization properties in practice
  - the more rounds, the more complex is the final classifier, so overfitting is expected as the training proceeds
  - but in the beginning researchers observed no overfitting of the data
  - It turns out it does overfit data eventually, if you run it really long
- It can be shown that boosting "aggressively" increases the margins of training examples, as iterations proceed
  - margins continue to increase even when training error reaches zero
  - Helps to explain empirically observed phenomena: test error continues to drop even after training error reaches zero

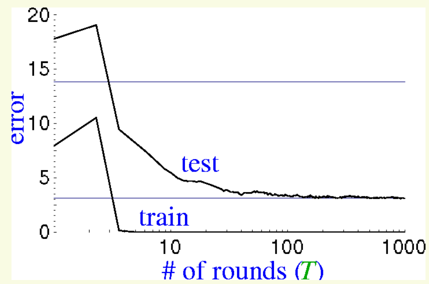
## AdaBoost Example

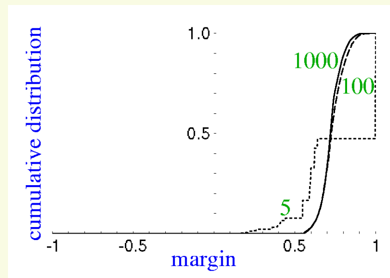



- Zero training error

- Zero training error
- Better "margins"
  - better generalization

## *The Margin Distribution*




| | | | |
|---|---|---|---|
| Iteration number | 5 | 100 | 1000 |
| training error | 0.0 | 0.0 | 0.0 |
| test error | 8.4 | 3.3 | 3.1 |
| %margins≤0.5 | 7.7 | 0.0 | 0.0 |
| Minimum margin | 0.14 | 0.52 | 0.55 |

## *Practical Advantages of AdaBoost*

- fast
- simple
- has only one parameter to tune ($T$)
- flexible: can be combined with any classifier
- provably effective (assuming weak learner)
  - shift in mind set: goal now is merely to find hypotheses that are better than random guessing

## *Caveats*

- AdaBoost can fail if
  - weak hypothesis too complex (overfitting)
  - weak hypothesis too weak ($\gamma_t \rightarrow 0$ too quickly),
    - underfitting
- empirically, AdaBoost seems especially susceptible to noise
  - noise is the data with wrong labels