CSE 176 Introduction to Machine Learning
Lecture 15: Ensemble models

Some materials from Olga Veksler, Robin Dhamankar, Vandi Verma & Sebastian Thrun

What we have learnt so far…

- Machine learning models
  - Bayesian classifier
  - K-nearest neighbor
  - Perceptron (Linear classifier)
  - Neural Network
  - Decision tree
- There are single models

Ensemble of multiple models

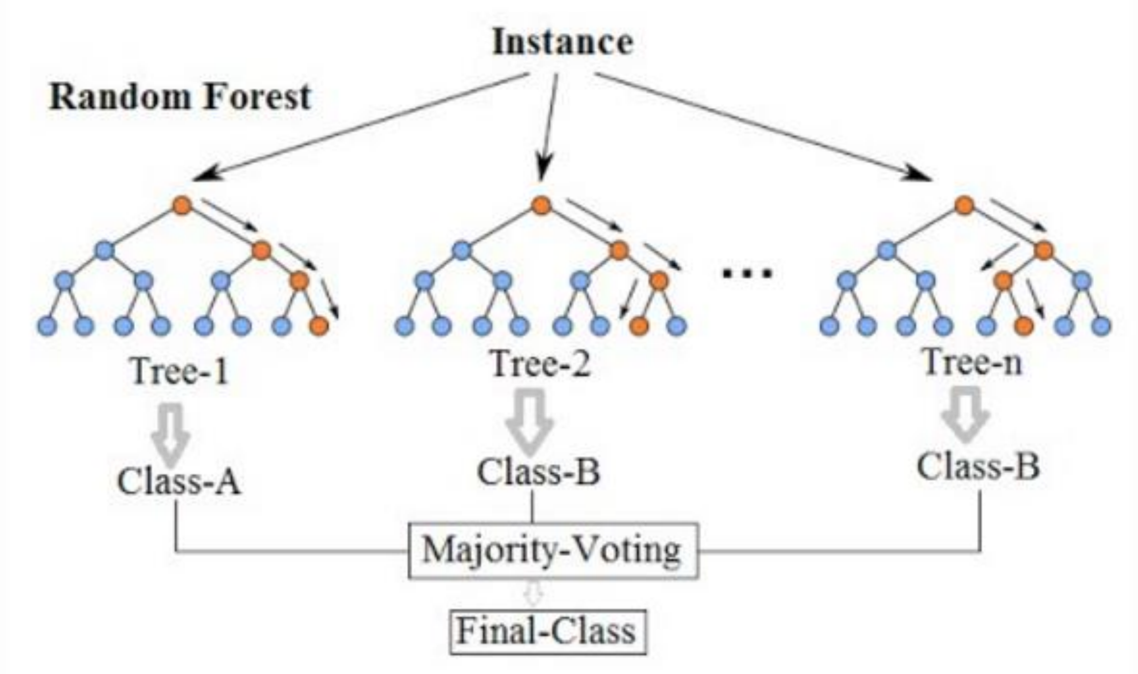
- Use multiple models, and “average” the predictions
- From statistics, it is good to average your predictions, reduces variance
- Consider L i.i.d. random variables $y_1, \ldots, y_L$ with expected value $\mathrm{E}\{y_l\} = \mu$ and variance $\mathrm{var}\{y_l\} = \sigma^2$

$$\mathrm{E}\{y\} \stackrel{✎}{=} \mathrm{E}\left\{\frac{1}{L}\sum_{l=1}^{L} y_l\right\} = \frac{1}{L}\sum_{l=1}^{L}\mathrm{E}\{y_l\} = \mu$$

$$\mathrm{var}\{y\} \stackrel{✎}{=} \mathrm{var}\left\{\frac{1}{L}\sum_{l=1}^{L} y_l\right\} = \frac{1}{L^2}\mathrm{var}\left\{\sum_{l=1}^{L} y_l\right\} = \frac{1}{L^2}\sum_{l=1}^{L}\mathrm{var}\{y_l\} = \frac{1}{L}\sigma^2.$$

Example: Random forest

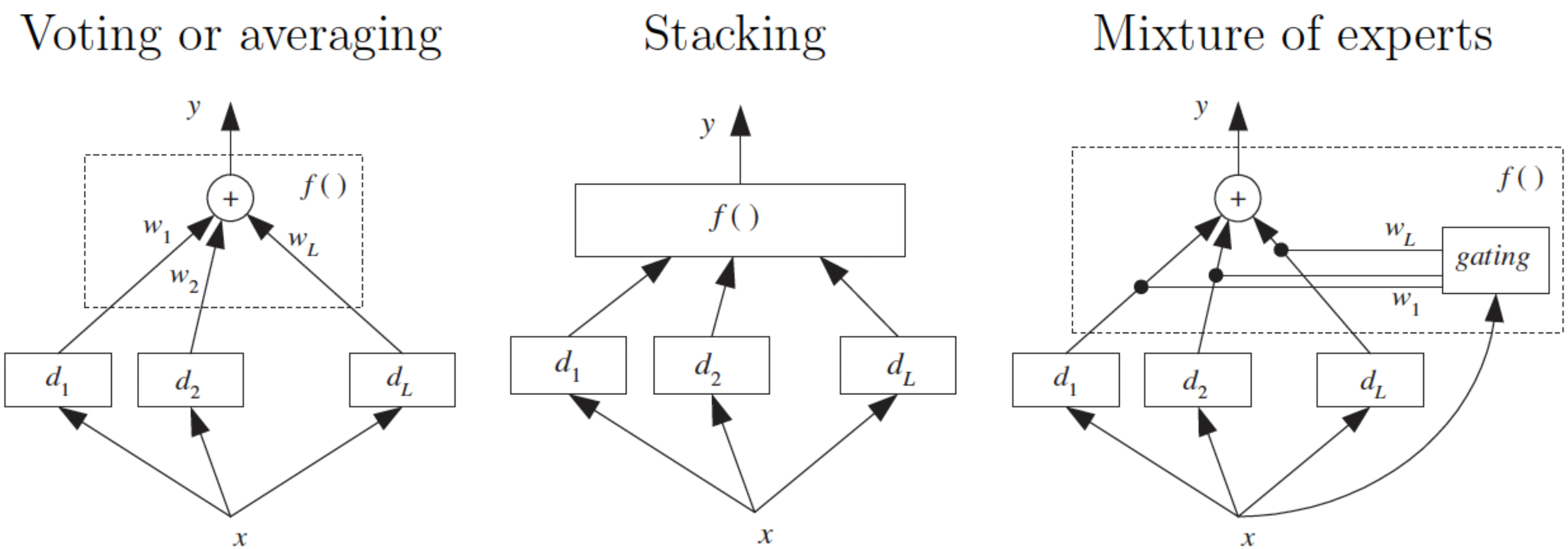
- Train an ensemble of L decision trees on L different subsets of the training set
- Define the ensemble output for a test instance as the majority vote (for classification) or the average (for regression) of the L trees



Mechanisms to generate diversity

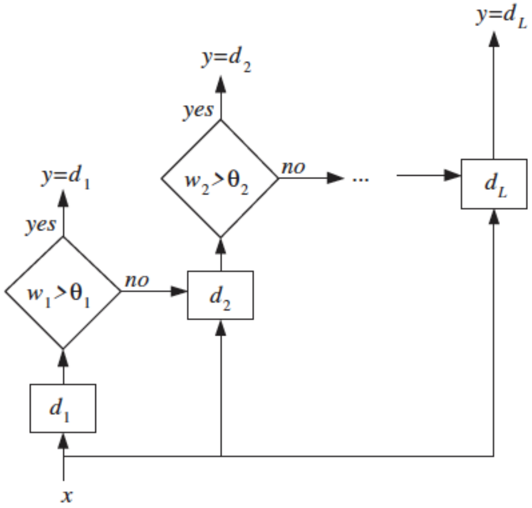
- ❑Different models: linear, neural net, decision tree. . .
- ❑Different hyperparameters
- ❑Different optimization algorithm or initialization
- ❑Different features
- ❑Different training sets

Model combination scheme



Model combination scheme

Boosting or cascading



Bagging

- We generate L (partly different) subsets of the training set
- We train L learners, each on a different subset
- The ensemble output is defined as the vote or average
- Random forest: a variation of bagging

Boosting

- Weak learner: a learner that has probability of error < 1/2 (i.e., better than random guessing on binary classification).
  - Ex: decision trees with only 1 or 2 levels.
- Strong learner: a learner that can have arbitrarily small probability of error.
  - Ex: neural net
- Boosting combines many weak learners to a strong learner

Ada Boost

- Assume 2-class problem, with labels +1 and -1
  - $\mathbf{y}^i$ in {-1,1}
- Ada boost produces a discriminant function:

$$\mathbf{g}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \mathbf{h}_t(\mathbf{x}) = \alpha_1 \mathbf{h}_1(\mathbf{x}) + \alpha_2 \mathbf{h}_2(\mathbf{x}) + \ldots \alpha_T \mathbf{h}_T(\mathbf{x})$$

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

$$\mathbf{h}_t(\mathbf{x}) = \begin{cases} -1 & \text{if email has word “money”} \\ 1 & \text{if email does not have word “money”} \end{cases}$$

- The final classifier is the sign of the discriminant function

$$\mathbf{f}_{final}(\mathbf{x}) = \text{sign}[\mathbf{g}(\mathbf{x})]$$

Idea Behind Ada Boost

- Maintains distribution of weights over the training examples
- Initially weights are equal
- Main Idea: at successive iterations, the weight of misclassified examples is increased

Idea Behind Ada Boost

- Examples of high weight are shown more often at later rounds
- Face/nonface classification problem:

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

best weak classifier:	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓
change weights:		1/8	1/32	11/32		1/2		1/8	1/32	1/32

Idea Behind Ada Boost

Round 3

- out of all available weak classifiers, we choose the one that works best on the data we have at round 3

Idea Behind Ada Boost

Round 3

- out of all available weak classifiers, we choose the one that works best on the data we have 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
- chosen weak classifier **has to** classify this image correctly

More Comments on Ada Boost

- Ada boost is simple to implement, provided you have an implementation of a “weak learner”

More Comments on Ada Boost

- Ada boost is simple to implement, provided you have an implementation of a "weak learner"
- Will work as long as the "basic" classifier $\mathbf{h}_t(\mathbf{x})$ 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 for 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 $\mathbf{x}$, set

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

Iteration step (for $\mathbf{t} = 1 \ldots T$):

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

$$\boldsymbol{\varepsilon_t} = \sum_{i=1}^{N} \mathbf{D}(\mathbf{x}^i) \cdot \mathbf{I}[\mathbf{y}^i \neq \mathbf{h_t}(\mathbf{x}^i)]$$

3. compute weight $\boldsymbol{\alpha_t}$ of classifier $\mathbf{h_t}$

$$\boldsymbol{\alpha_t} = \mathbf{log}\left((1 - \boldsymbol{\varepsilon_t}) / \boldsymbol{\varepsilon_t}\right)$$

4. For each $\mathbf{x}^i$, $\mathbf{D}(\mathbf{x}^i) = \mathbf{D}(\mathbf{x}^i) \cdot \mathbf{exp}\left(\boldsymbol{\alpha}_t \cdot \mathbf{I}[\mathbf{y}^i \neq \mathbf{h_t}(\mathbf{x}^i)]\right)$
5. Normalize $\mathbf{D}(\mathbf{x}^i)$ so that $\sum_{i=1}^{N} \mathbf{D}(\mathbf{x}^i) = 1$

$$\boxed{\mathbf{f}_{\text{final}}(\mathbf{x}) = \text{sign}\left[\sum \boldsymbol{\alpha_t}\, \mathbf{h_t}(\mathbf{x})\right]}$$

Ada Boost: Step 1

1. Find best weak classifier $\mathbf{h}_t(\mathbf{x})$ using weights $\mathbf{D}(\mathbf{x})$
    - some classifiers accept weighted samples, not all
    - if classifier does not take weighted samples, sample from the training samples according to the distribution $\mathbf{D}(\mathbf{x})$

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

Ada Boost: Step 1

1. Find best weak classifier $\mathbf{h}_t(\mathbf{x})$ using weights $\mathbf{D}(\mathbf{x})$
   - some classifiers accept weighted samples, not all
   - if classifier does not take weighted samples, sample from the training samples according to the distribution $\mathbf{D}(\mathbf{x})$

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

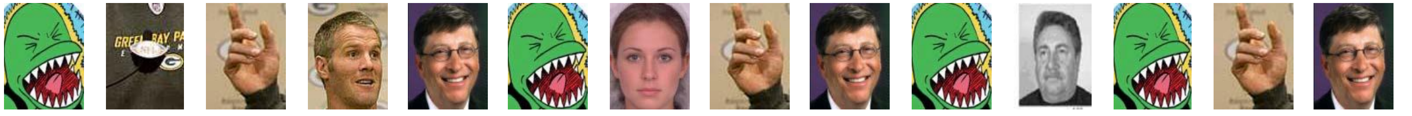
- Draw **k** samples, each **x** with probability equal to $\mathbf{D}(\mathbf{x})$:

re-sampled examples

Ada Boost: Step 1

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

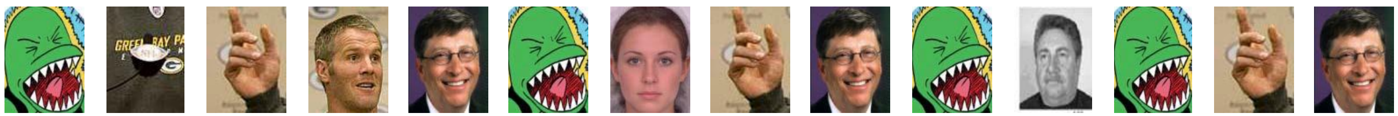
- Give to the classifier the re-sampled examples:



Ada Boost: Step 1

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

- Give to the classifier the re-sampled examples:



- To find the best weak classifier, go through **all** weak classifiers, and find the one that gives the smallest error on the re-sampled examples

weak classifiers	$\mathbf{h}_1(\mathbf{x})$	$\mathbf{h}_2(\mathbf{x})$	$\mathbf{h}_3(\mathbf{x})$	………	$\mathbf{h}_m(\mathbf{x})$
errors:	0.46	0.36	0.16		0.43

the best classifier $h_t(x)$
to choose at iteration t

Ada Boost: Step 2

2. Compute $\boldsymbol{\varepsilon}_t$ the error rate as

$$\varepsilon_t = \sum_{i=1}^{N} D(x^i) \cdot I\left[y^i \neq h_t(x^i)\right]$$

1/16	1/4	1/16	1/16	1/4	1/16	1/4
✓	✓	✓	✗	✗	✓	✓

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

- $\boldsymbol{\varepsilon}_t$ is the weight of all misclassified examples added
  - the error rate is computed over original examples, not the re-sampled examples
- If a weak classifier is better than random, then $\boldsymbol{\varepsilon}_t < ½$

Ada Boost: Step 3

3. compute weight $\boldsymbol{\alpha}_t$ of classifier $\mathbf{h}_t$

$$\boldsymbol{\alpha}_t = \log\left(\left(1 - \boldsymbol{\varepsilon}_t\right)/\boldsymbol{\varepsilon}_t\right)$$

In example from previous slide:

$$\boldsymbol{\varepsilon}_t = \frac{5}{16} \quad \Rightarrow \quad \boldsymbol{\alpha}_t = \log\frac{1-\frac{5}{16}}{\frac{5}{16}} = \log\frac{11}{5} \approx 0.8$$

Ada Boost: Step 3

3. compute weight $\boldsymbol{\alpha}_t$ of classifier $\mathbf{h}_t$

$$\boldsymbol{\alpha}_t = \log\left((1 - \boldsymbol{\varepsilon}_t)/\boldsymbol{\varepsilon}_t\right)$$

In example from previous slide:

$$\boldsymbol{\varepsilon}_t = \frac{5}{16} \quad \Rightarrow \quad \boldsymbol{\alpha}_t = \mathbf{log}\frac{1-\frac{5}{16}}{\frac{5}{16}} = \mathbf{log}\frac{11}{5} \approx 0.8$$

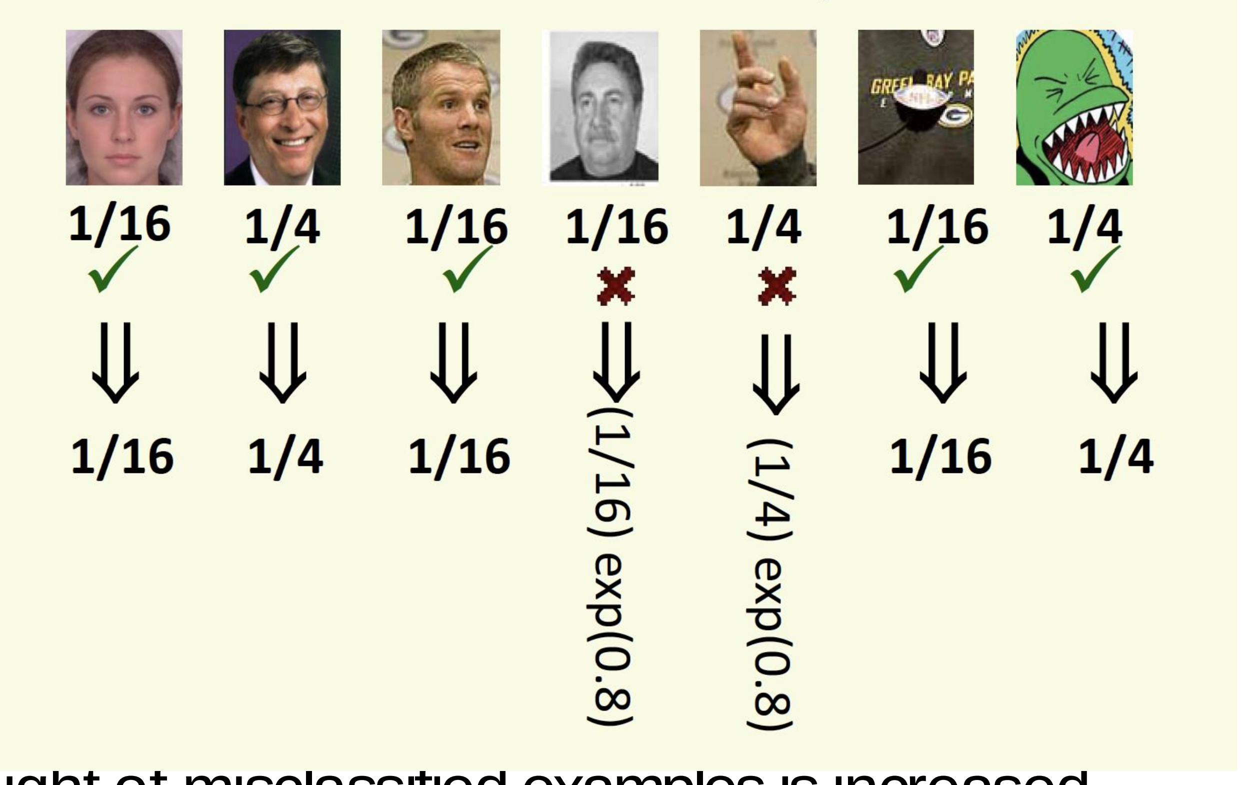
- Recall that $\boldsymbol{\varepsilon}_t < ½$
- Thus $(1 - \boldsymbol{\varepsilon}_t)/\boldsymbol{\varepsilon}_t > 1 \Rightarrow \boldsymbol{\alpha}_t > 0$
- The smaller is $\boldsymbol{\varepsilon}_t$, the larger is $\boldsymbol{\alpha}_t$, and thus the more importance (weight) classifier $\mathbf{h}_t(x)$

$$\text{final}(\mathbf{x}) = \text{sign}\left[\sum \boldsymbol{\alpha}_t \, \mathbf{h}_t(\mathbf{x})\right]$$

Ada Boost: Step 4

4. For each $\mathbf{x}^i$, $\mathbf{D}(\mathbf{x}^i) = \mathbf{D}(\mathbf{x}^i) \cdot \mathbf{exp}(\boldsymbol{\alpha}_t \cdot \mathbf{I}[\mathbf{y}^i \neq \mathbf{h}_t(\mathbf{x}^i)])$

from previous slide $\boldsymbol{\alpha}_t = 0.8$



- weight of misclassified examples is increased

Ada Boost: Step 5

5. Normalize $\mathbf{D}(\mathbf{x}^i)$ so that $\sum \mathbf{D}(\mathbf{x}^i) = 1$

from previous slide:

1/16 1/4 1/16 0.14 0.56 1/16 1/4

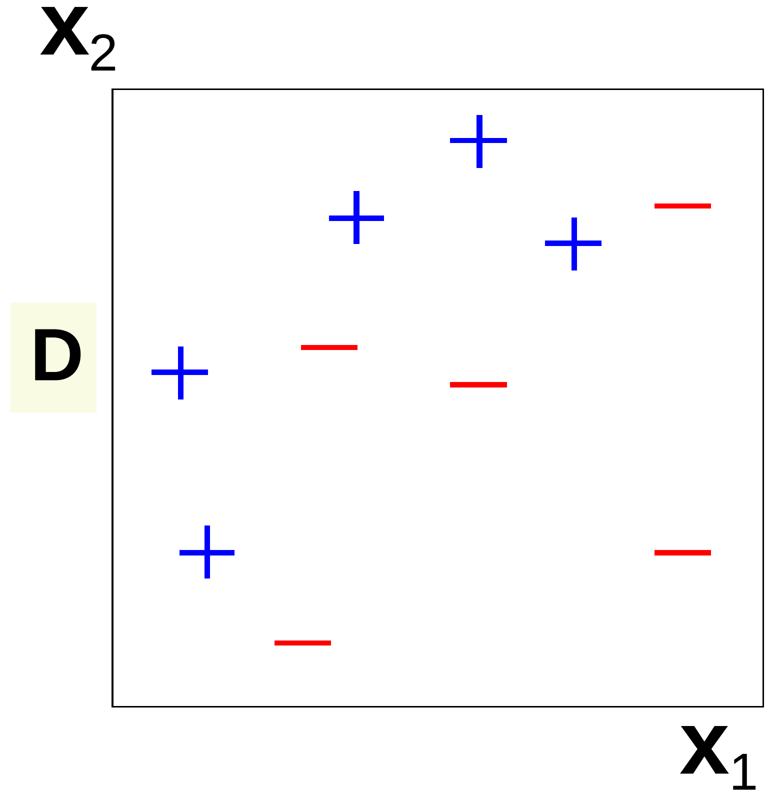
- after normalization

0.05 0.18 0.05 0.10 0.40 0.05 0.18

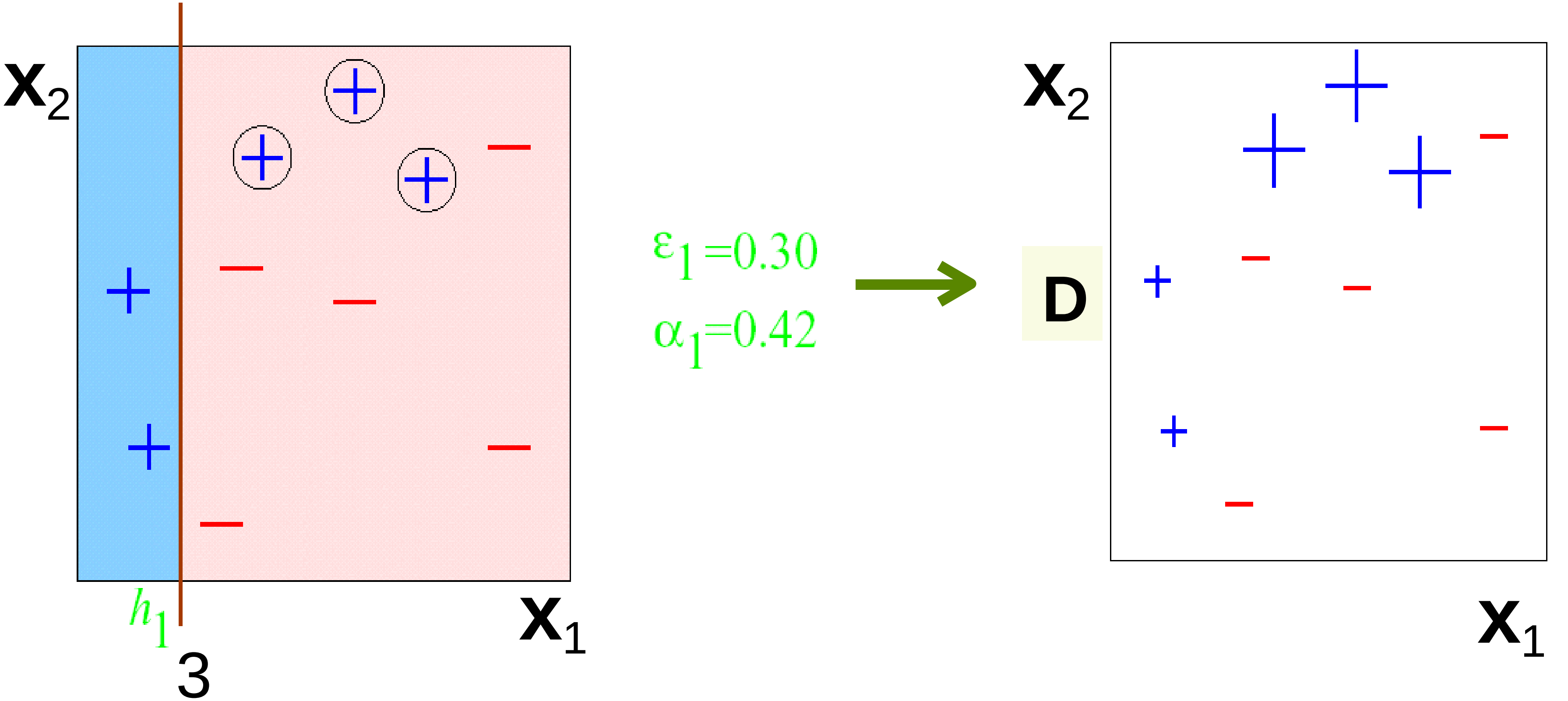
AdaBoost Example

- Initialization: all examples have equal weights



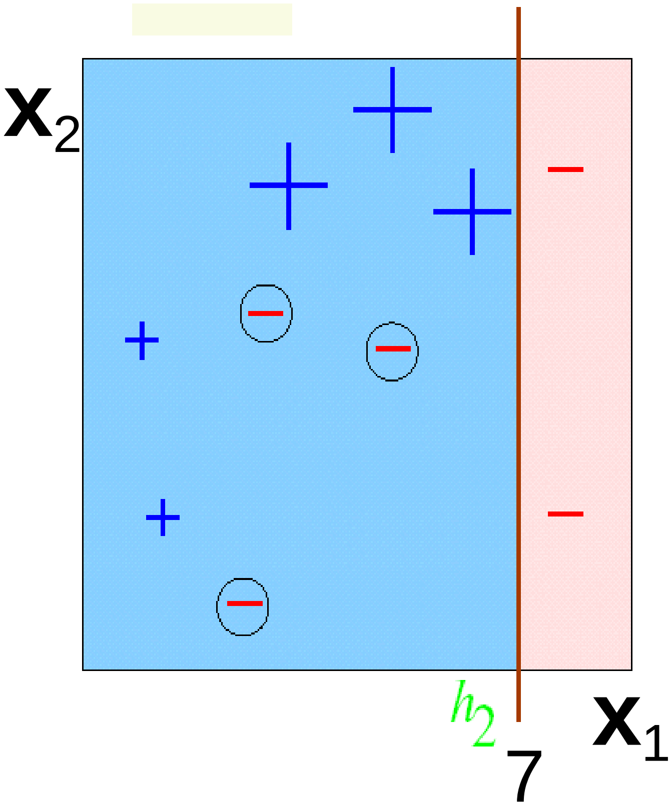
from “A Tutorial on Boosting” by Yoav Freund and Rob Schapire

AdaBoost Example: Round 1



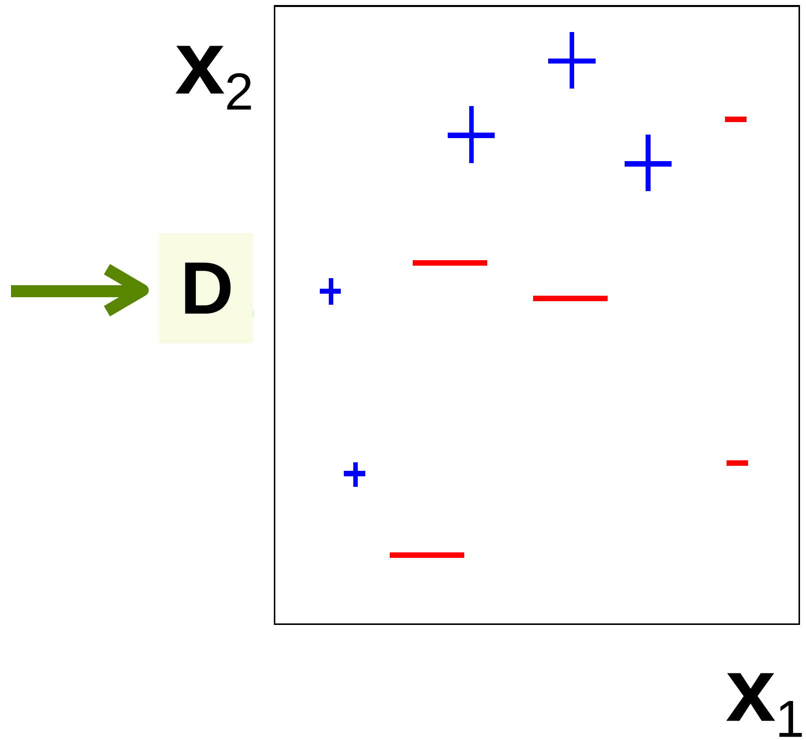
$$h_1(x) = \mathrm{sign}\left(3 - x_1\right)$$

AdaBoost Example: Round 2



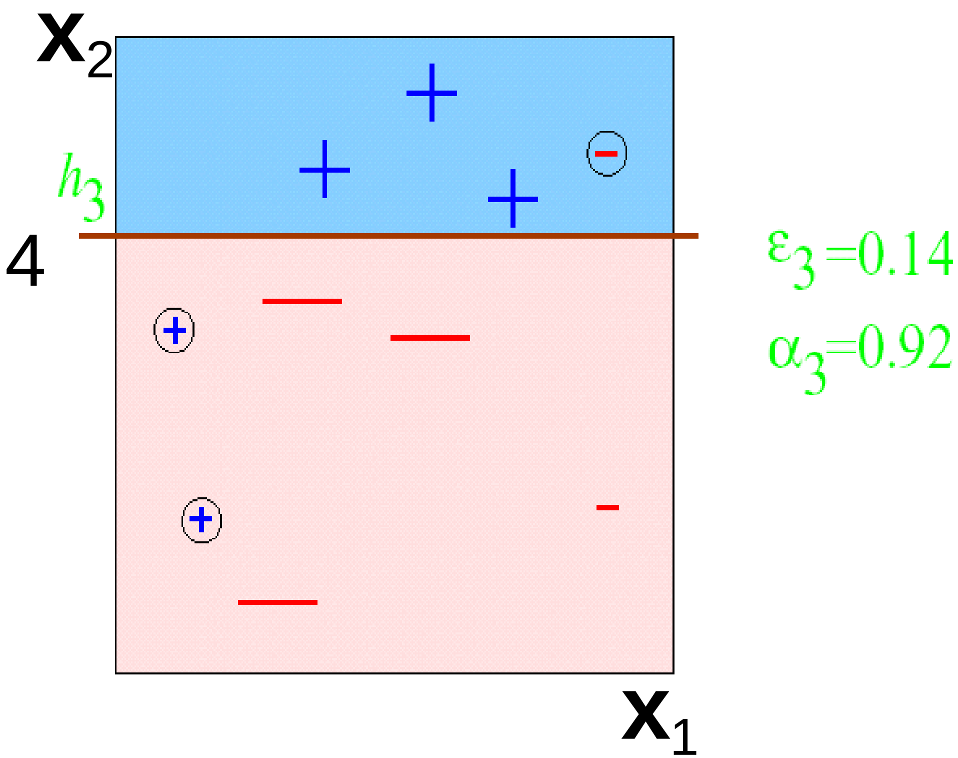
$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$



$$h_2(x) = \text{sign}(7 - x_1)$$

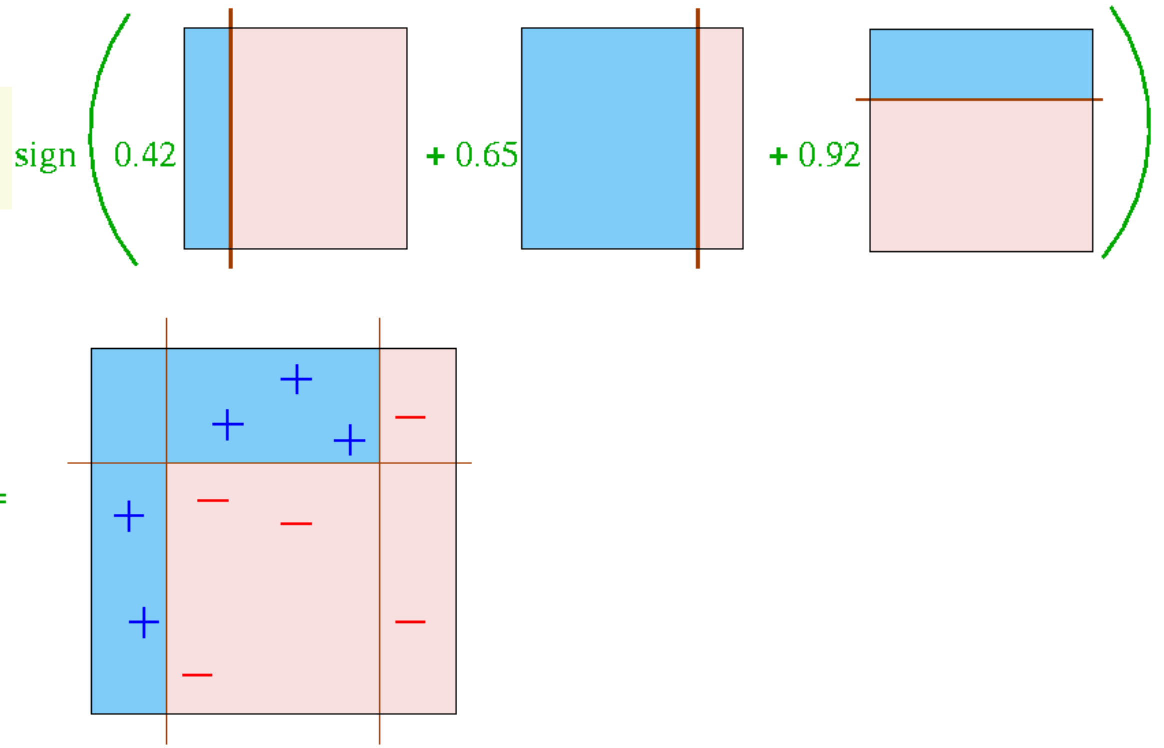
AdaBoost Example: Round 3



$$h_3(x) = \text{sign}(x_2 - 4)$$

AdaBoost Example

$$\mathbf{f}_{\text{final}}(\mathbf{x}) =$$



$$\mathbf{f}_{\mathbf{final}}(\mathbf{x}) = \mathbf{sign}\left(0.42\,\mathbf{sign}(3 - \mathbf{x}_1) + 0.65\,\mathbf{sign}(7 - \mathbf{x}_1) + 0.92\,\mathbf{sign}(\mathbf{x}_2 - 4)\right)$$

- Decision boundary non-linear

Practical Advantages of AdaBoost

- Can construct arbitrarily complex decision regions
- Fast and 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