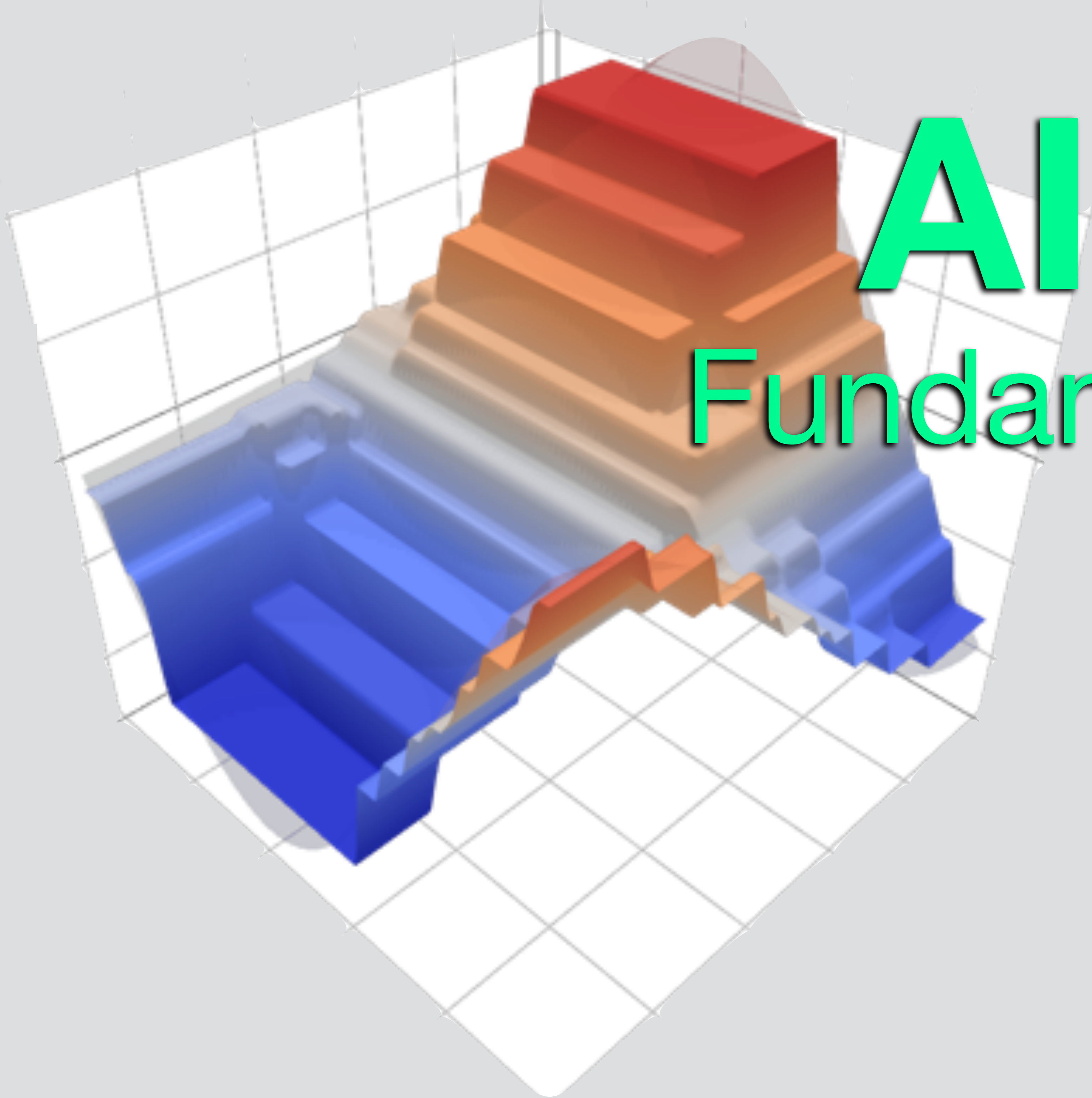


Illustration adapted from Alex Rogozhnikov

AIN311

Fundamentals of Machine Learning

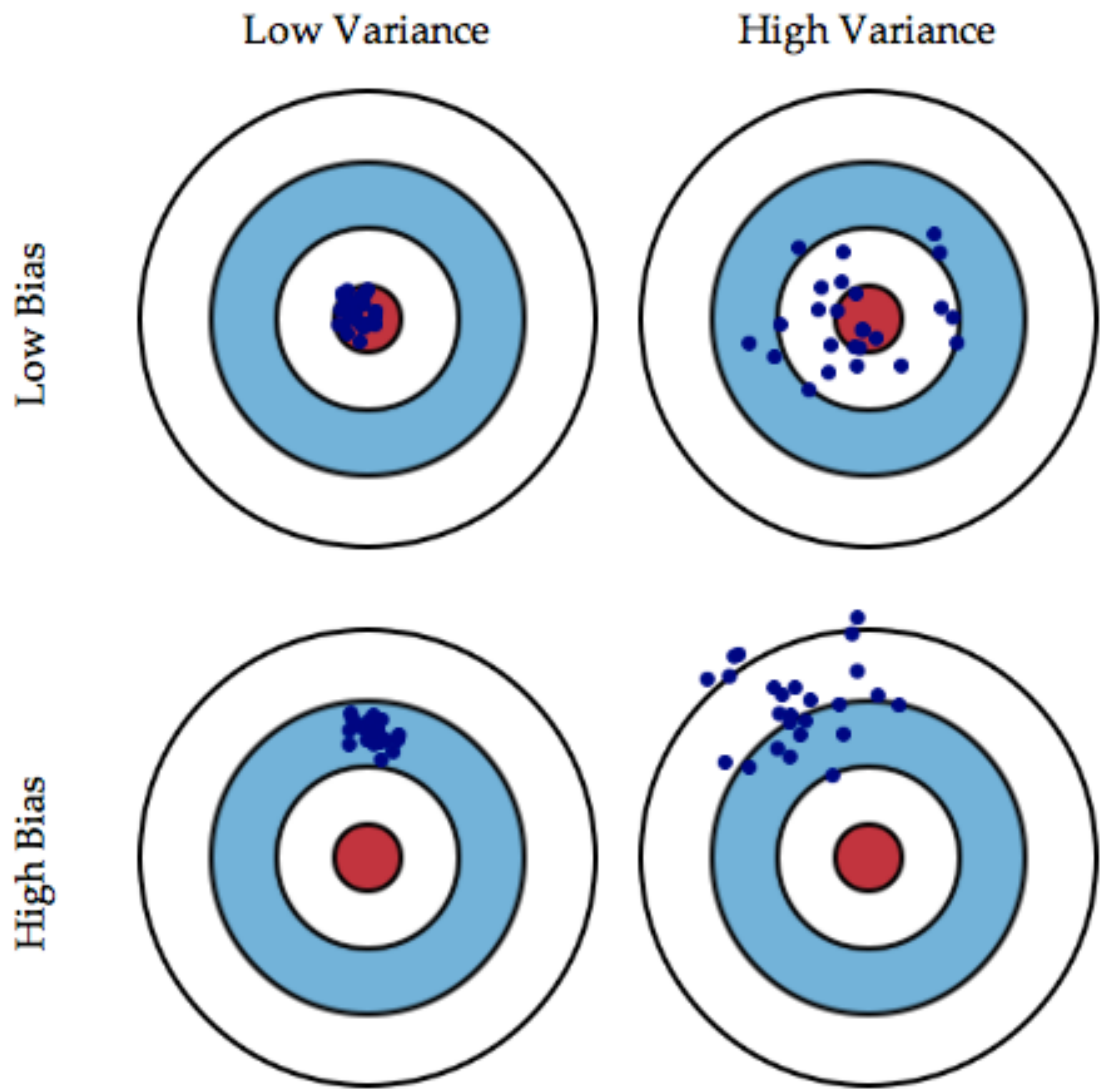
Lecture 20:

AdaBoost

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Erkut Erdem // Hacettepe University // Fall 2024

Last time… **Bias/Variance Tradeoff**



Graphical illustration of bias and variance.

http://scott.fortmann-roe.com/docs/BiasVariance.html

slide by David Sontag

Last time... **Bagging**

- Leo Breiman (1994)
- Take repeated bootstrap samples from training set D.
- **Bootstrap sampling:** Given set D containing N training examples, create D′ by drawing N examples at random with replacement from D.

- Bagging:
  - Create *k* bootstrap samples D_1 ... D_k.
  - Train distinct classifier on each D_i.
  - Classify new instance by majority vote / average.

$$Var(Bagging(L(x, D))) = \frac{Var(L(x, D)}{N}$$

slide by David Sontag

Last time… **Random Forests**

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

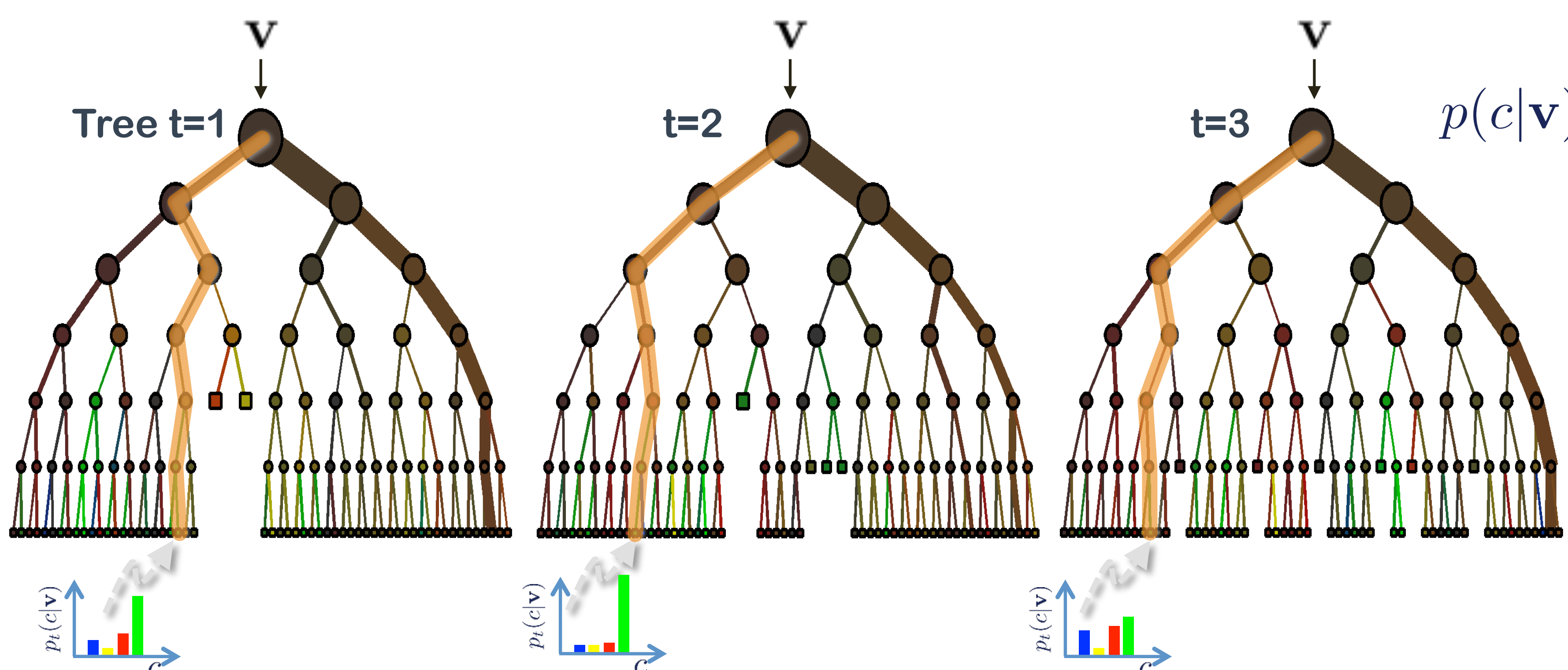
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.



$$p(c|\mathbf{v}) = \frac{1}{T}\sum_t^T p_t(c|\mathbf{v})$$

slide by Nando de Freitas

Boosting

Boosting Ideas

- Main idea: use weak learner to create strong learner.
- Ensemble method: combine base classifiers returned by weak learner.
- Finding simple relatively accurate base classifiers often not hard.
- But, how should base classifiers be combined?

Example: “How May I Help You?”

- **Goal:** automatically categorize type of call requested by phone customer (Collect, CallingCard, PersonToPerson, etc.)
  - `yes I’d like to place a collect call long distance please` (Collect)
  - `operator I need to make a call but I need to bill it to my office` (ThirdNumber)
  - `yes I’d like to place a call on my master card please` (CallingCard)
  - `I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill` (BillingCredit)
- **Observation:**
  - easy to find “rules of thumb” that are “often” correct
    - e.g.: “IF ‘card’ occurs in utterance THEN predict ‘CallingCard’ ”
  - hard to find single highly accurate prediction rule

[Gorin et al.]

Boosting: Intuition

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**
- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most “sure” will vote with more conviction
  - Classifiers will be most “sure” about a particular part of the space
  - On average, do better than single classifier!
- **But how do you???**
  - force classifiers to learn about different parts of the input space?
  - weigh the votes of different classifiers?

slide by Aarti Singh & Barnabas Poczos

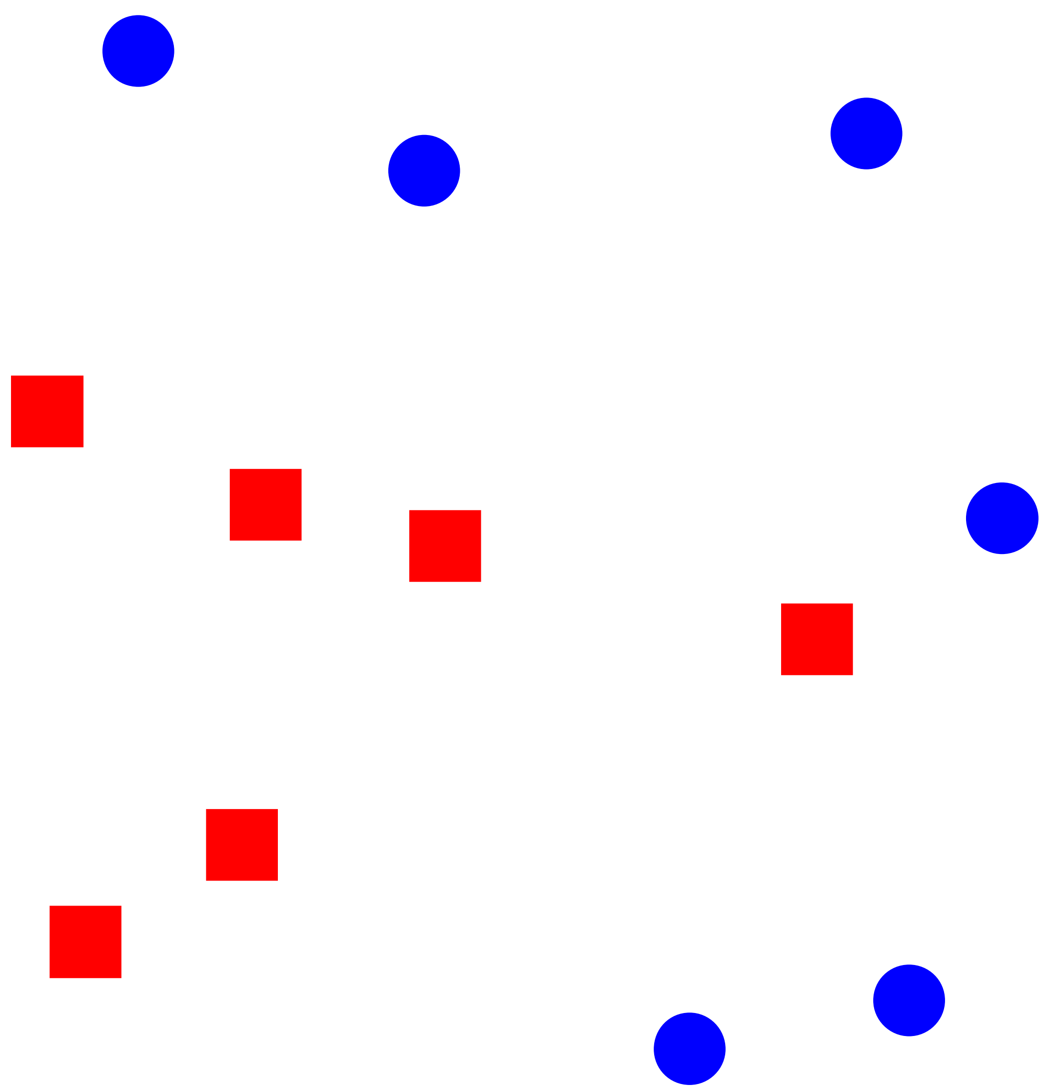
Boosting [Schapire, 1989]

- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let the learned classifiers vote
- On each iteration t:
  - weight each training example by how incorrectly it was classified
  - Learn a hypothesis – h_t
  - A strength for this hypothesis – a_t
- Final classifier:
  - A linear combination of the votes of the different classifiers weighted by their strength $H(X) = \text{sign}\left(\sum \alpha_t h_t(X)\right)$
- **Practically useful**
- **Theoretically interesting**

slide by Aarti Singh & Barnabas Poczos

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,\ldots,M$

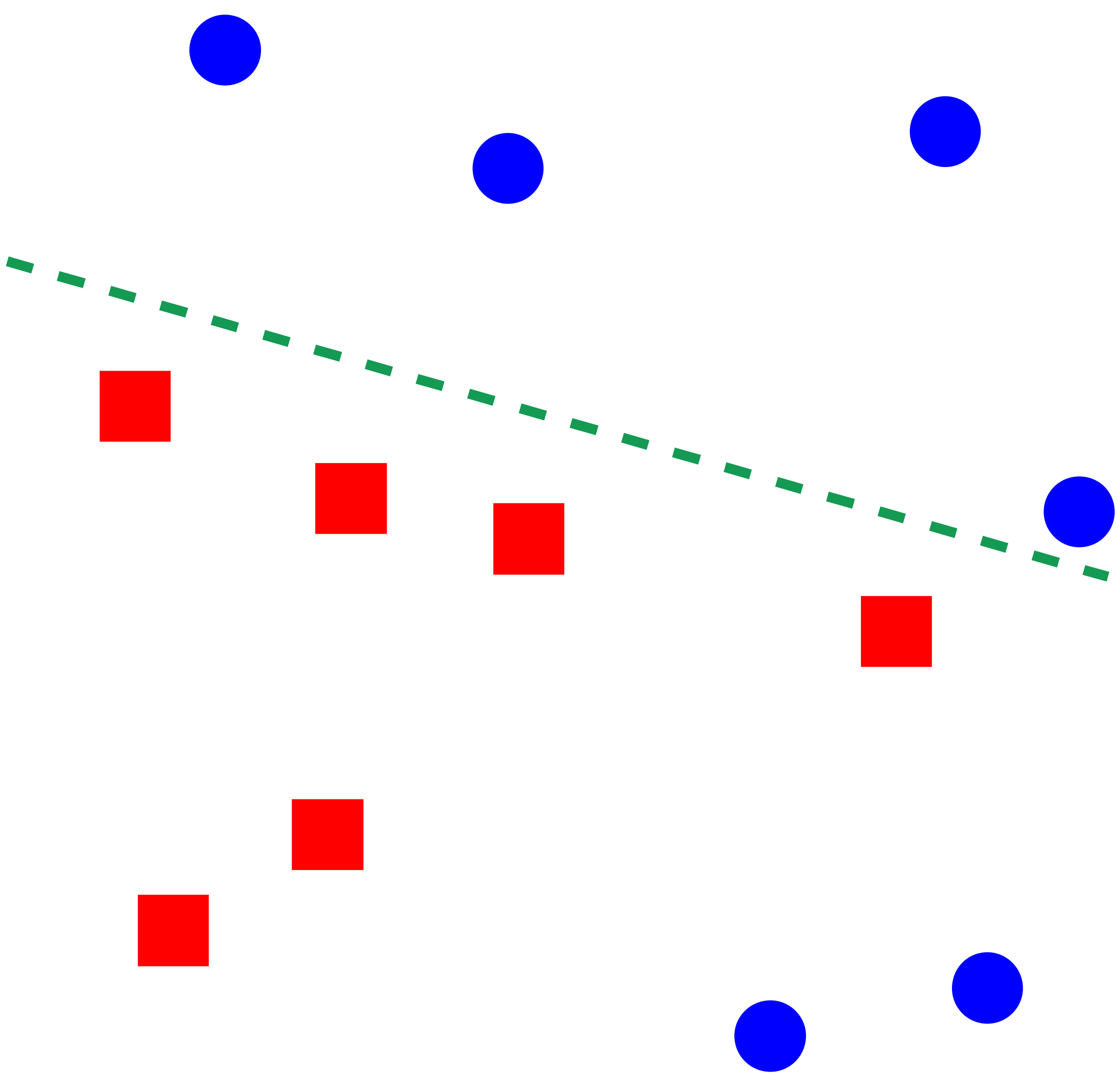
- Pick a weak classifier h_m
- Adjust weights: misclassified examples get “heavier”
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,...,M$

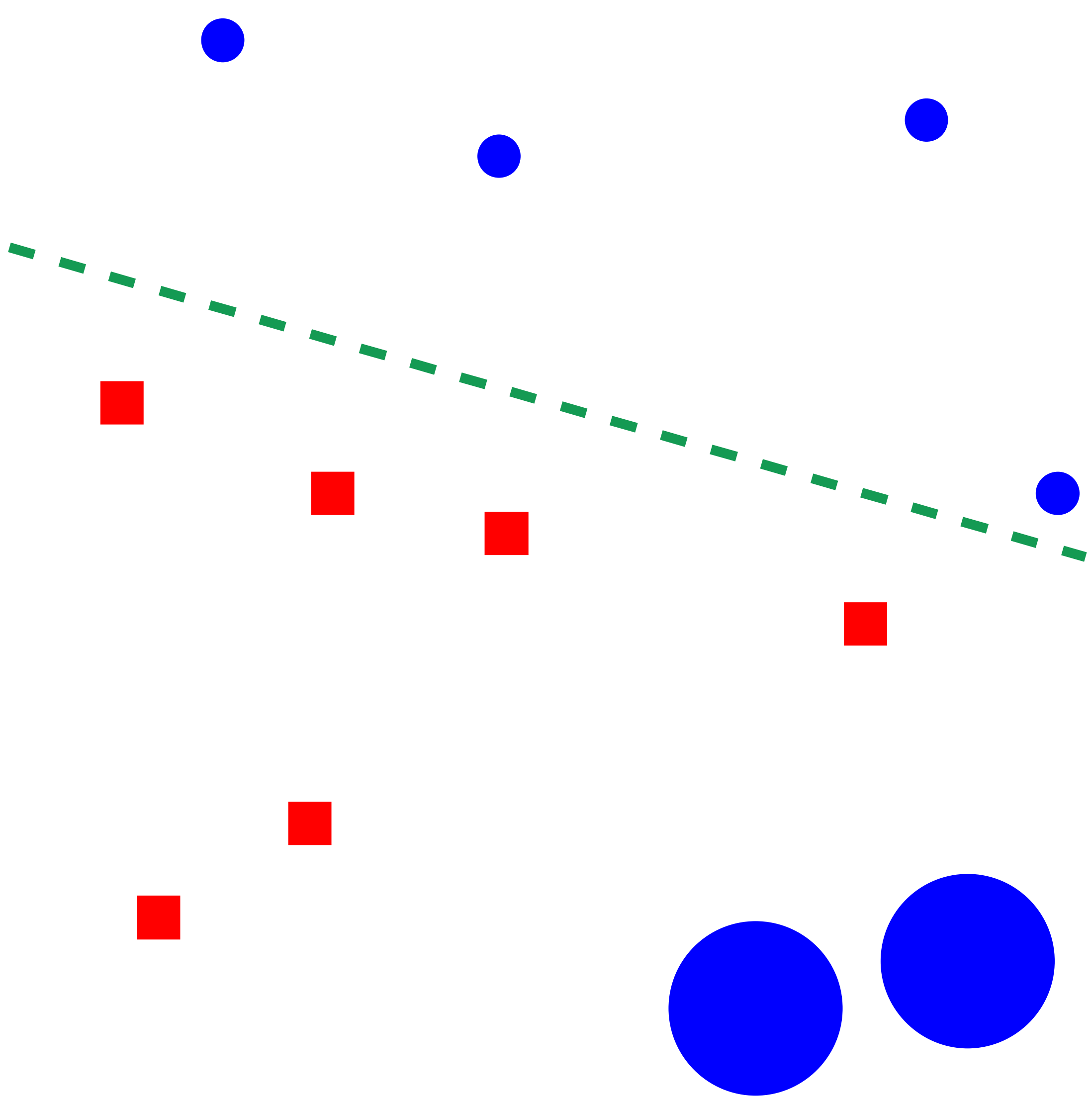
- **Pick a weak classifier h_m**
- Adjust weights: misclassified examples get “heavier”
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,\ldots,M$

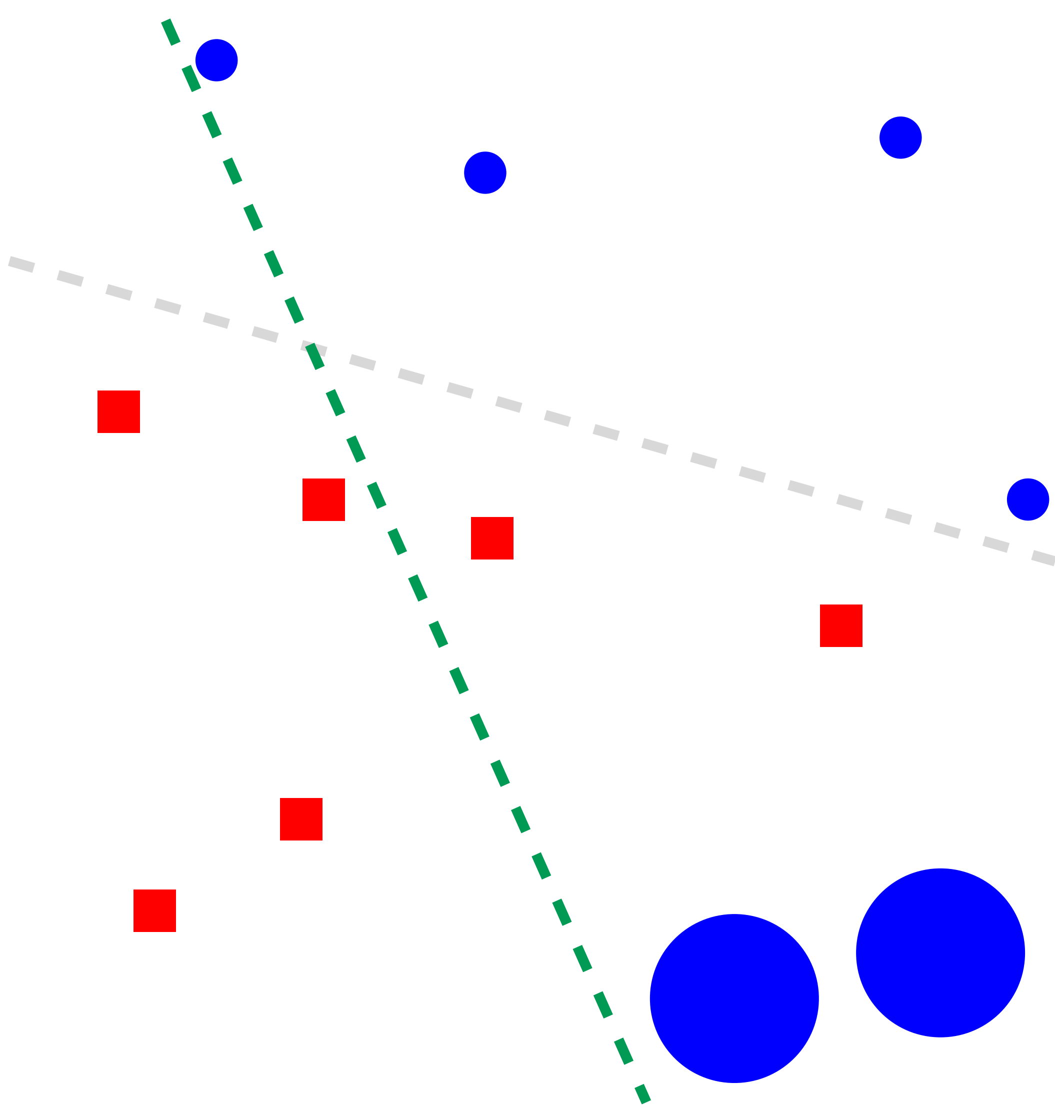
- Pick a weak classifier h_m
- **Adjust weights: misclassified examples get "heavier"**
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,...,M$

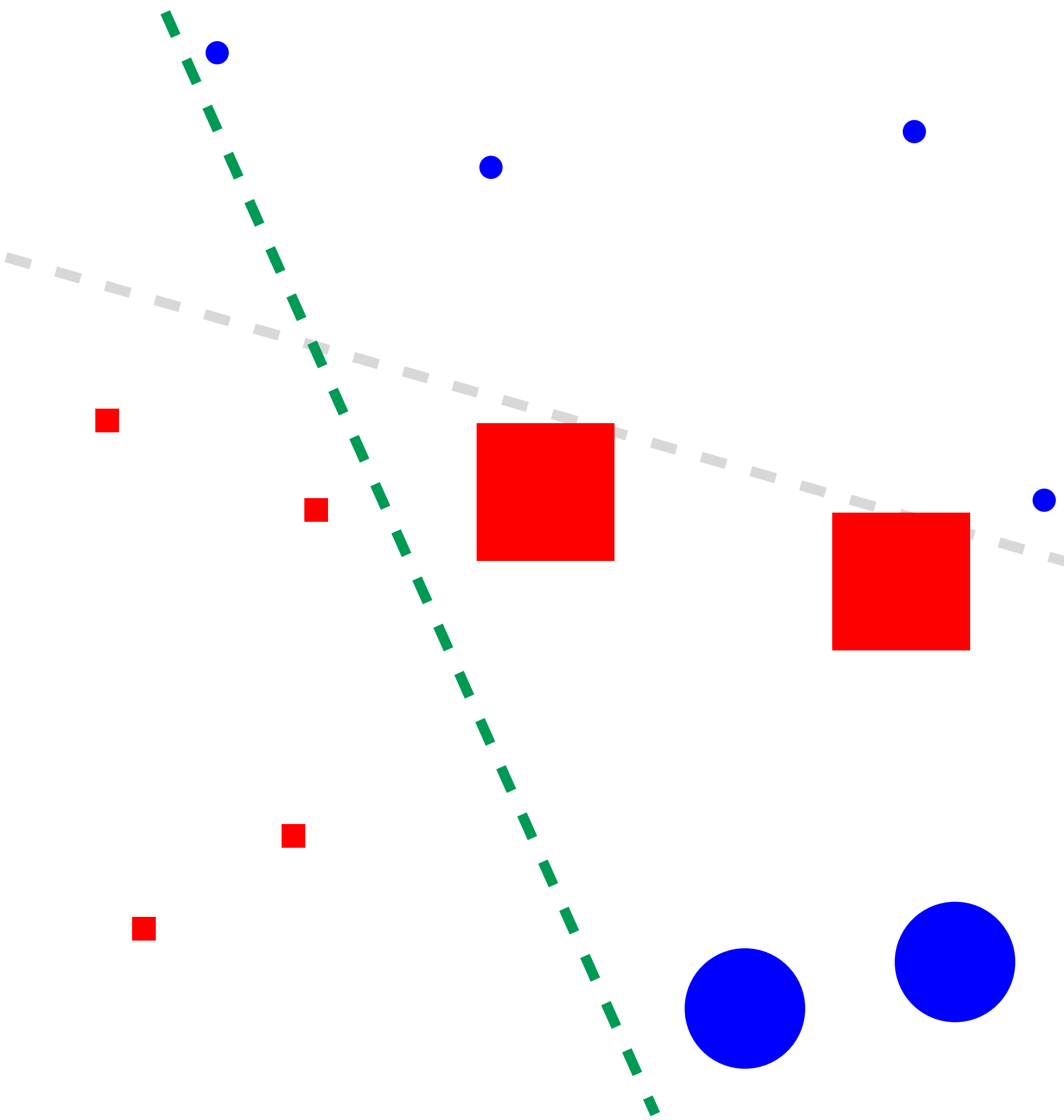
- **Pick a weak classifier h_m**
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for m=1,...,M

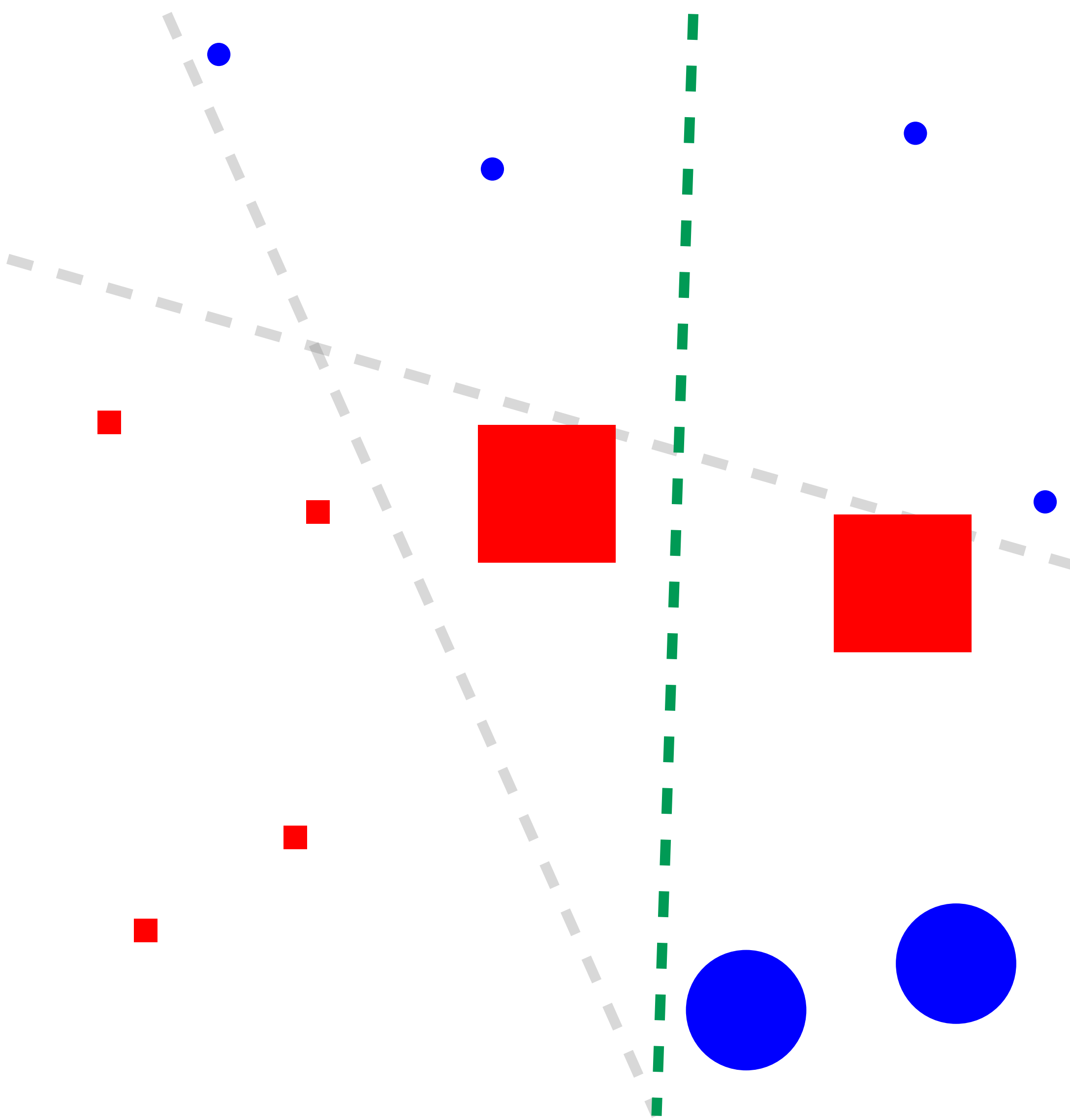
- Pick a weak classifier h_m
- **Adjust weights: misclassified examples get "heavier"**
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,\ldots,M$

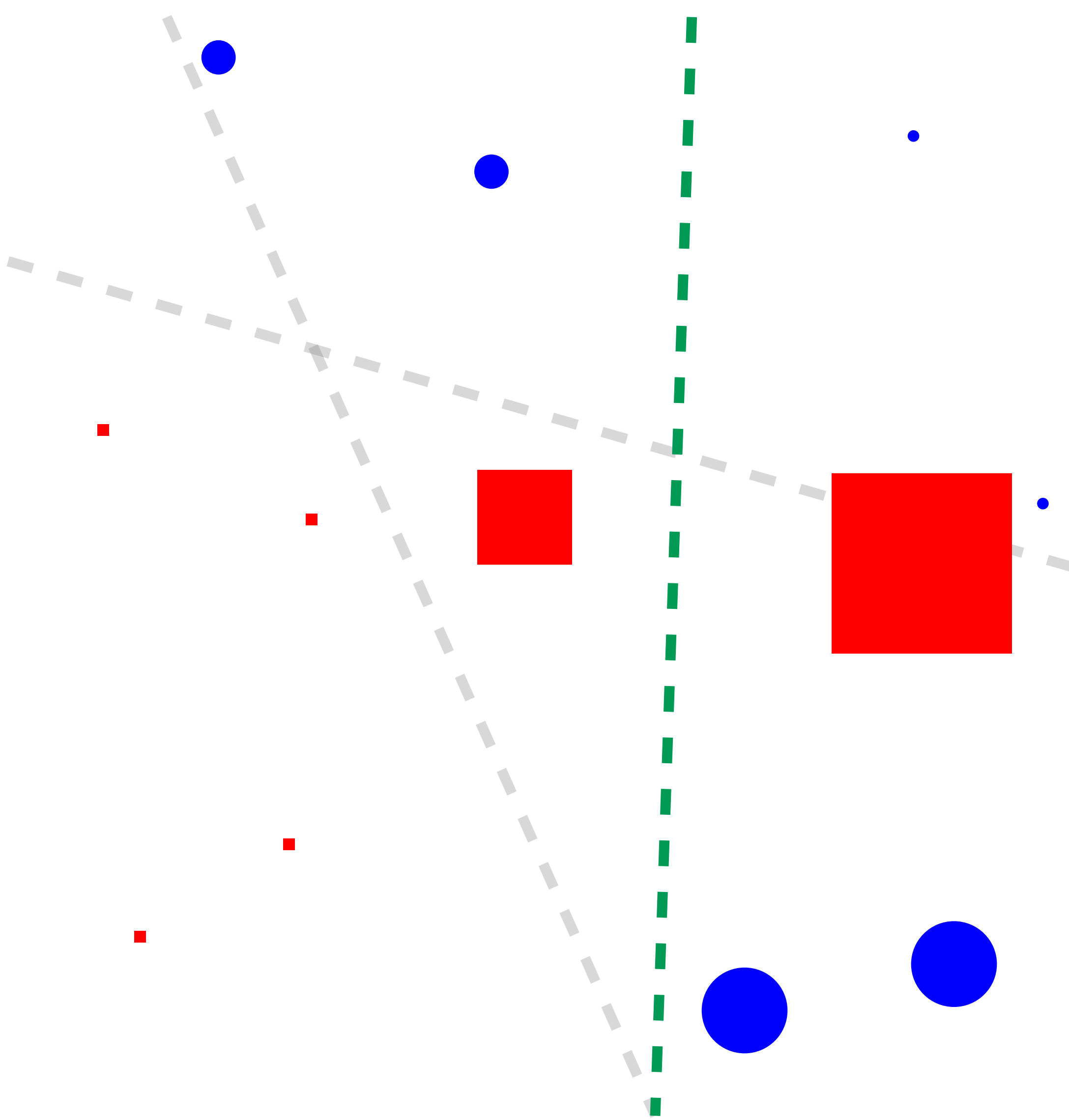
- **Pick a weak classifier h_m**
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,\ldots,M$

- Pick a weak classifier h_m
- **Adjust weights: misclassified examples get "heavier"**
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

First Boosting Algorithms

- [Schapire ’89]:
  - first provable boosting algorithm
- [Freund ’90]:
  - “optimal” algorithm that “boosts by majority”
- [Drucker, Schapire & Simard ’92]:
  - first experiments using boosting
  - limited by practical drawbacks
- [Freund & Schapire ’95]:
  - introduced “**AdaBoost**” algorithm
  - strong practical advantages over previous boosting algorithms

slide by Rob Schapire

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

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

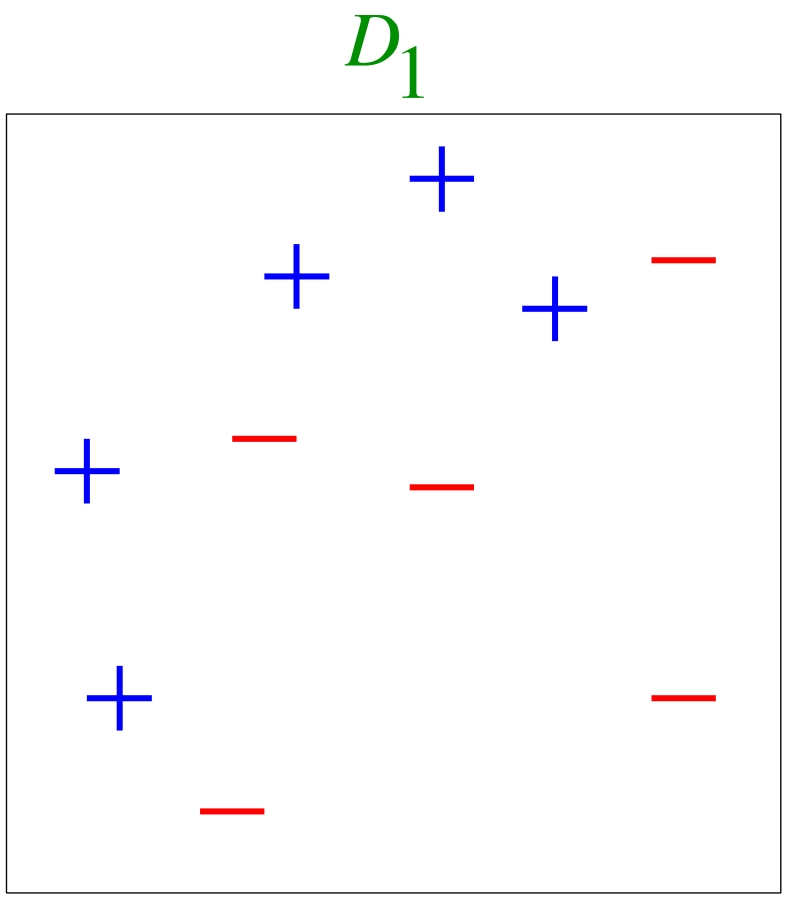
$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Toy Example



Minimize the error

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

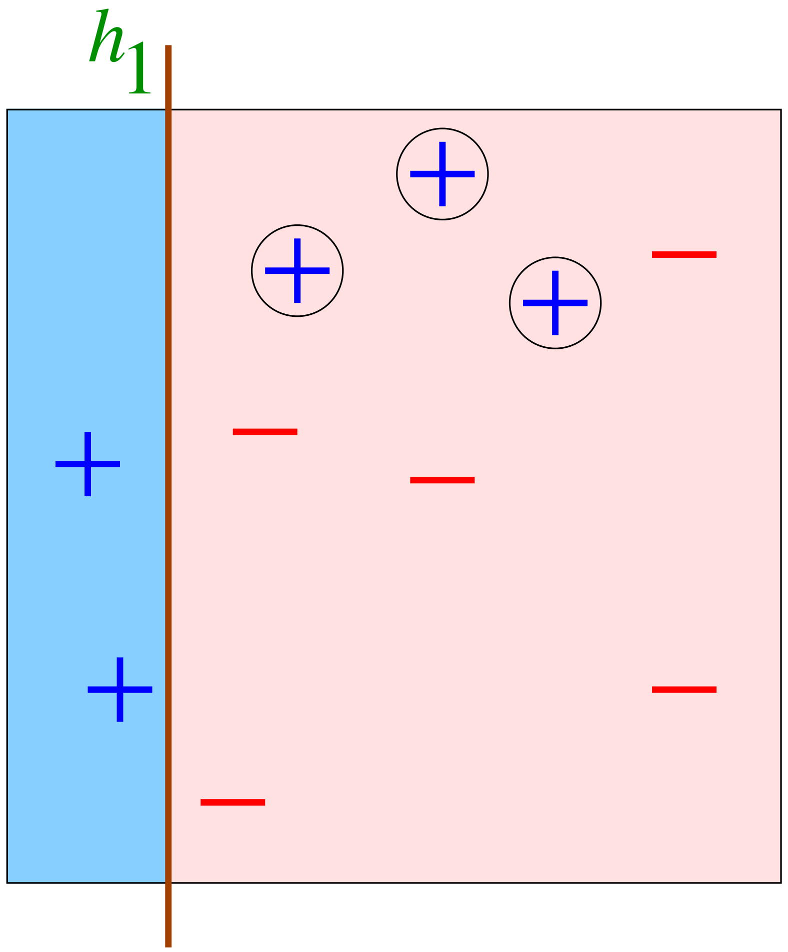
For binary h_t , typically use

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

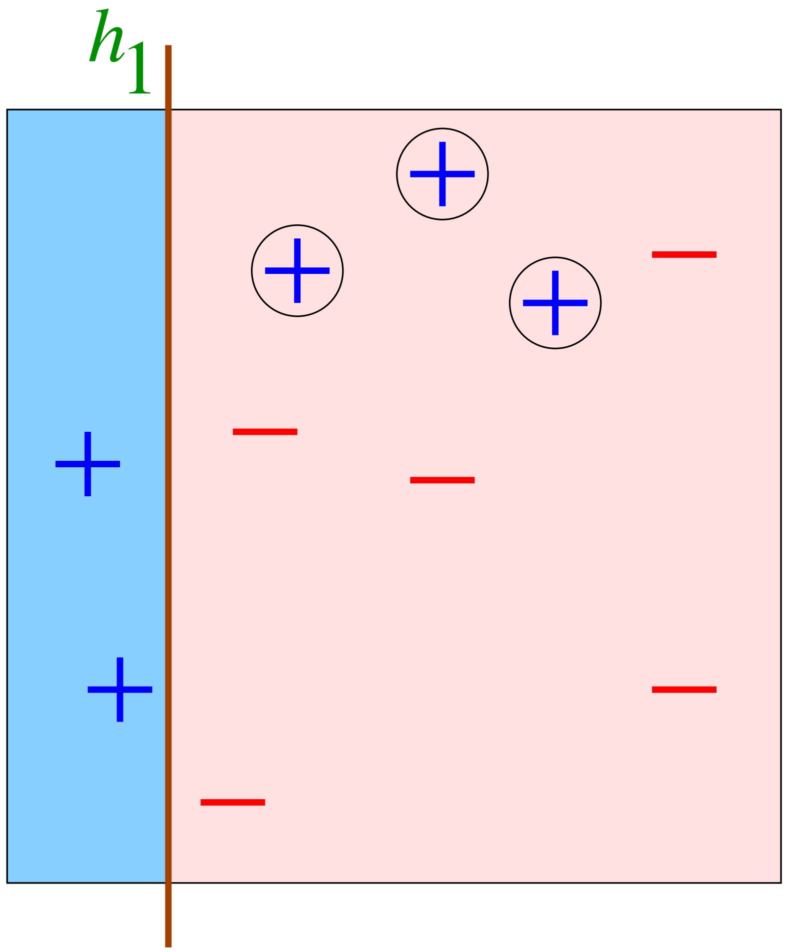
slide by Rob Schapire

weak hypotheses = vertical or horizontal half-planes

Round 1



Round 1

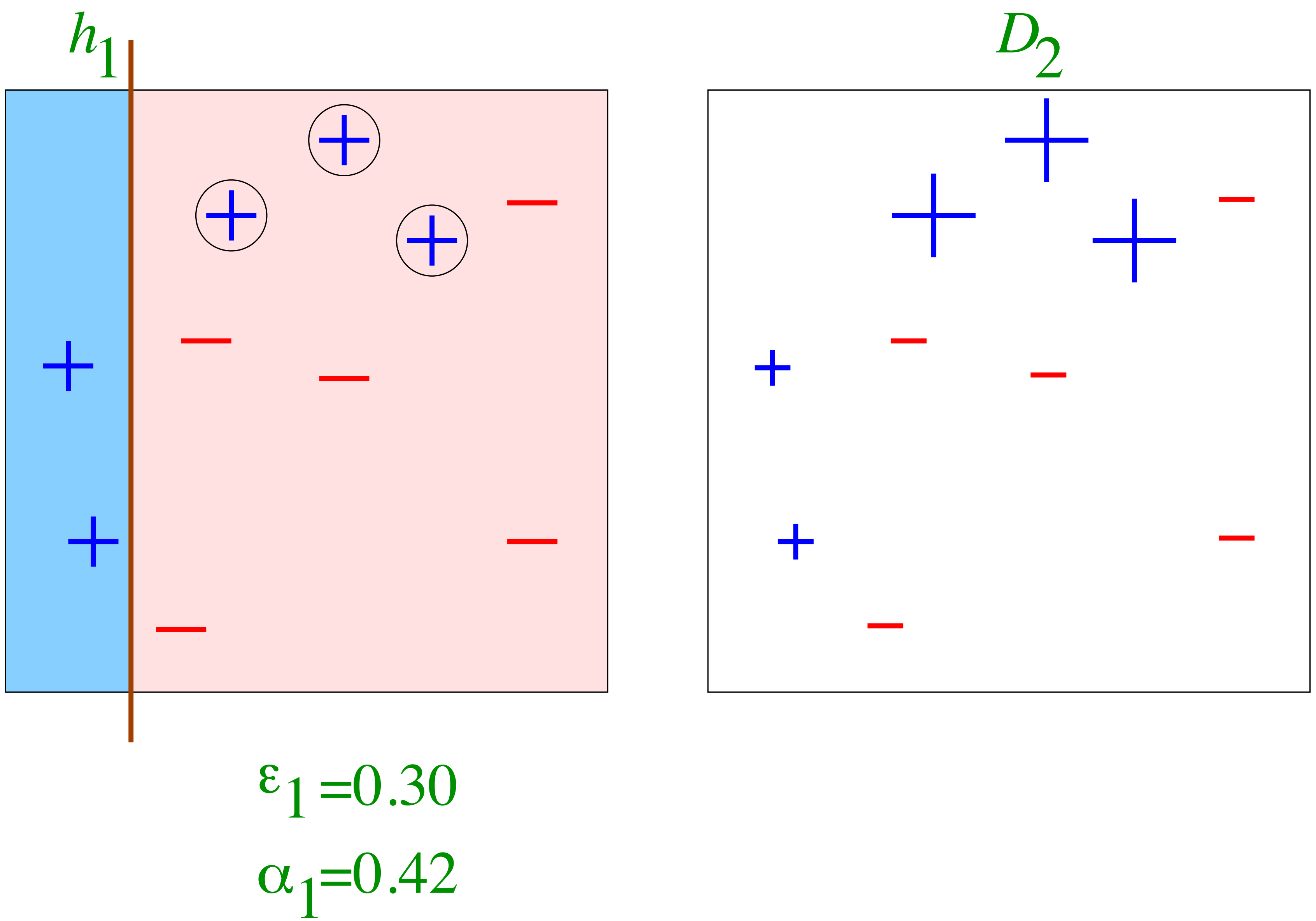


$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

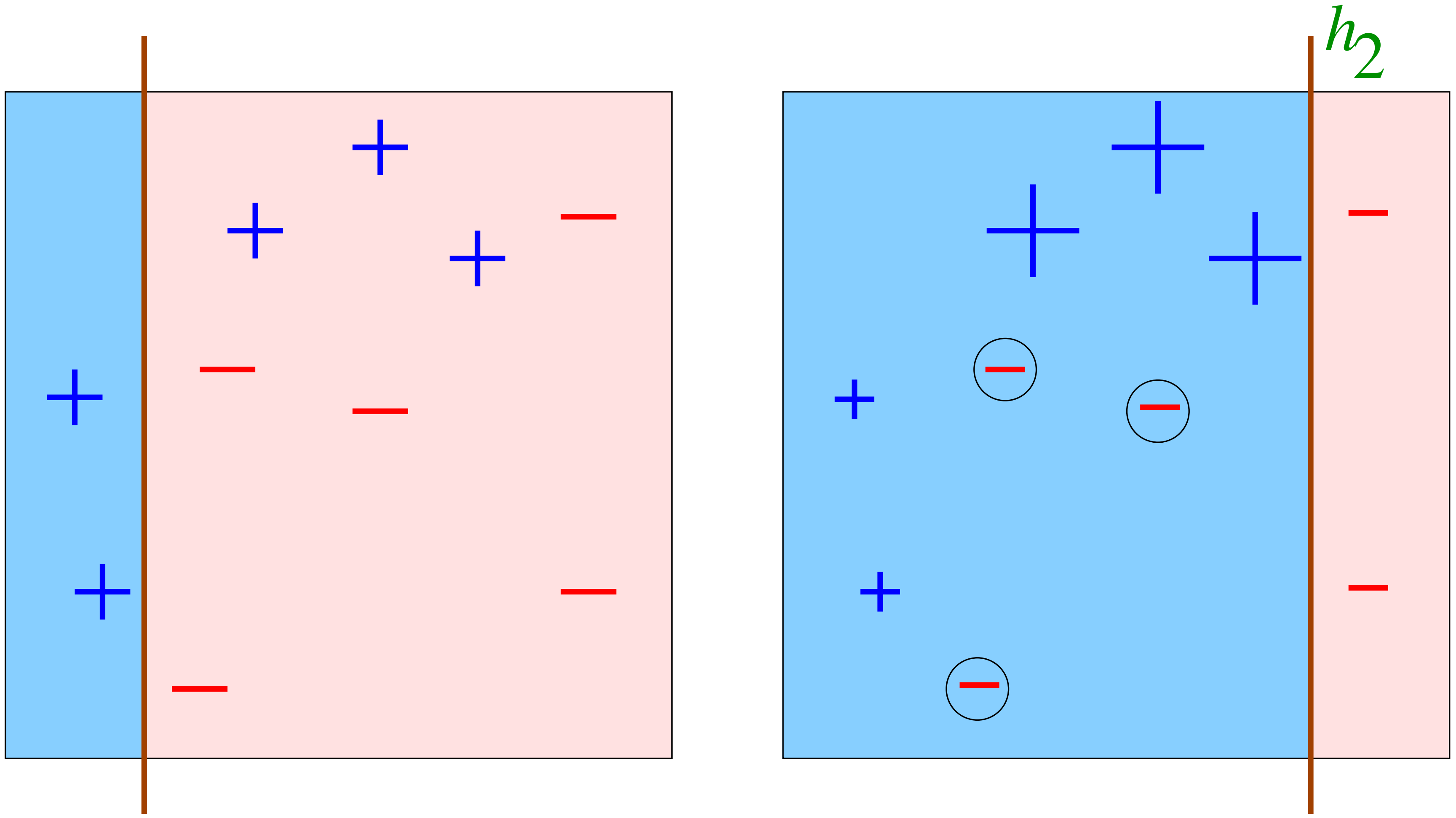
slide by Rob Schapire

Round 1



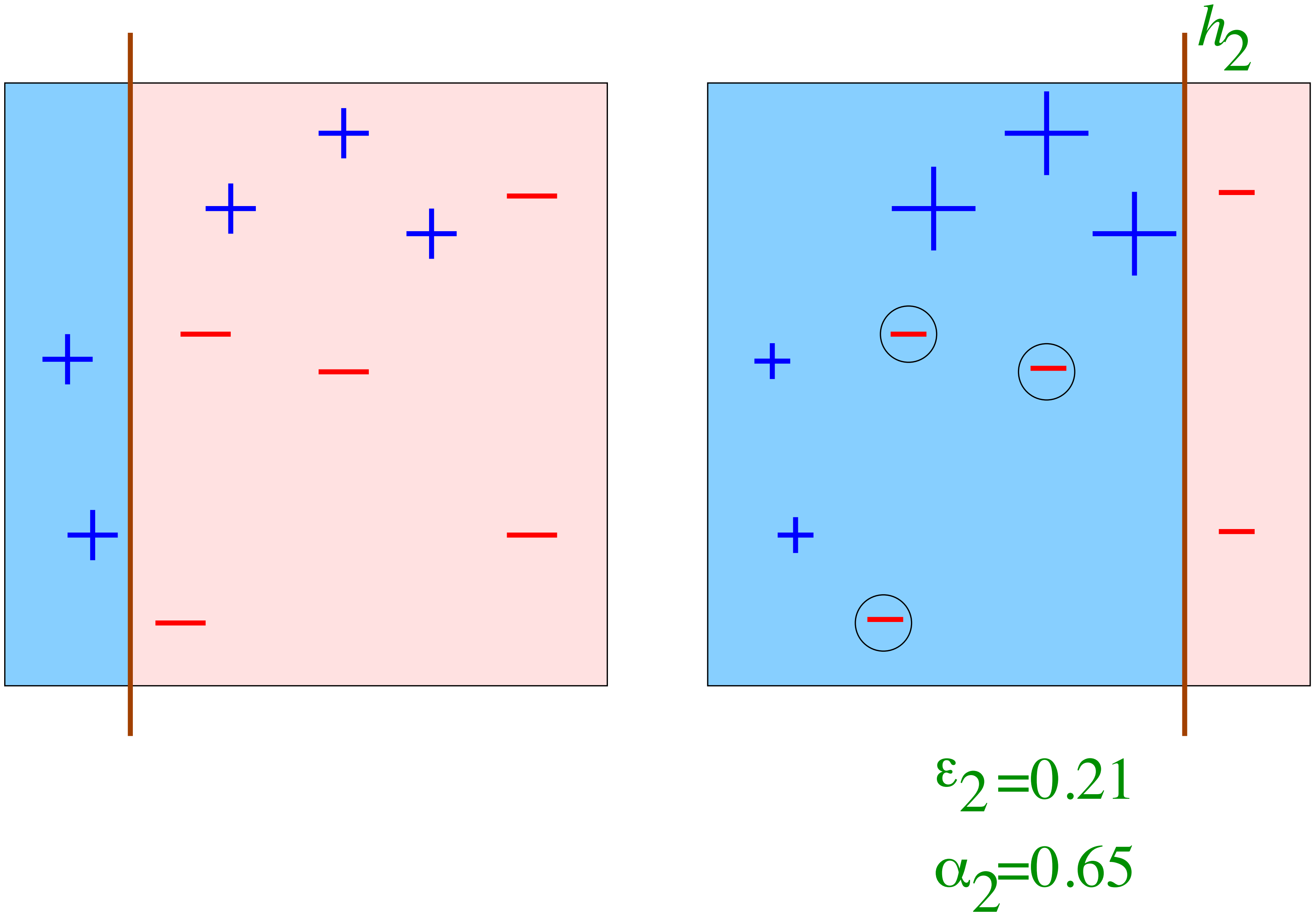
slide by Rob Schapire

Round 2



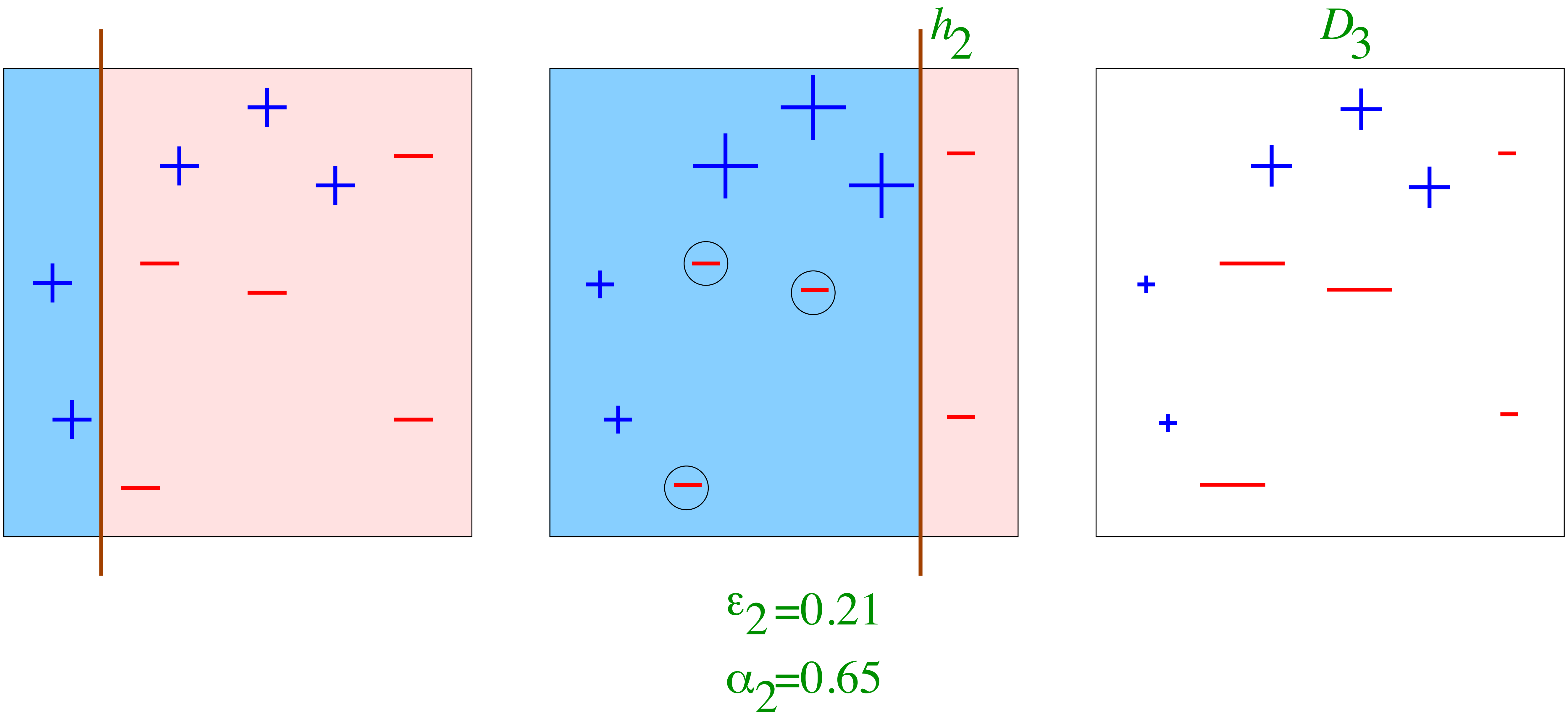
slide by Rob Schapire

Round 2



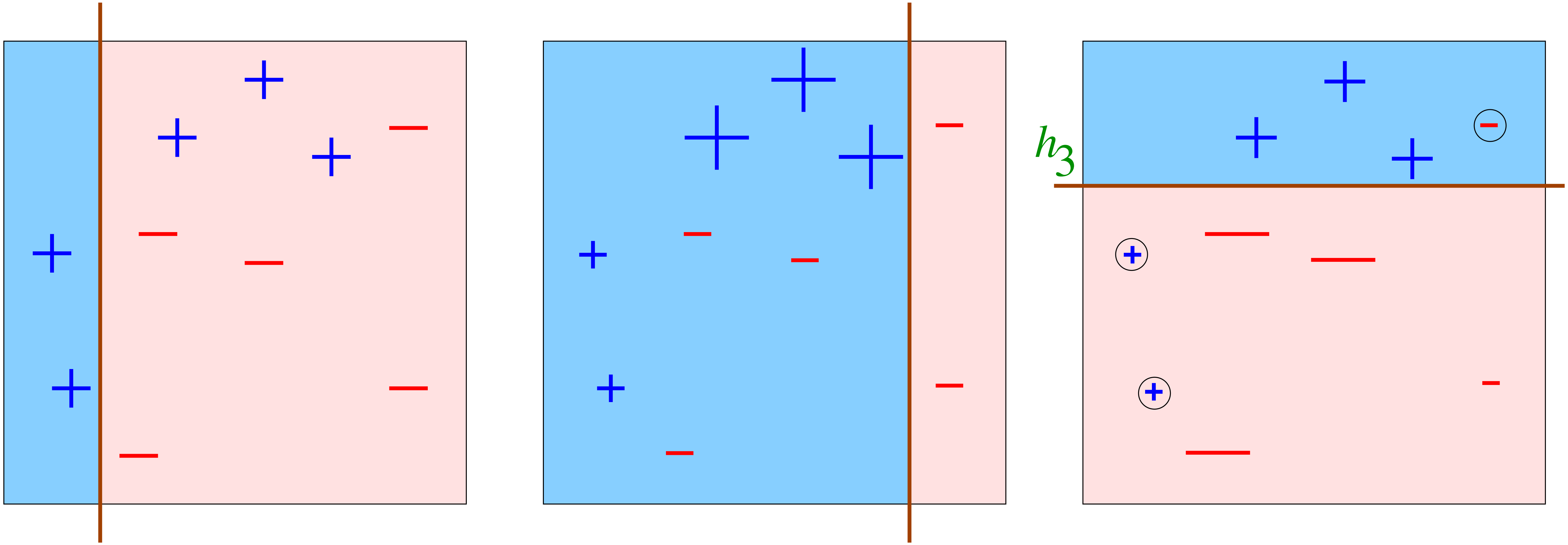
slide by Rob Schapire

Round 2

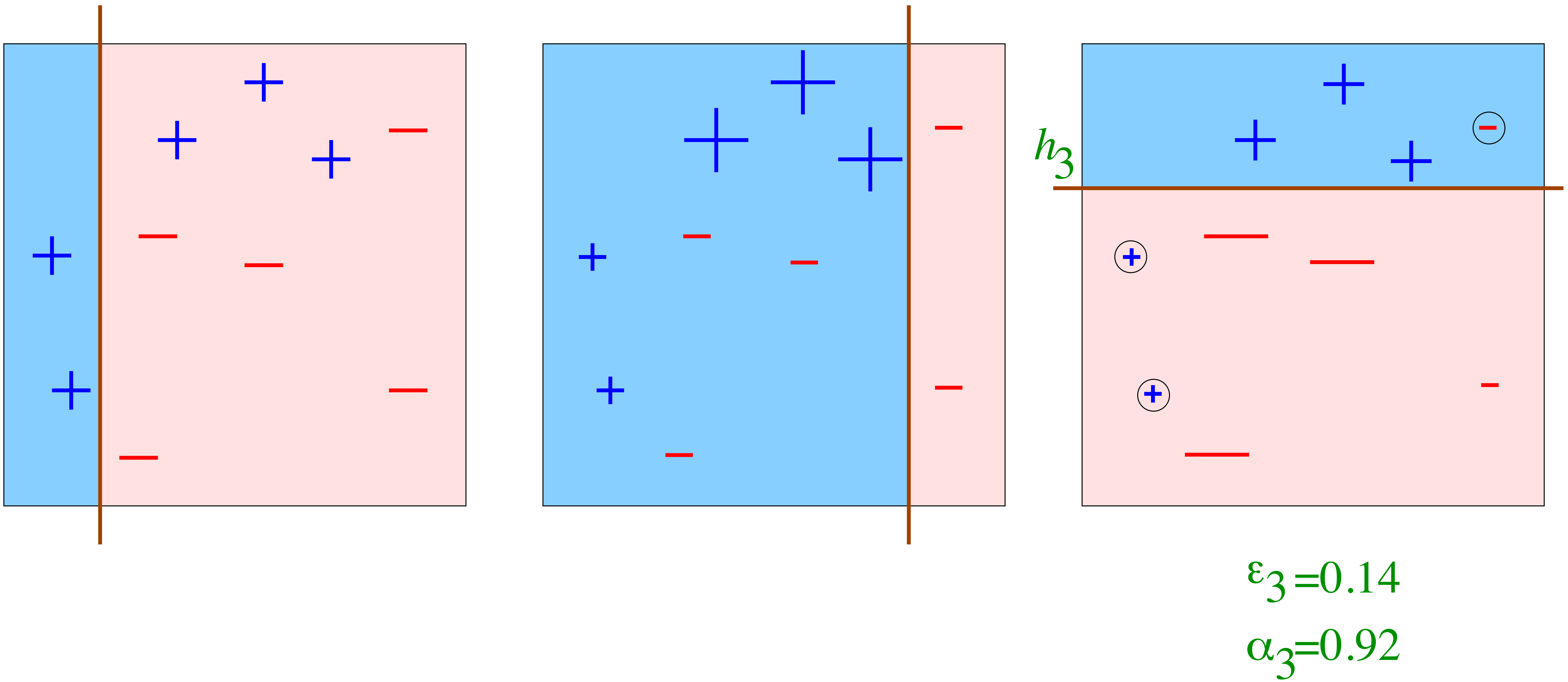


slide by Rob Schapire

Round 3

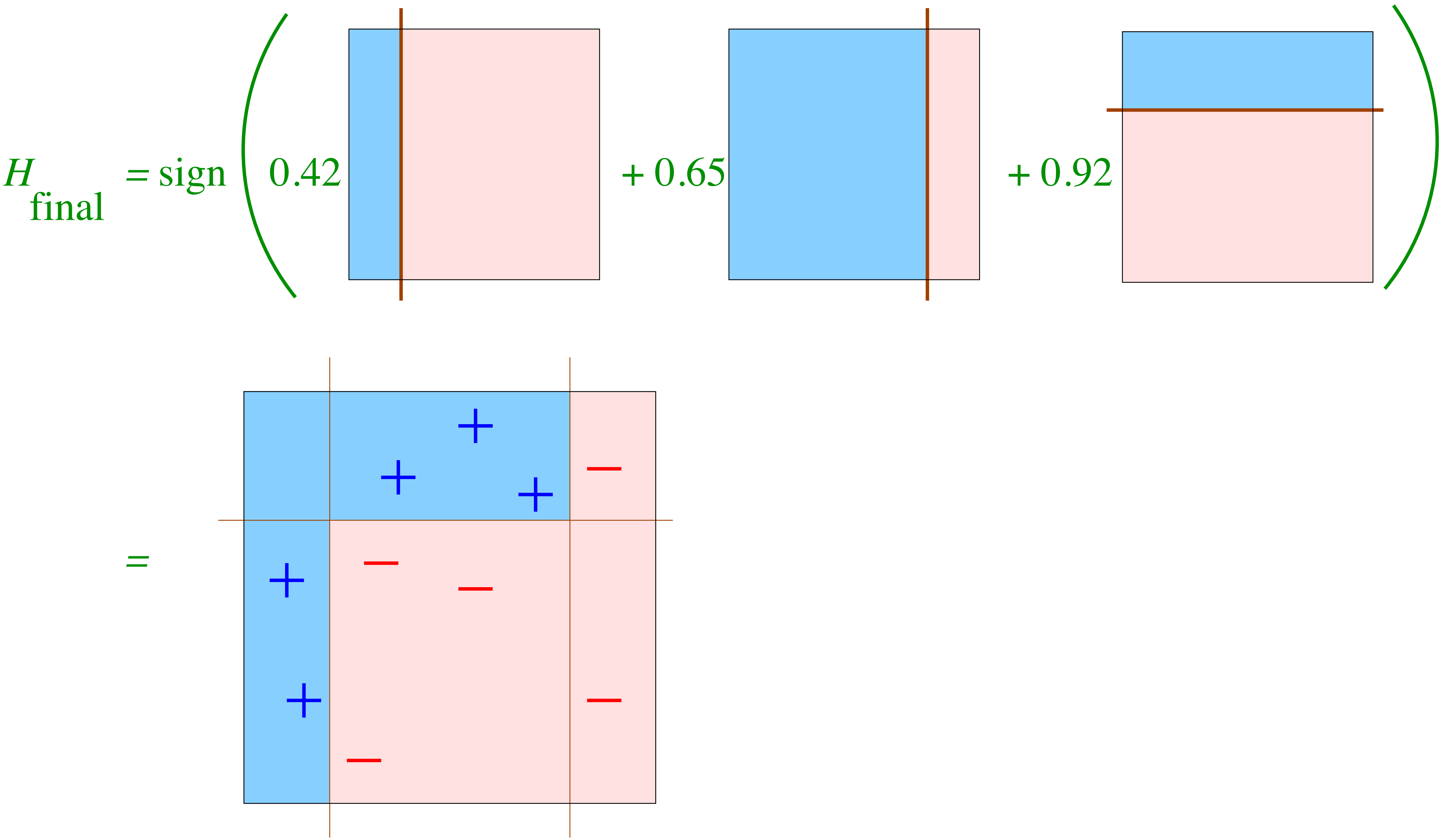


Round 3



slide by Rob Schapire

Final Hypothesis



slide by Rob Schapire

Voted combination of classifiers

- The general problem here is to try to combine many simple “weak” classifiers into a single “strong” classifier
- We consider voted combinations of simple binary ± 1 component classifiers

$$h_m(\mathbf{x}) = \alpha_1 h(\mathbf{x}; \theta_1) + \ldots + \alpha_m h(\mathbf{x}; \theta_m)$$

where the (non-negative) votes α_i can be used to emphasize component classifiers that are more reliable than others

slide by Tommi S. Jaakkola

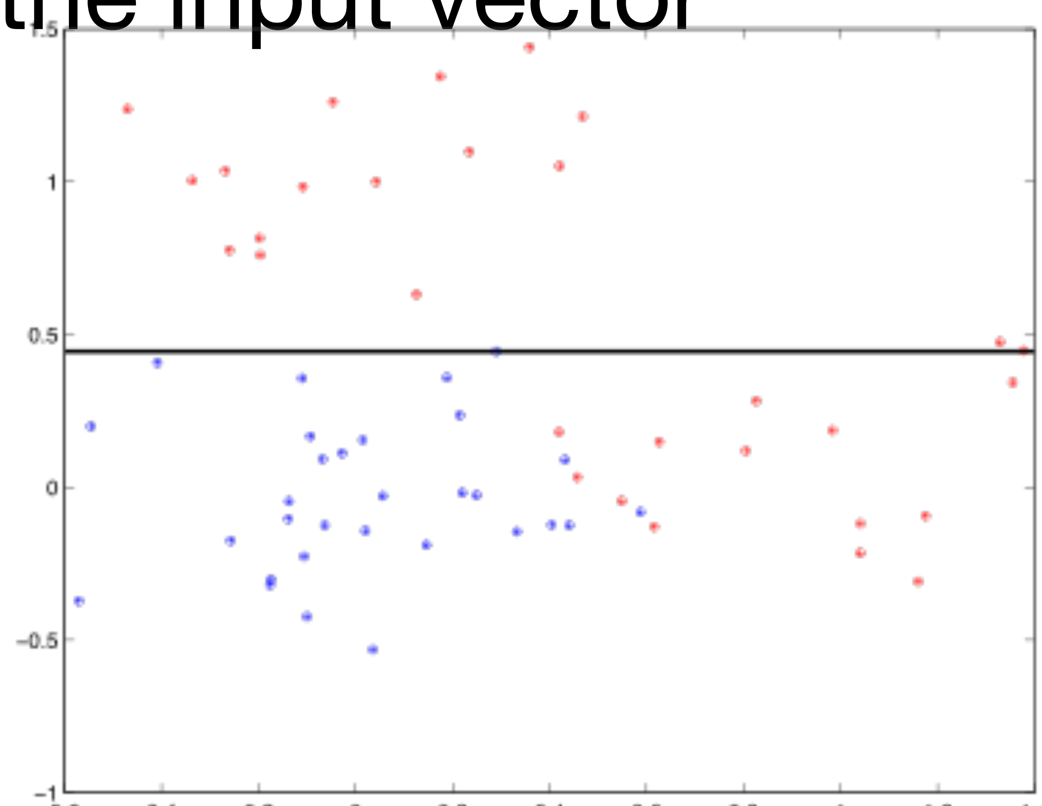
Components: Decision stumps

- Consider the following simple family of component classifiers generating ± 1 labels:

$$h(\mathbf{x};\theta) = \text{sign}(\, w_1 \, x_k - w_0 \,)$$

where $\theta = \{k, w_1, w_0\}$. These are called decision stumps.

- Each decision stump pays attention to only a single component of the input vector



slide by Tommi S. Jaakkola

Voted combinations (cont'd.)

- We need to define a loss function for the combination so we can determine which new component $h(\mathrm{x};\theta)$ to add and how many votes it should receive

$$h_m(\mathbf{x}) = \alpha_1 h(\mathbf{x};\theta_1) + \ldots + \alpha_m h(\mathbf{x};\theta_m)$$

- While there are many options for the loss function we consider here only a simple exponential loss

$$\sum_{i=1}^{n} \exp\{ -y\, h_m(\mathbf{x}) \}$$

slide by Tommi S. Jaakkola

Modularity, errors, and loss

- Consider adding the m^{th} component:

$$\sum_{i=1}^{n} \exp\{ -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)] \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i;\theta_m) \}$$

slide by Tommi S. Jaakkola

Modularity, errors, and loss

- Consider adding the m^{th} component:

$$\sum_{i=1}^{n} \exp\{ -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)] \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i;\theta_m) \}$$

$$= \sum_{i=1}^{n} \underbrace{\exp\{ -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m) \}$$

slide by Tommi S. Jaakkola

Modularity, errors, and loss

- Consider adding the m^{th} component:

$$\sum_{i=1}^{n} \exp\{ -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i; \theta_m)] \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i; \theta_m) \}$$

$$= \sum_{i=1}^{n} \underbrace{\exp\{ -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i; \theta_m) \}$$

- So at the m^{th} iteration the new component (and the votes) should optimize a weighted loss (weighted towards mistakes).

slide by Tommi S. Jaakkola

Empirical exponential loss (cont'd.)

- To increase modularity we'd like to further decouple the optimization of $h(\mathrm{x};\theta_m)$ from the associated votes α_m
- To this end we select $h(\mathrm{x};\theta_m)$ that optimizes the rate at which the loss would decrease as a function of α_m

$$\frac{\partial}{\partial \alpha_m}\Big|_{\alpha_m=0} \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\} =$$

$$\left[\sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\} \cdot \big(-y_i h(\mathbf{x}_i;\theta_m)\big)\right]_{\alpha_m=0}$$

$$= \left[\sum_{i=1}^{n} W_i^{(m-1)} \big(-y_i h(\mathbf{x}_i;\theta_m)\big)\right]$$

slide by Tommi S. Jaakkola

Empirical exponential loss (cont'd.)

- We find $h(\mathbf{x};\hat{\theta}_m)$ that minimizes

$$-\sum_{i=1}^{n} W_i^{(m-1)} y_i h(\mathbf{x}_i;\theta_m)$$

- We can also normalize the weights:

$$-\sum_{i=1}^{n} \frac{W_i^{(m-1)}}{\sum_{j=1}^{n} W_j^{(m-1)}} y_i h(\mathbf{x}_i;\theta_m)$$

$$= -\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} y_i h(\mathbf{x}_i;\theta_m)$$

so that $\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} = 1.$

slide by Tommi S. Jaakkola

Empirical exponential loss (cont'd.)

- We find $h(\mathbf{x};\hat{\theta}_m)$ that minimizes

$$-\sum_{i=1}^{n} W_i^{(m-1)}\, y_i h(\mathbf{x}_i;\theta_m)$$

  where $\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} = 1.$

- α_m is subsequently chosen to minimize

$$\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} \exp\{-y_i\alpha_m h(\mathbf{x}_i;\hat{\theta}_m)\,\}$$

slide by Tommi S. Jaakkola

The AdaBoost Algorithm

0) Set $\tilde{W}_i^{(0)} = 1/n$ for $i = 1, \ldots, n$

1) At the m^{th} iteration we find (any) classifier $h(\mathbf{x}; \hat{\theta}_m)$ for which the *weighted classification error* ϵ_m

$$\epsilon_m = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} y_i h(\mathbf{x}_i; \hat{\theta}_m)\right)$$

is better than chance.

2) The new component is assigned votes based on its error:

$$\hat{\alpha}_m = 0.5 \log(\,(1 - \epsilon_m)/\epsilon_m\,)$$

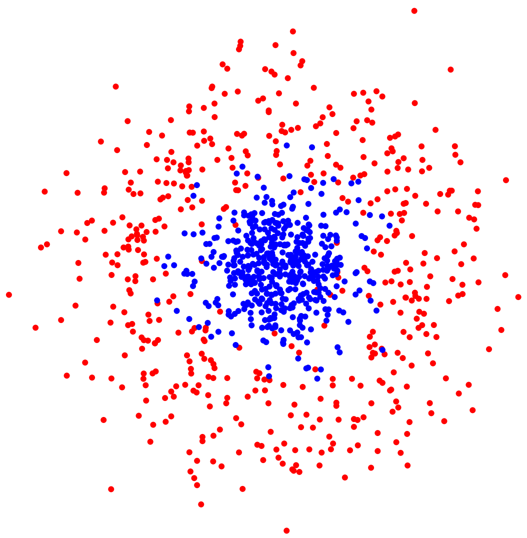
3) The weights are updated according to (Z_m is chosen so that the new weights $\tilde{W}_i^{(m)}$ sum to one):

$$\tilde{W}_i^{(m)} = \frac{1}{Z_m} \cdot \tilde{W}_i^{(m-1)} \cdot \exp\{\, -y_i \hat{\alpha}_m h(\mathbf{x}_i; \hat{\theta}_m)\,\}$$

slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

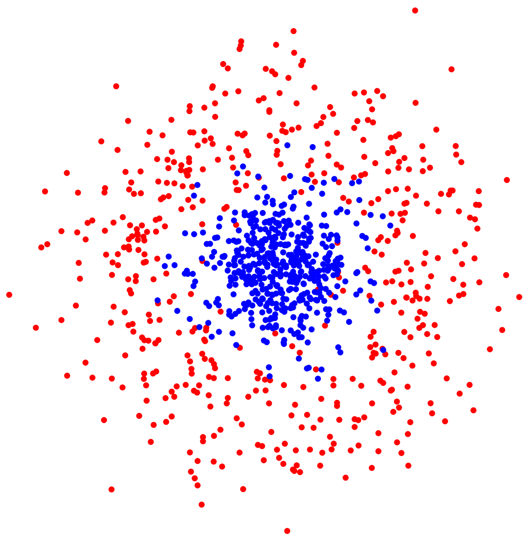


slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$



The AdaBoost Algorithm

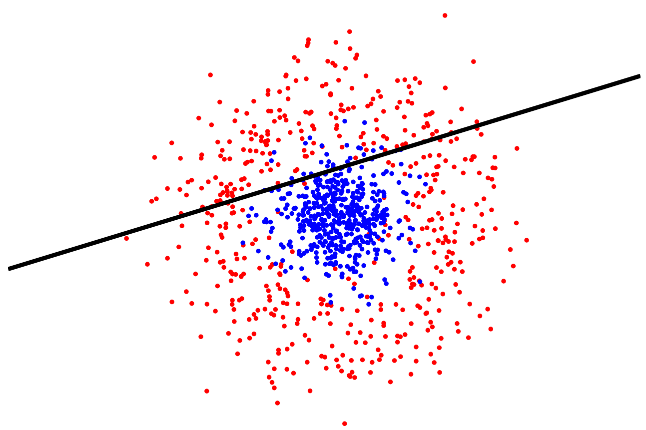
Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop

$t = 1$



slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

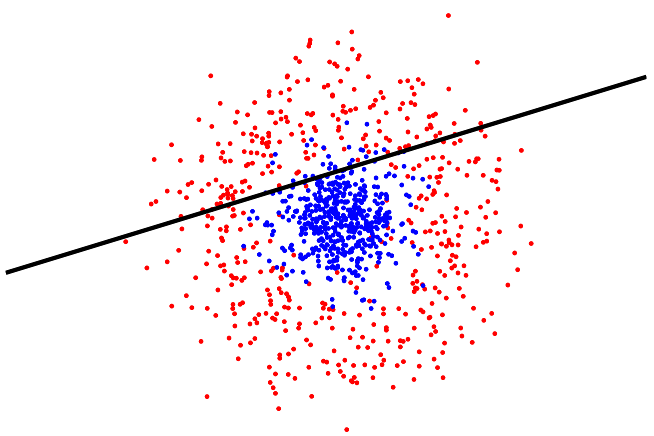
Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) [\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$

$t = 1$



slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

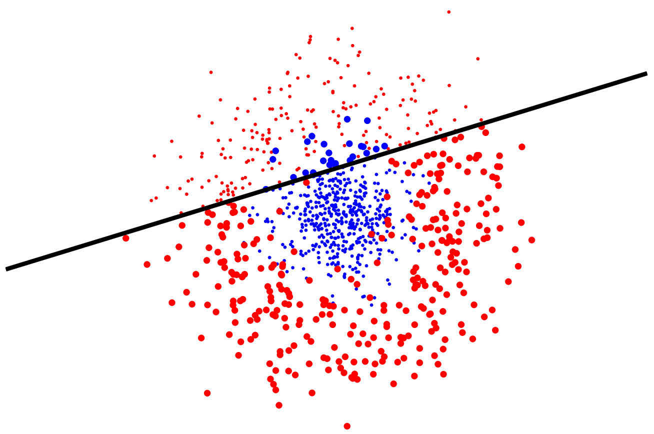
For $t = 1, \ldots, T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

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

where Z_t is normalisation factor

$t = 1$



slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

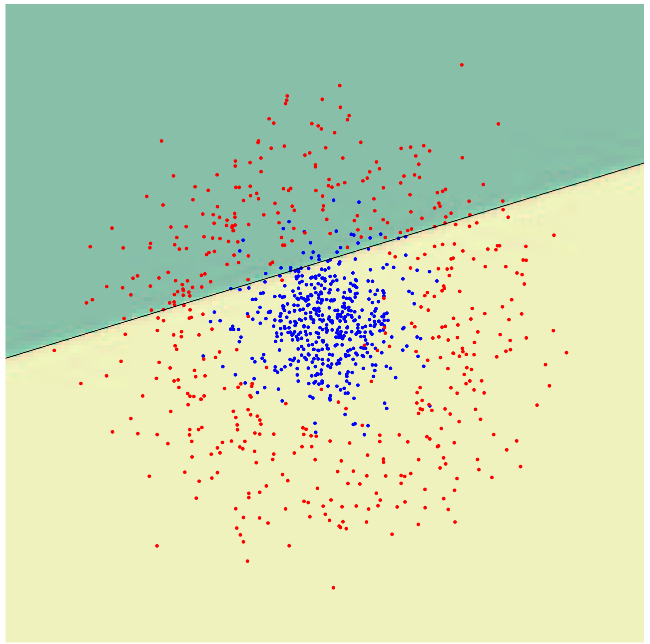
$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

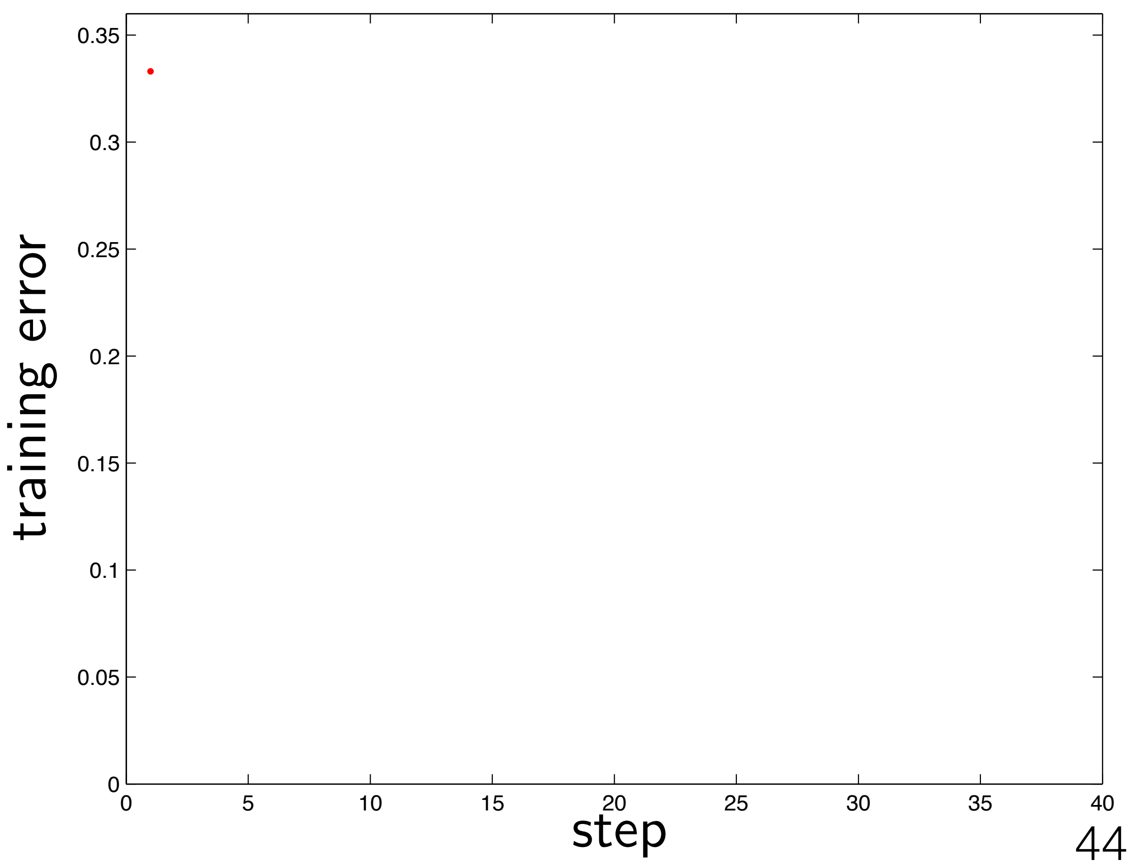
where Z_t is normalisation factor

Output the final classifier:

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

$t = 1$





slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) [\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

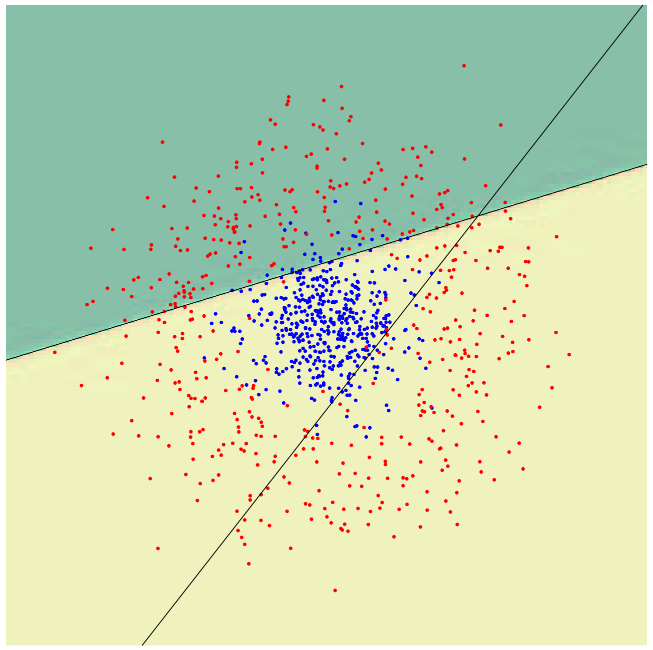
$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

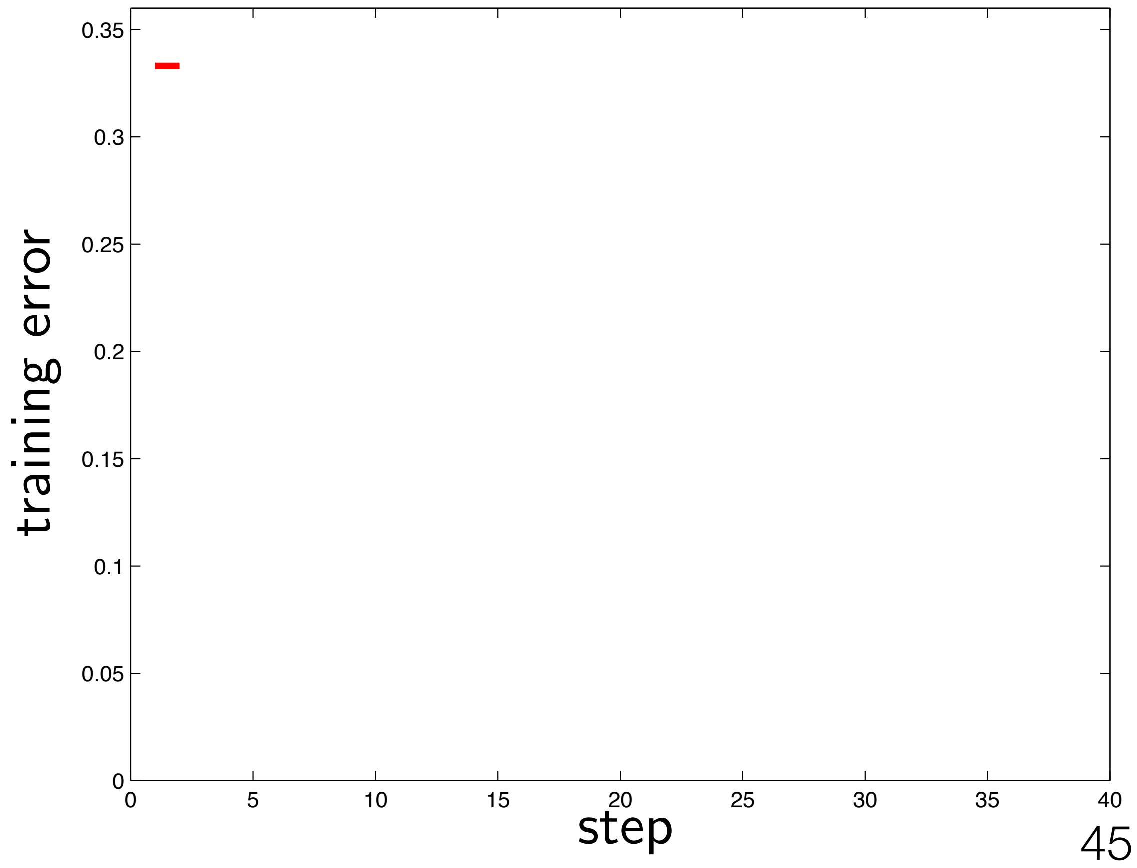
where Z_t is normalisation factor

Output the final classifier:

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

$t = 2$





slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

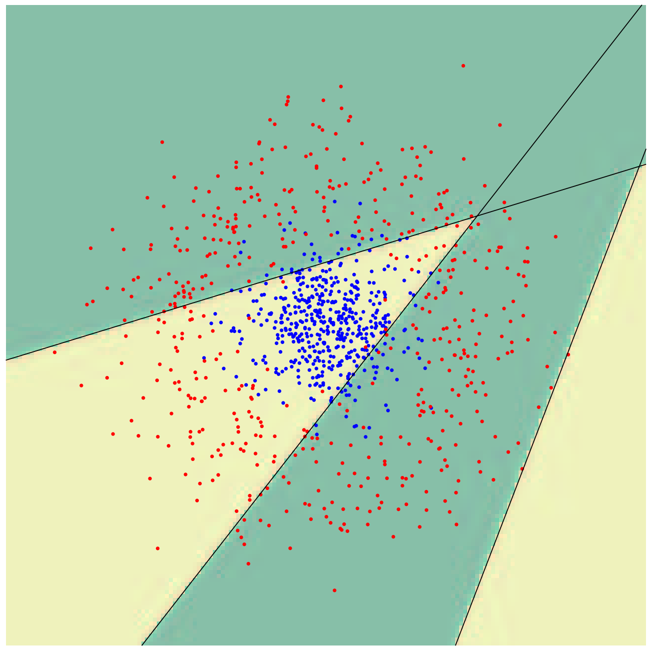
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

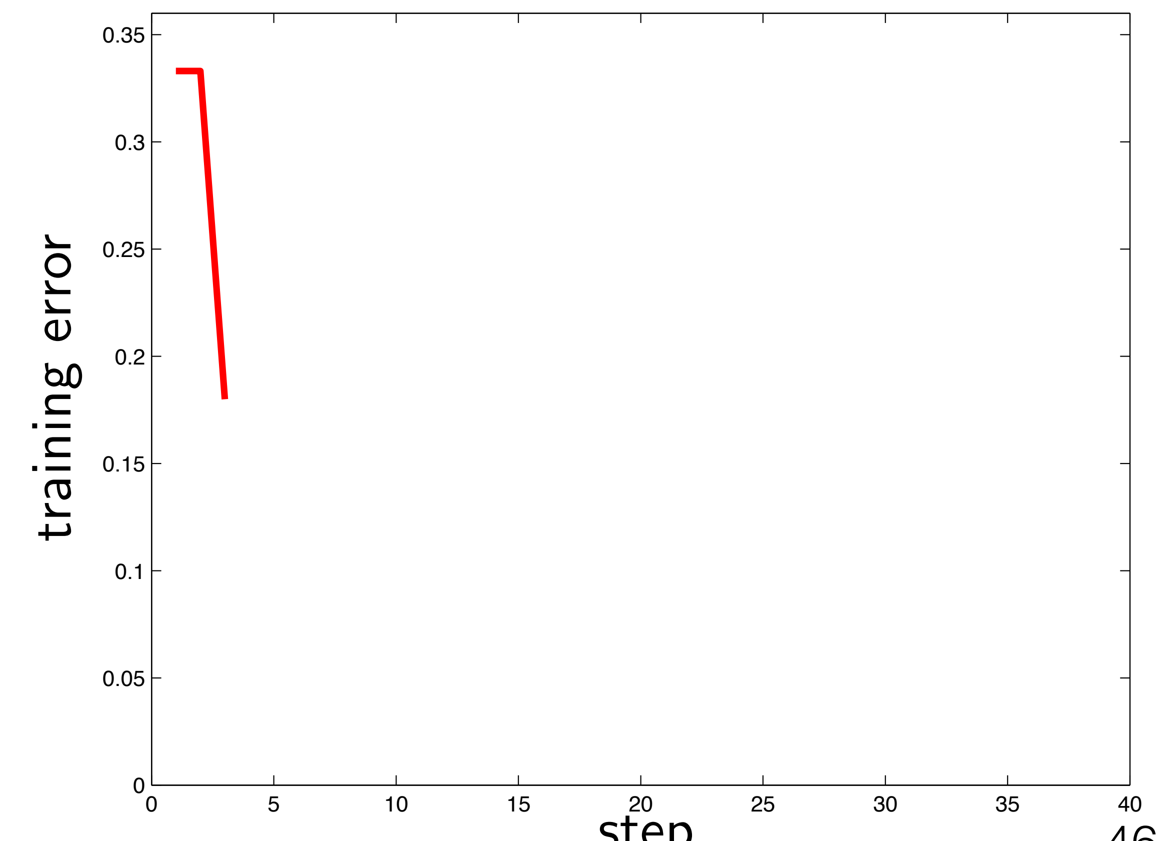
where Z_t is normalisation factor

Output the final classifier:

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

$t = 3$





slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) [\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

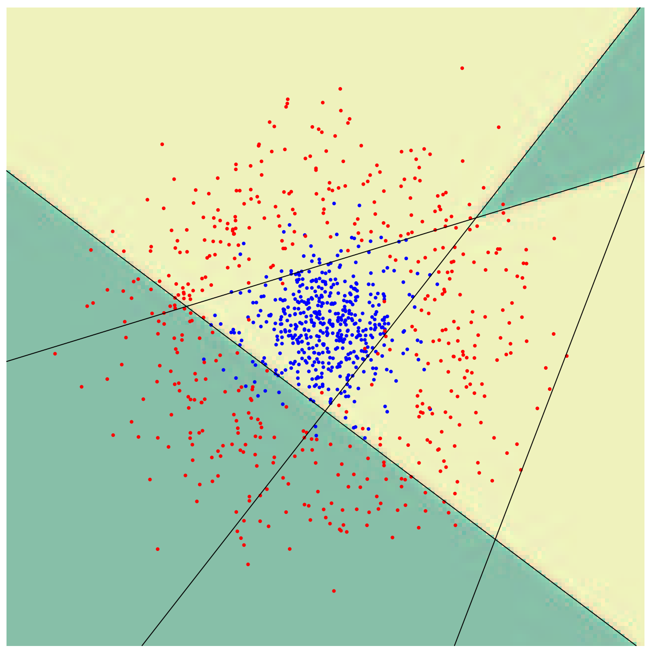
$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

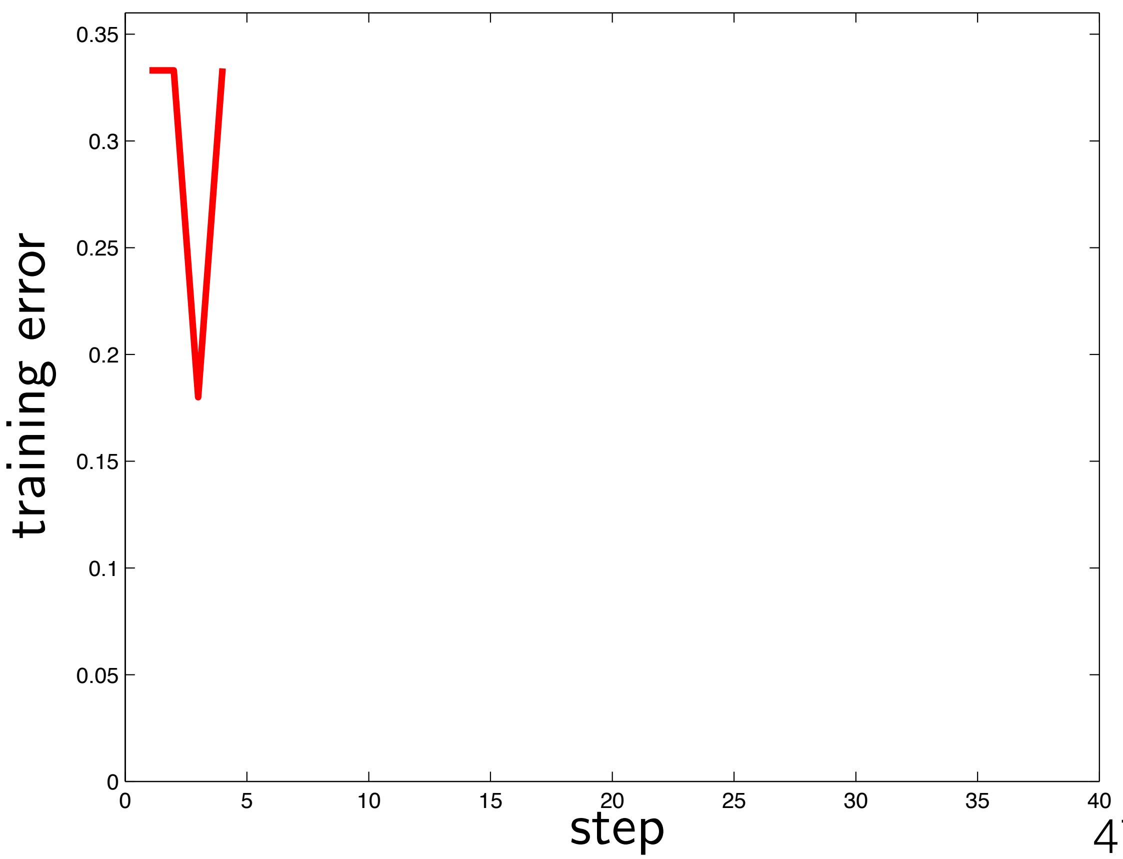
where Z_t is normalisation factor

Output the final classifier:

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

$t = 4$





slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

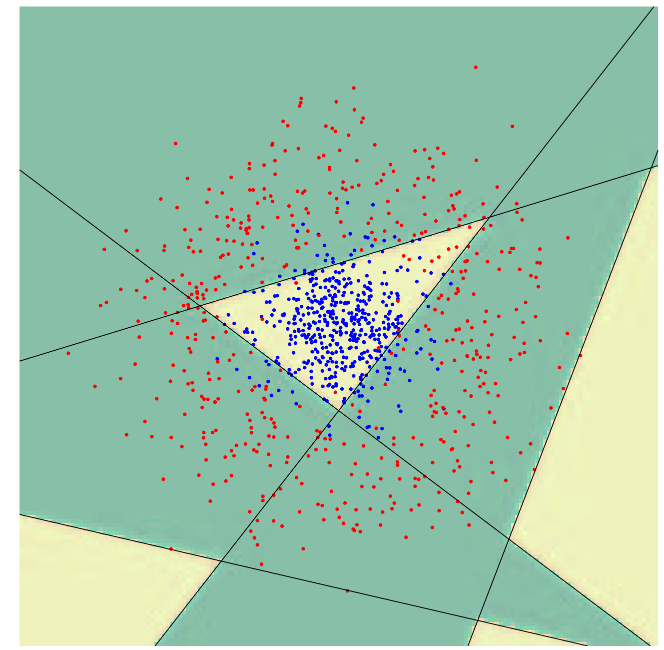
$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

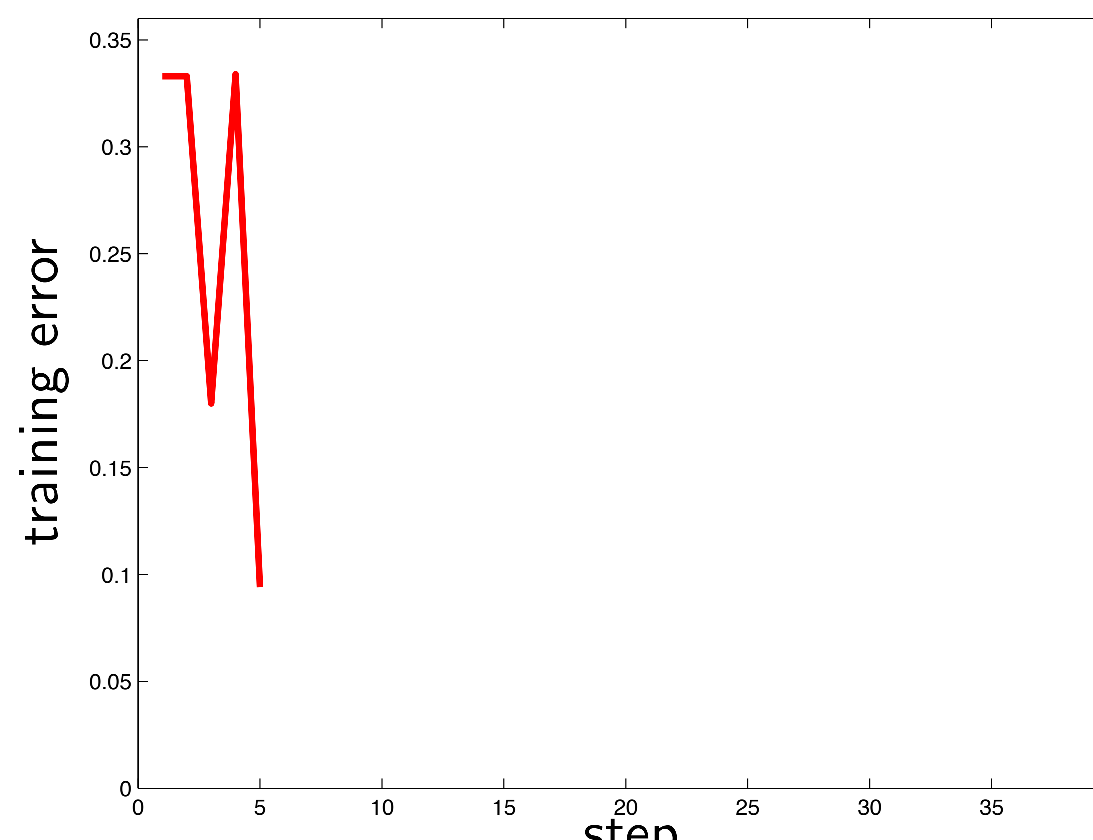
where Z_t is normalisation factor

Output the final classifier:

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

$t = 5$





slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) [\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

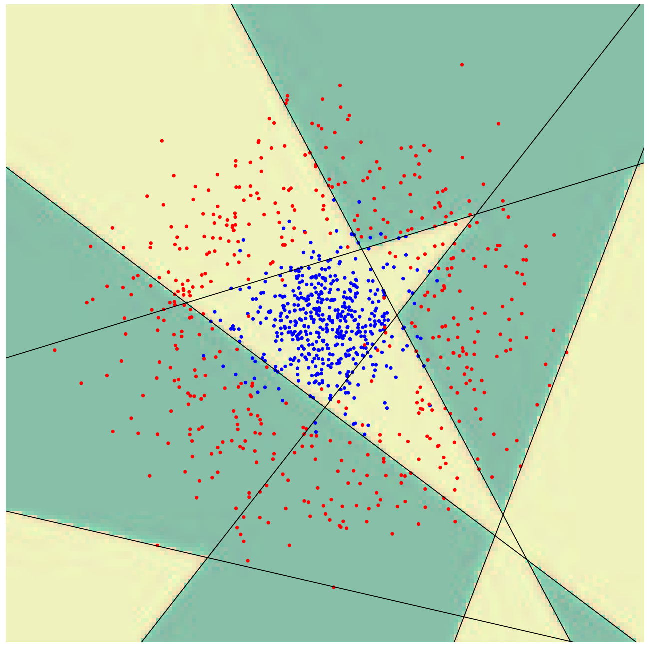
$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

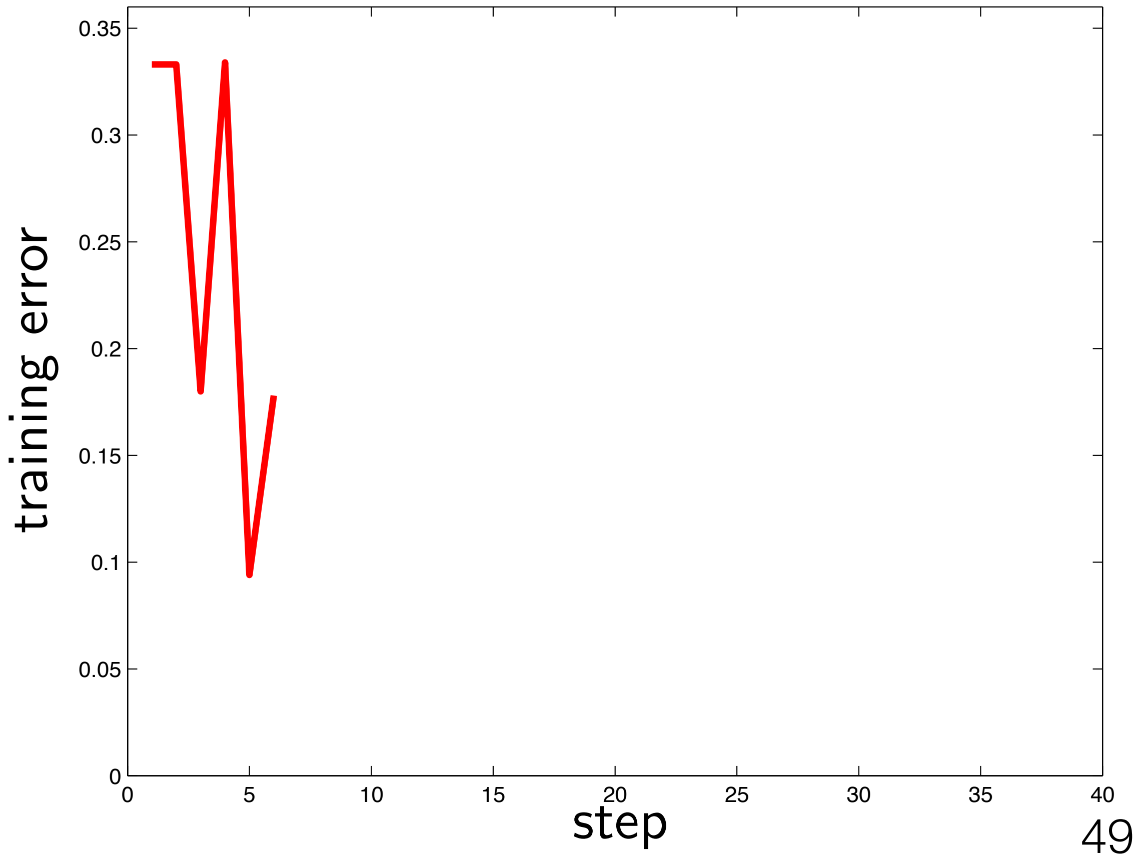
where Z_t is normalisation factor

Output the final classifier:

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

$t = 6$





slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) [\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

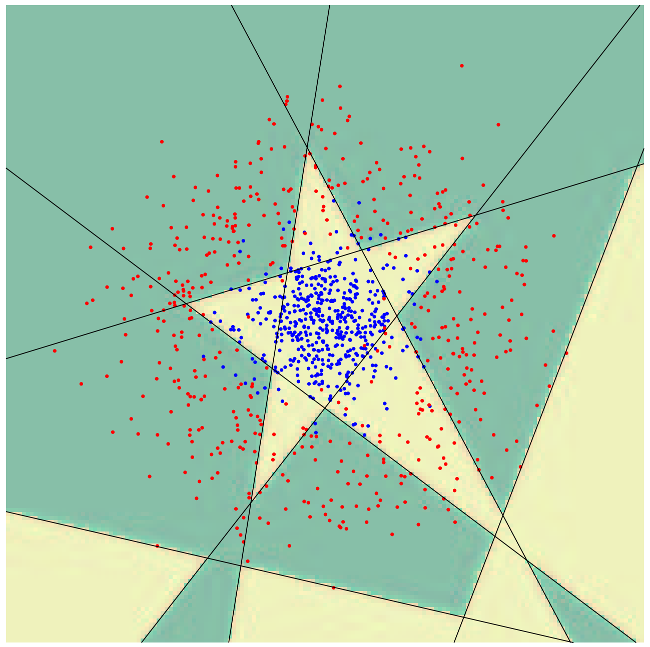
$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

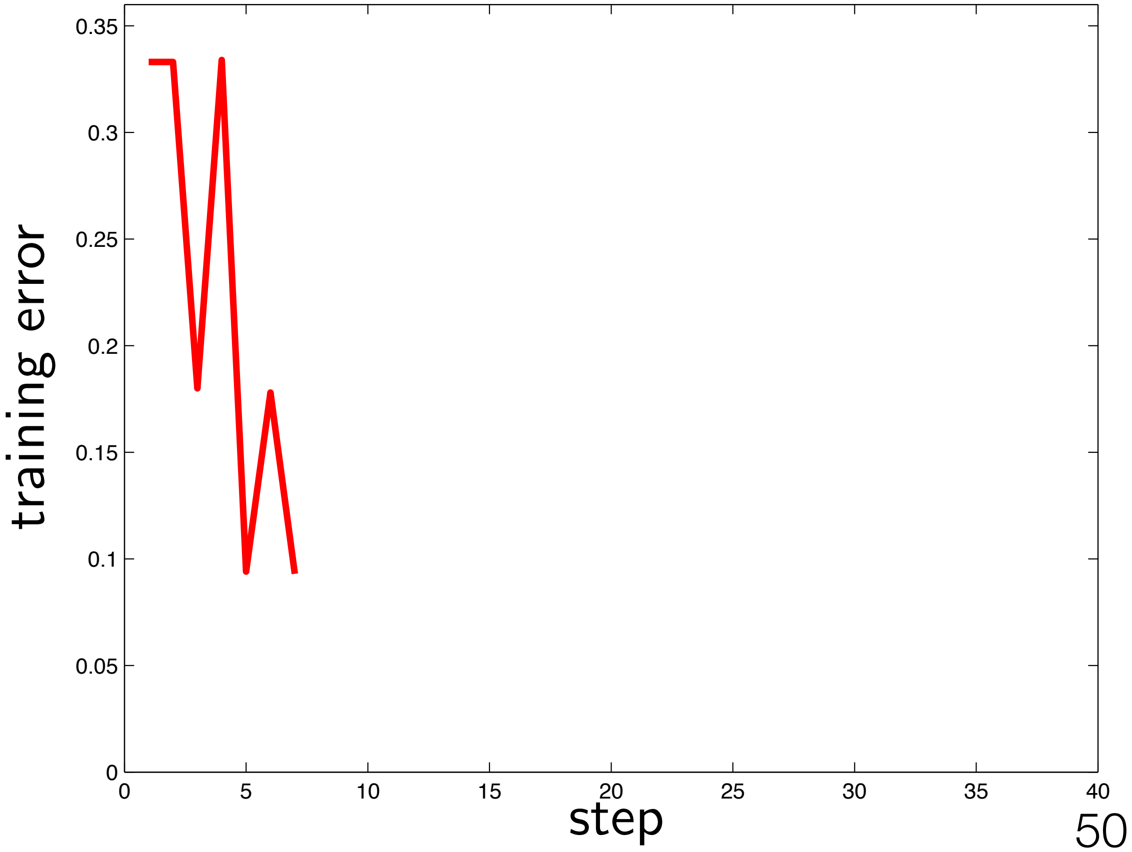
where Z_t is normalisation factor

Output the final classifier:

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

$t = 7$





slide by Jiri Matas and Jan Šochman

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

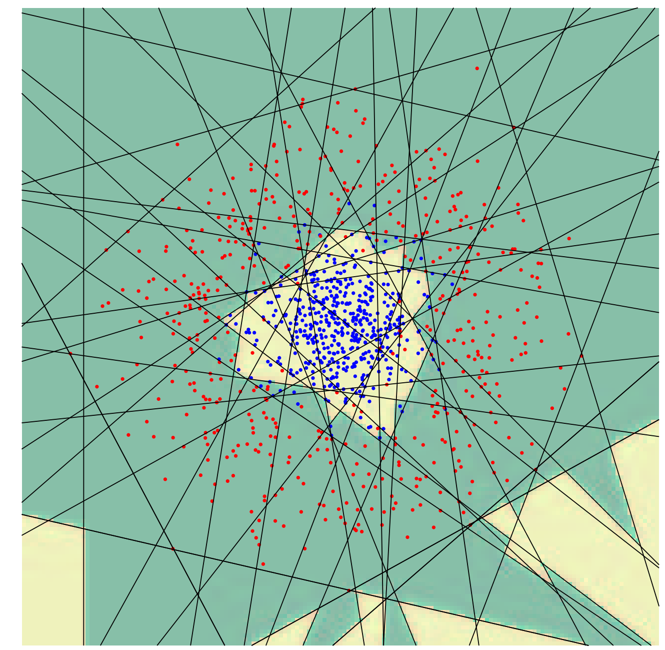
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

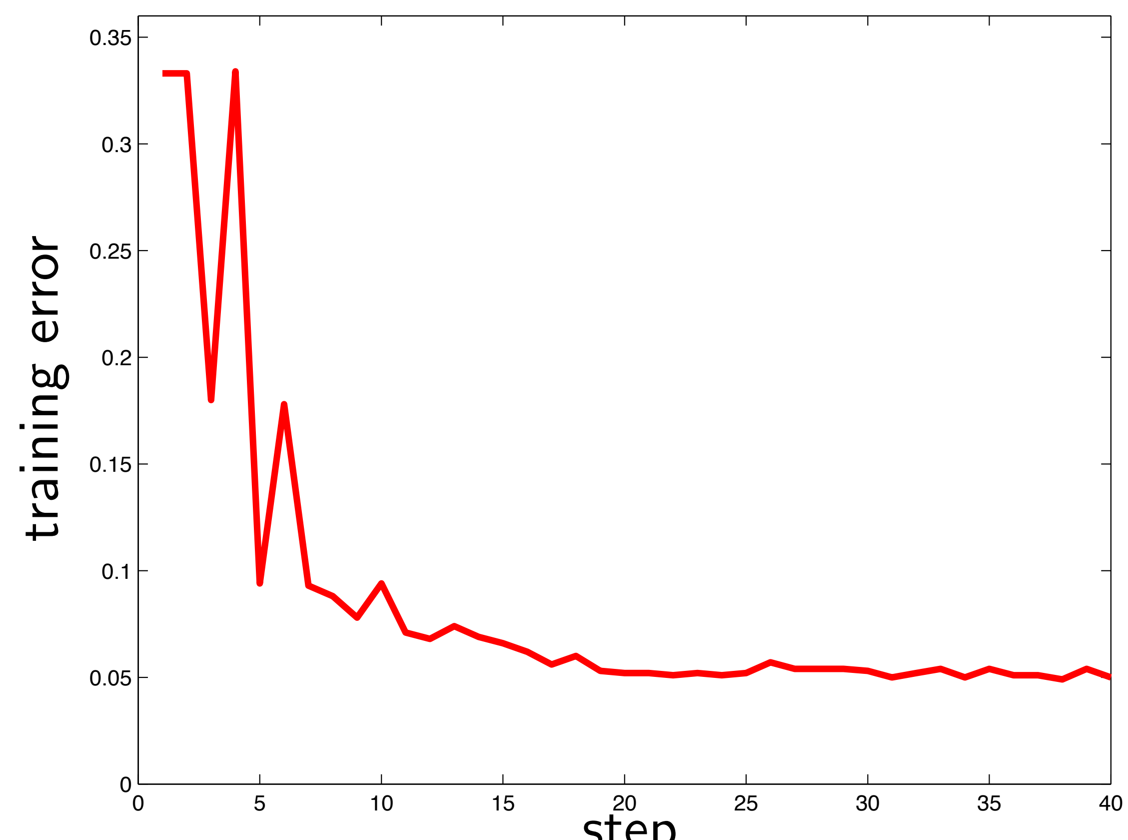
where Z_t is normalisation factor

Output the final classifier:

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

$t = 40$





slide by Jiri Matas and Jan Šochman

Reweighting

Effect on the training set

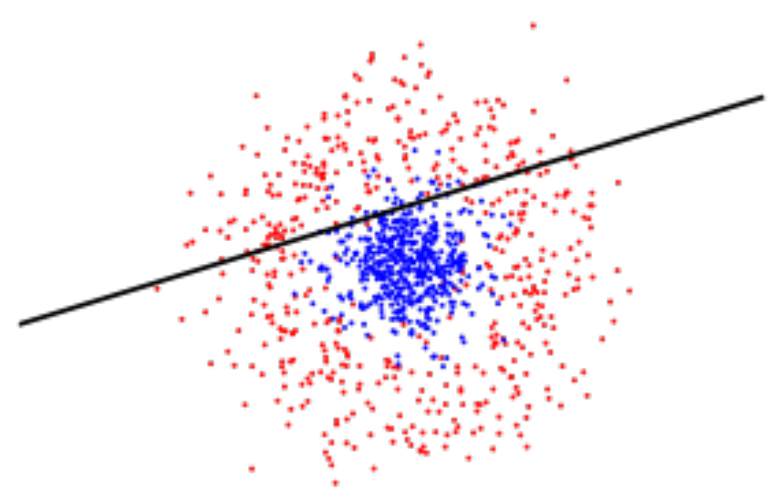
$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples

$\Rightarrow$ The weight is the upper bound on the error of a given example

$\Rightarrow$ All information about previously selected "features" is captured in D_t



slide by Jiri Matas and Jan Šochman

Reweighting

Effect on the training set

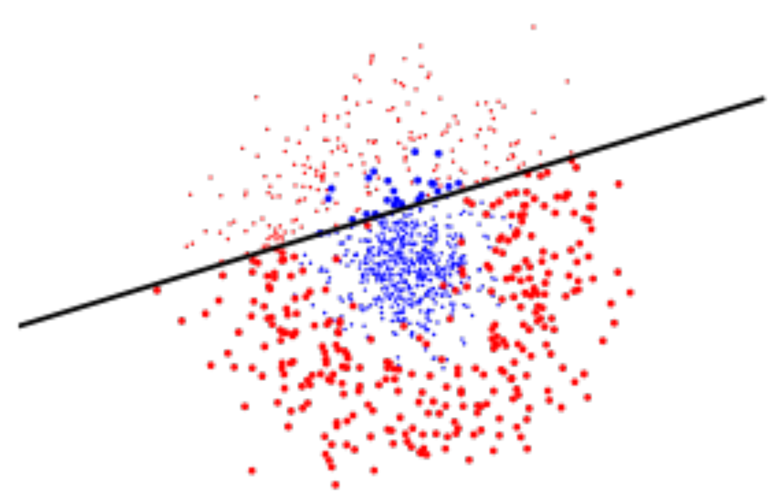
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples

$\Rightarrow$ The weight is the upper bound on the error of a given example

$\Rightarrow$ All information about previously selected "features" is captured in D_t



slide by Jiri Matas and Jan Šochman

Reweighting

Effect on the training set

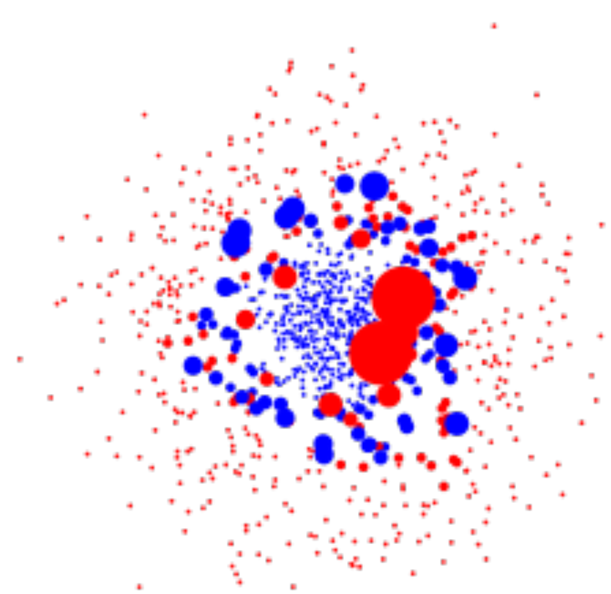
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

⇒ Increase (decrease) weight of wrongly (correctly) classified examples

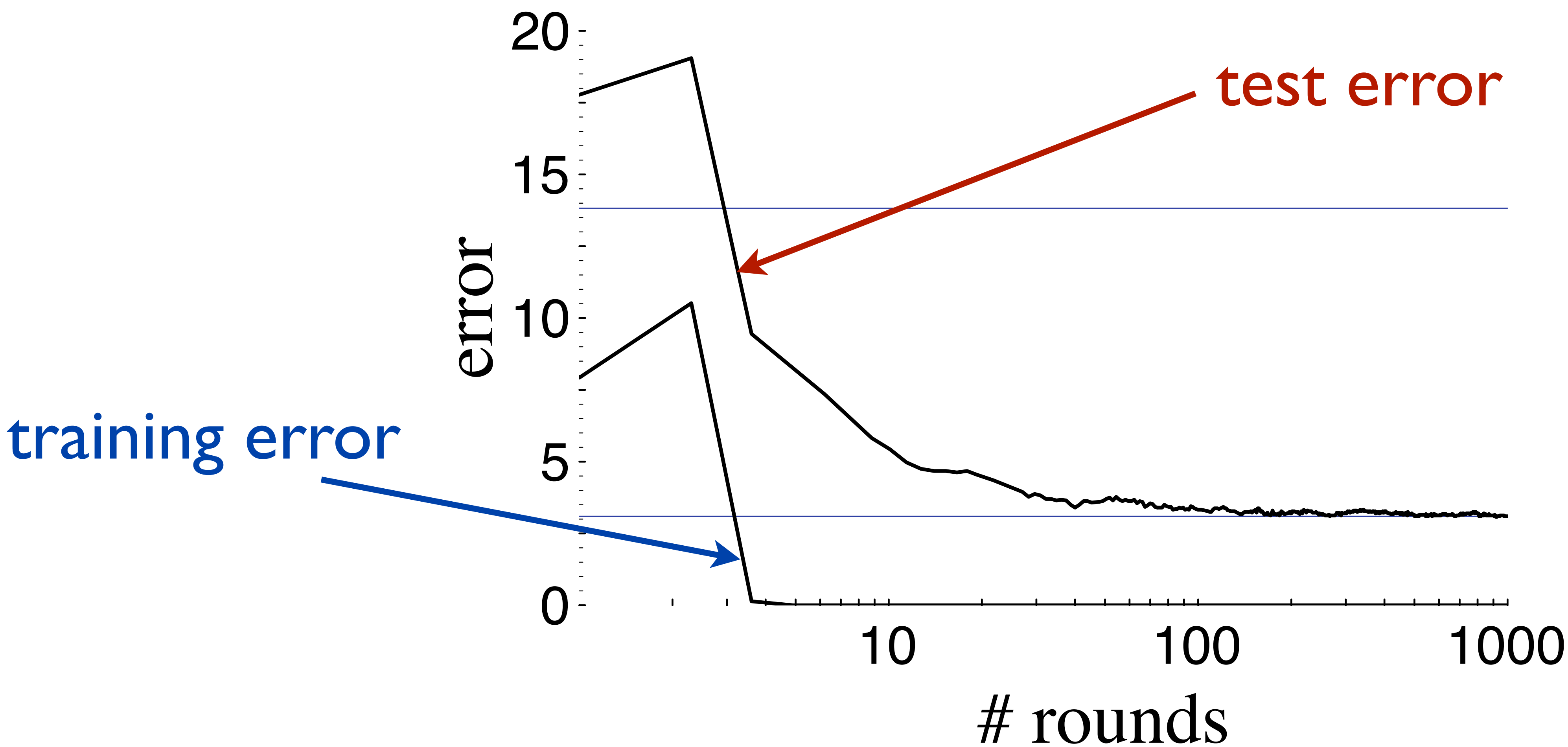
⇒ The weight is the upper bound on the error of a given example

⇒ All information about previously selected "features" is captured in D_t



slide by Jiri Matas and Jan Šochman

Boosting results – Digit recognition



- Boosting often (but not always)
  - Robust to overfitting
  - Test set error decreases even after training error is zero

[Schapire, 1989]

slide by Carlos Guestrin

Application: Detecting Faces

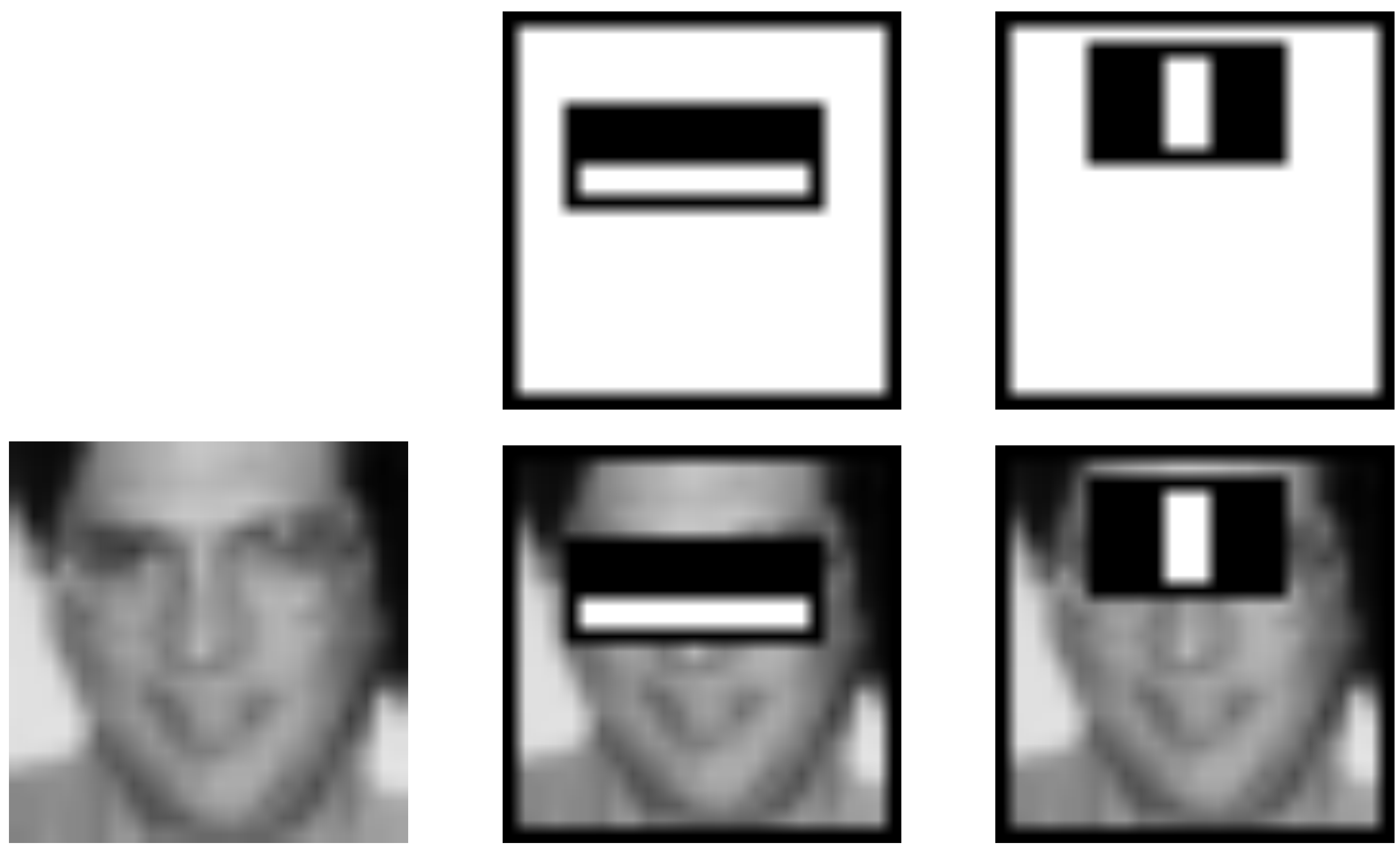
- Training Data
  - 5000 faces
    - All frontal
  - 300 million non-faces
    - 9500 non-face images

[Viola & Jones]

slide by Rob Schapire

Application: Detecting Faces

- **Problem:** find faces in photograph or movie
- **Weak classifiers:** detect light/dark rectangle in image



- Many clever tricks to make extremely fast and accurate

[Viola & Jones]

slide by Rob Schapire

Boosting vs. Logistic Regression

Logistic regression:

- Minimize log loss

$$\sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i)))$$

- Define

$$f(x) = \sum_j w_j x_j$$

where x_j predefined features (linear classifier)

- Jointly optimize over all weights $w_0, w_1, w_2, \ldots$

Boosting:

- Minimize exp loss

$$\sum_{i=1}^{m} \exp(-y_i f(x_i))$$

- Define

$$f(x) = \sum_t \alpha_t h_t(x)$$

where $h_t(x)$ defined dynamically to fit data (not a linear classifier)

- Weights α_t learned per iteration t incrementally

slide by Aarti Singh

Boosting vs. Bagging

Bagging:

- Resample data points
- Weight of each classifier is the same
- Only variance reduction

Boosting:

- Reweights data points (modifies their distribution)
- Weight is dependent on classifier's accuracy
- Both bias and variance reduced – learning rule becomes more complex with iterations

slide by Aarti Singh

Next Lecture:

K-Means Clustering