# Outline

1. Boosting (Thanks to Ameet Talkwalkar for these slides)
   - AdaBoost
   - Derivation of AdaBoost

# Boosting

**High-level idea**: combine a lot of classifiers

- Sequentially construct / identify these classifiers, $h_t(\cdot)$, one at a time
- Use *weak* classifiers to arrive at a complex decision boundary (*strong* classifier), where $\beta_t$ is the contribution of each weak classifier

$$h[\boldsymbol{x}] = \text{sign}\left[\sum_{t=1}^{T} \beta_t h_t(\boldsymbol{x})\right]$$

**Our plan**

- Describe AdaBoost algorithm
- Derive the algorithm

# Adaboost Algorithm

- Given: $n$ samples $\{\boldsymbol{x}_i, y_i\}$, where $y_i \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(i) = \frac{1}{n}$ for every training sample
- For $t = 1$ to $T$
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(i)$, by minimizing

$$\epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

# Adaboost Algorithm

- Given: $n$ samples $\{\boldsymbol{x}_i, y_i\}$, where $y_i \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(i) = \frac{1}{n}$ for every training sample
- For $t = 1$ to $T$
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(i)$, by minimizing

$$\epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] \quad \text{(the weighted classification error)}$$

  2. Compute contribution for this classifier

# Adaboost Algorithm

- Given: $n$ samples $\{\boldsymbol{x}_i, y_i\}$, where $y_i \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(i) = \frac{1}{n}$ for every training sample
- For $t = 1$ to $T$
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(i)$, by minimizing

$$\epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] \quad \text{(the weighted classification error)}$$

  2. Compute contribution for this classifier: $\beta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$

# Adaboost Algorithm

- Given: $n$ samples $\{\boldsymbol{x}_i, y_i\}$, where $y_i \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(i) = \frac{1}{n}$ for every training sample
- For $t = 1$ to $T$
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(i)$, by minimizing

$$\epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] \quad \text{(the weighted classification error)}$$

  2. Compute contribution for this classifier: $\beta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  3. Update weights on training points

# Adaboost Algorithm

- Given: $n$ samples $\{\boldsymbol{x}_i, y_i\}$, where $y_i \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(i) = \frac{1}{n}$ for every training sample
- For $t = 1$ to $T$
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(i)$, by minimizing

$$\epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] \quad \text{(the weighted classification error)}$$

  2. Compute contribution for this classifier: $\beta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  3. Update weights on training points

$$w_{t+1}(i) \propto w_t(i) e^{-\beta_t y_i h_t(\boldsymbol{x}_i)}$$

# Adaboost Algorithm

- Given: $n$ samples $\{\boldsymbol{x}_i, y_i\}$, where $y_i \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(i) = \frac{1}{n}$ for every training sample
- For $t = 1$ to $T$
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(i)$, by minimizing

$$\epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] \quad \text{(the weighted classification error)}$$

  2. Compute contribution for this classifier: $\beta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  3. Update weights on training points

$$w_{t+1}(i) \propto w_t(i) e^{-\beta_t y_i h_t(\boldsymbol{x}_i)}$$

     and normalize them such that $\sum_i w_{t+1}(i) = 1$

# Adaboost Algorithm

- Given: $n$ samples $\{\boldsymbol{x}_i, y_i\}$, where $y_i \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(i) = \frac{1}{n}$ for every training sample
- For $t = 1$ to $T$
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(i)$, by minimizing

$$\epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] \quad \text{(the weighted classification error)}$$

  2. Compute contribution for this classifier: $\beta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  3. Update weights on training points

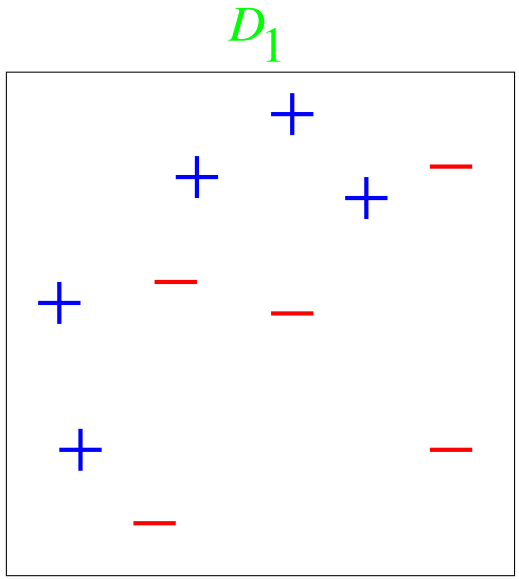
$$w_{t+1}(i) \propto w_t(i) e^{-\beta_t y_i h_t(\boldsymbol{x}_i)}$$

     and normalize them such that $\sum_i w_{t+1}(i) = 1$
- Output the final classifier

$$h[\boldsymbol{x}] = \text{sign}\left[\sum_{t=1}^{T} \beta_t h_t(\boldsymbol{x})\right]$$

# Example

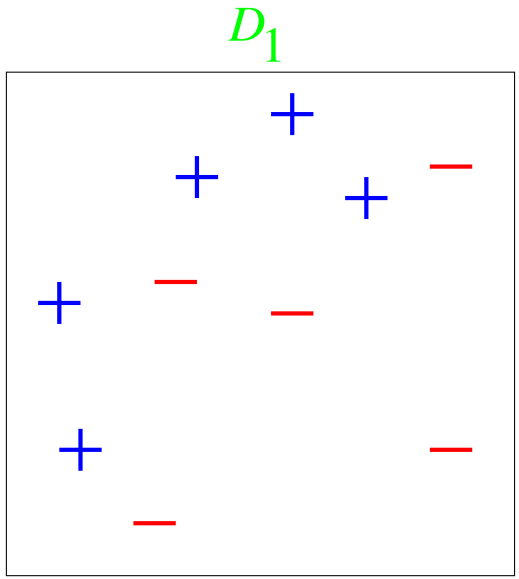
## 10 data points and 2 features




- The data points are clearly not linear separable
- In the beginning, all data points have equal weights (the size of the data markers "+" or "-")

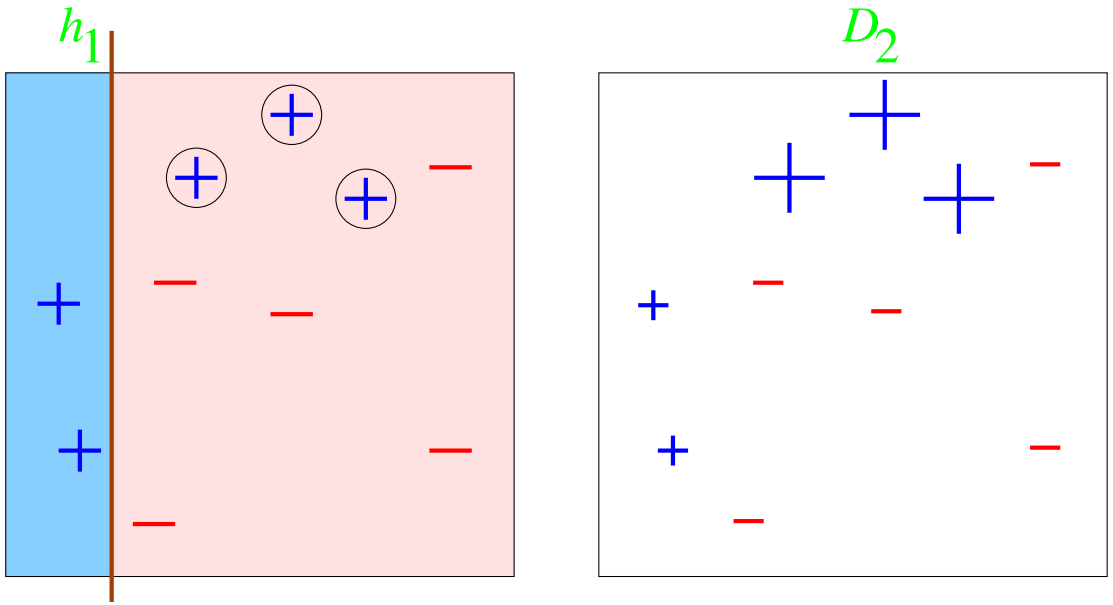
# Example

## 10 data points and 2 features

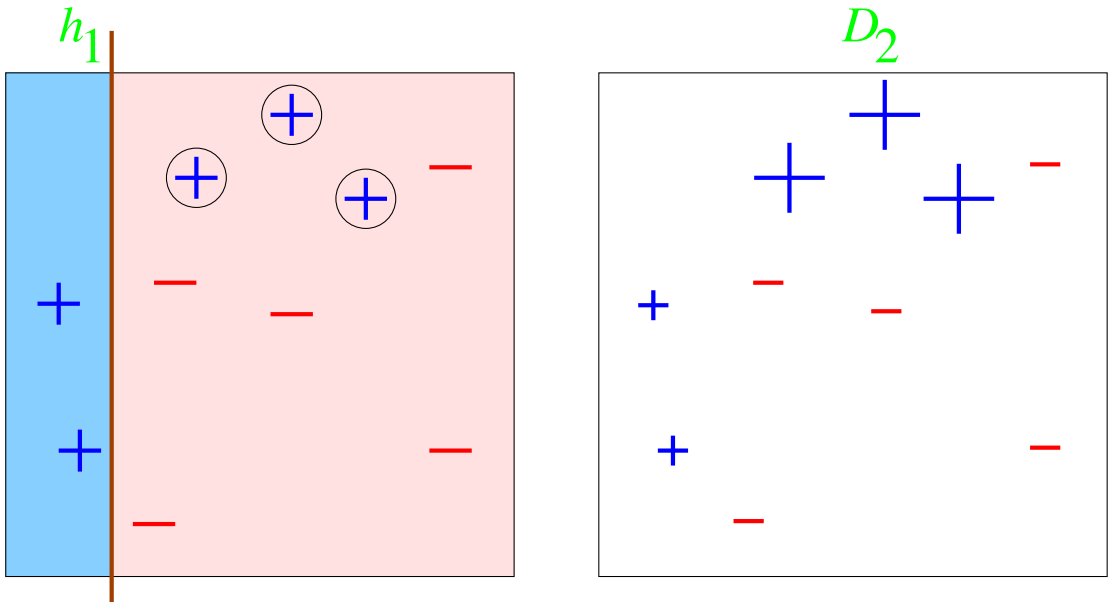



- The data points are clearly not linear separable
- In the beginning, all data points have equal weights (the size of the data markers "+" or "-")
- Base classifier $h(\cdot)$: horizontal or vertical lines ('decision stumps')
  - Depth-1 decision trees, i.e., classify data based on a single attribute

# Round 1: $t = 1$

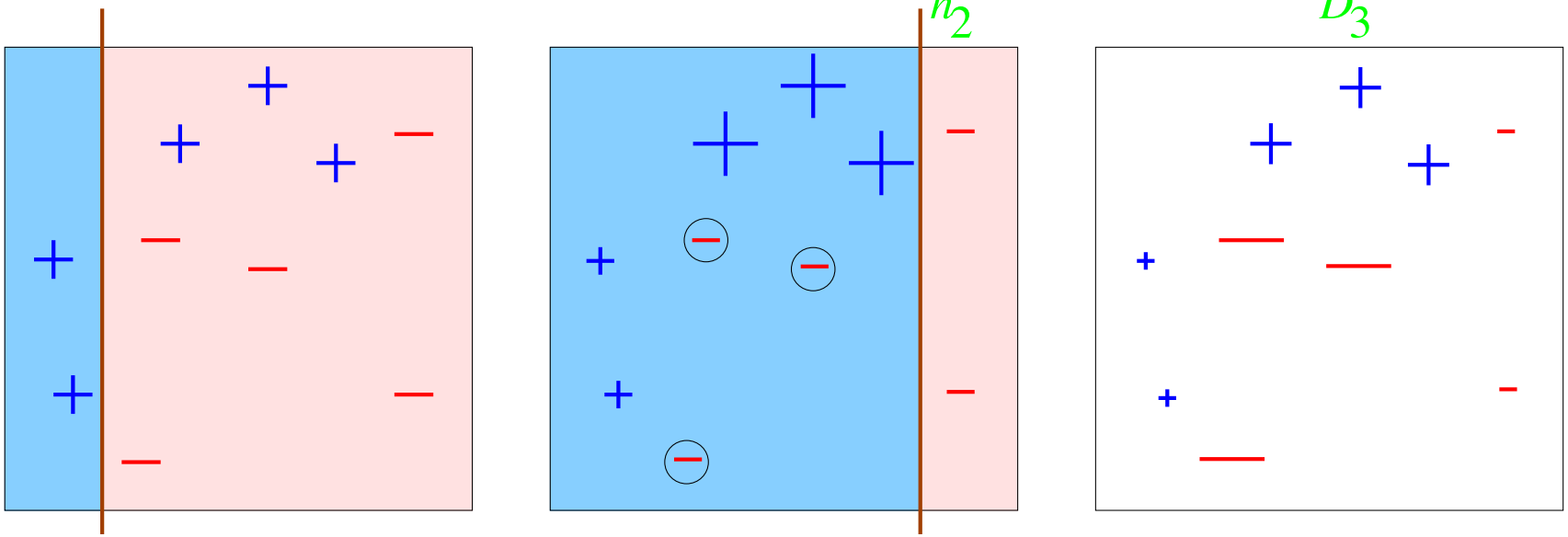


# Round 1: $t = 1$

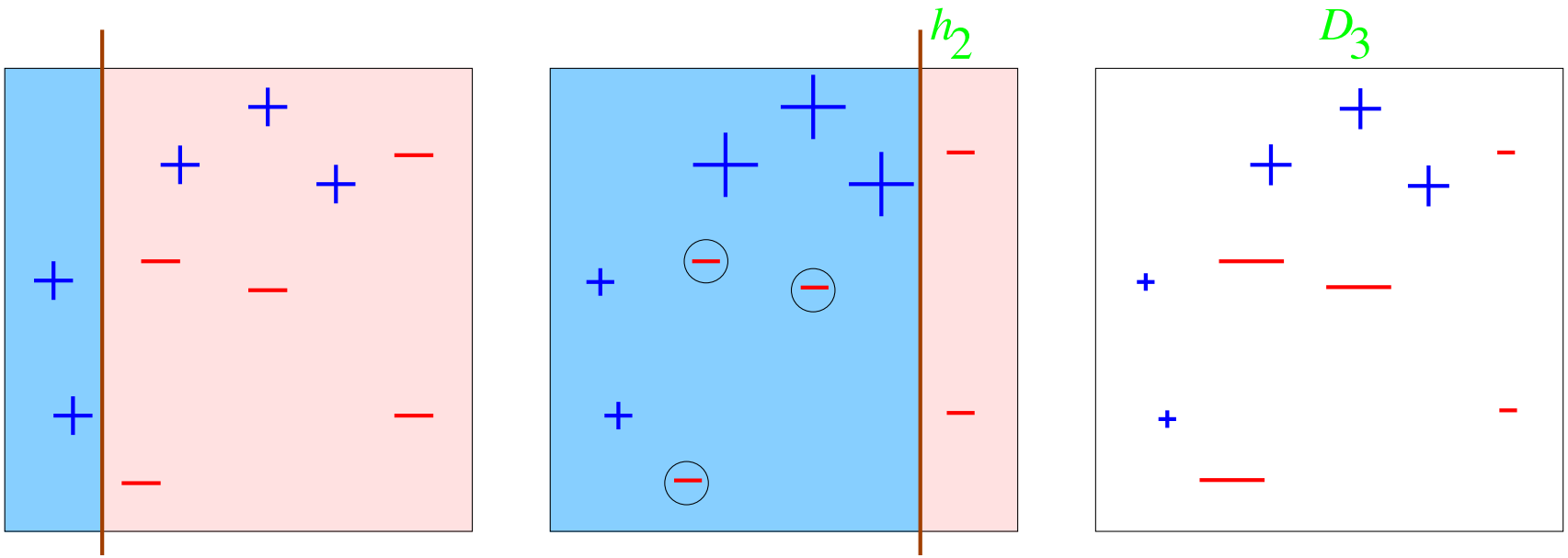



- 3 misclassified (with circles): $\epsilon_1 = 0.3 \rightarrow \beta_1 = 0.42$.
- Weights recomputed; the 3 misclassified data points receive larger weights

# Round 2: $t = 2$

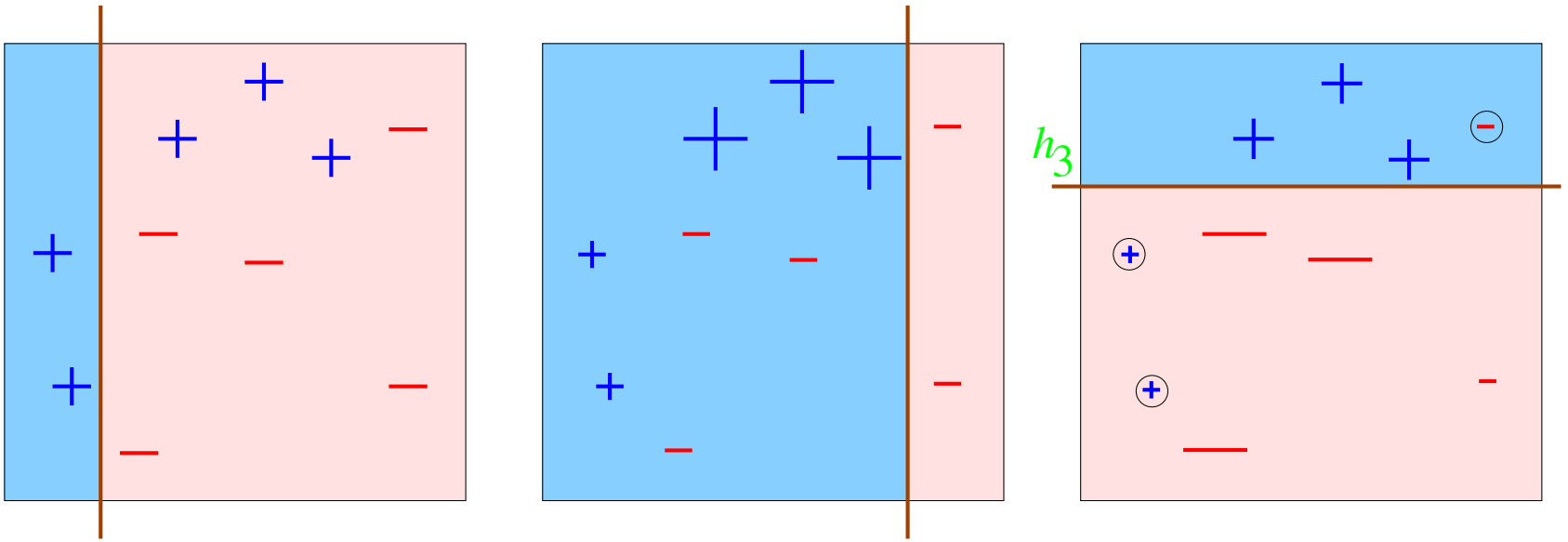


# Round 2: $t = 2$

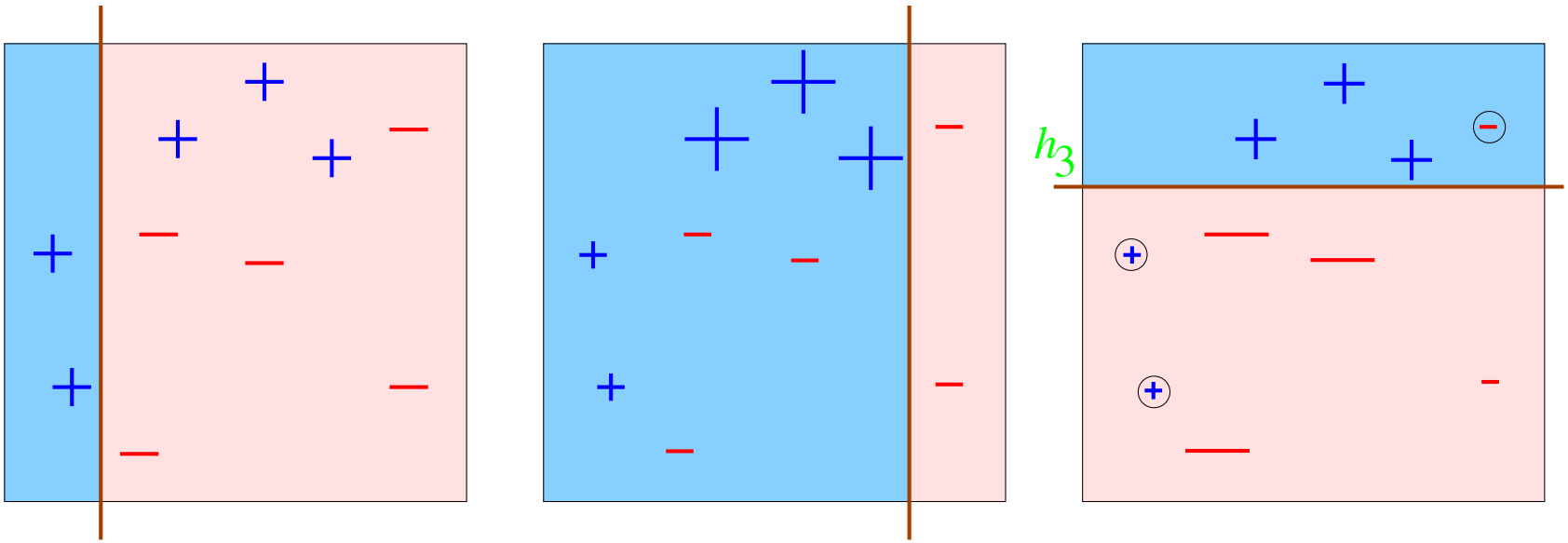



- 3 misclassified (with circles): $\epsilon_2 = 0.21 \rightarrow \beta_2 = 0.65$.
  note that $\epsilon_2 \neq 0.3$ as those 3 data points have weights less than $1/10$
- 3 misclassified data points get larger weights
- Data points classified correctly in both rounds have small weights

# Round 3: $t = 3$

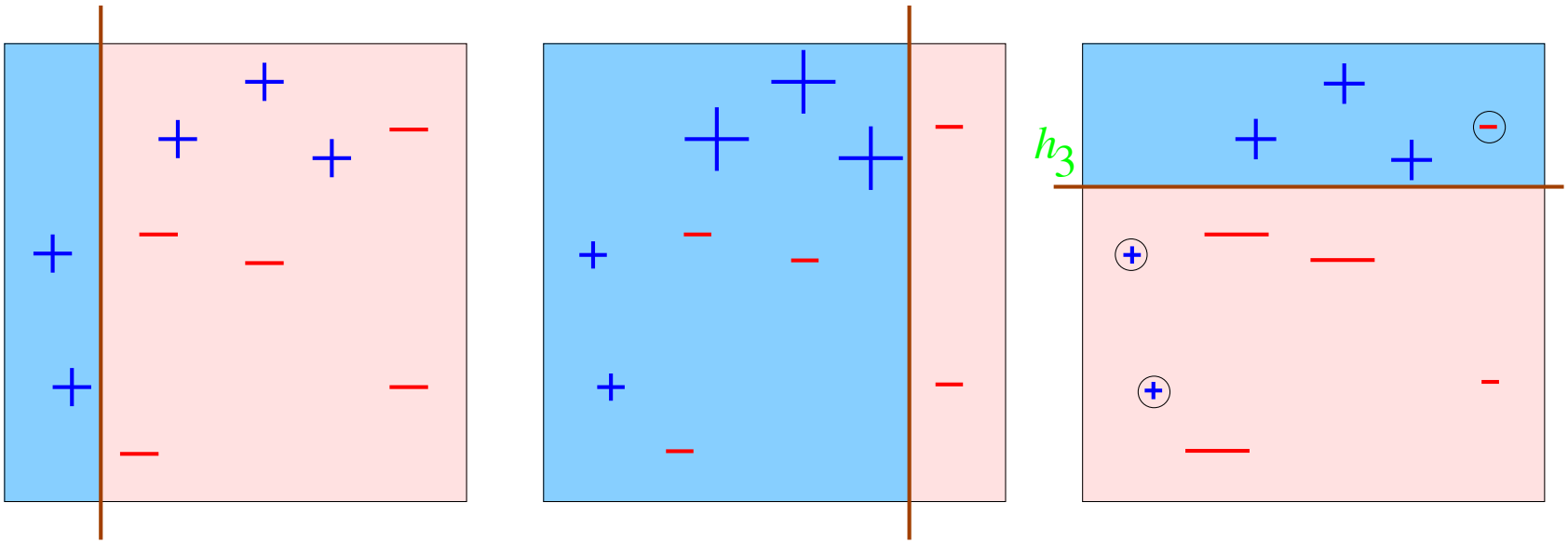


# Round 3: $t = 3$



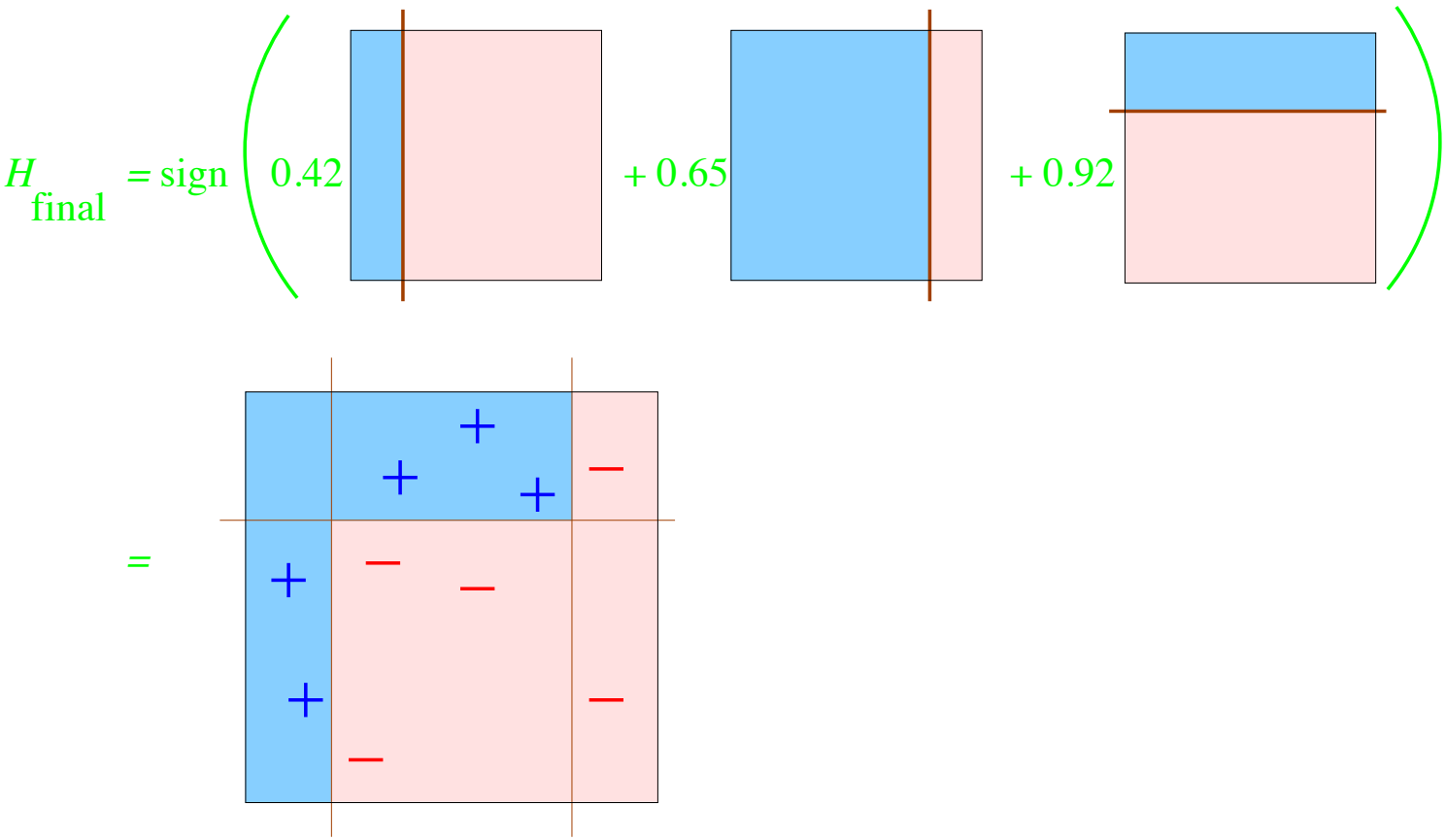

- 3 misclassified (with circles): $\epsilon_3 = 0.14 \rightarrow \beta_3 = 0.92$.
- Previously correctly classified data points are now misclassified, hence our error is low; what's the intuition?

# Round 3: $t = 3$




- 3 misclassified (with circles): $\epsilon_3 = 0.14 \rightarrow \beta_3 = 0.92$.
- Previously correctly classified data points are now misclassified, hence our error is low; what's the intuition?
  - Since they have been consistently classified correctly, this round's mistake will hopefully not have a huge impact on the overall prediction

# Final classifier: combining 3 classifiers




- All data points are now classified correctly!

# Why AdaBoost works?

It minimizes a loss function related to classification error.

**Classification loss**

- Suppose we want to have a classifier

$$h(\boldsymbol{x}) = \text{sign}[f(\boldsymbol{x})] = \begin{cases} 1 & \text{if } f(\boldsymbol{x}) > 0 \\ -1 & \text{if } f(\boldsymbol{x}) < 0 \end{cases}$$

- One seemingly natural loss function is 0-1 loss:

$$\ell(h(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } yf(\boldsymbol{x}) > 0 \\ 1 & \text{if } yf(\boldsymbol{x}) < 0 \end{cases}$$

namely, the function $f(\boldsymbol{x})$ and the target label $y$ should have the same sign to avoid a loss of $1$.

# Surrogate loss

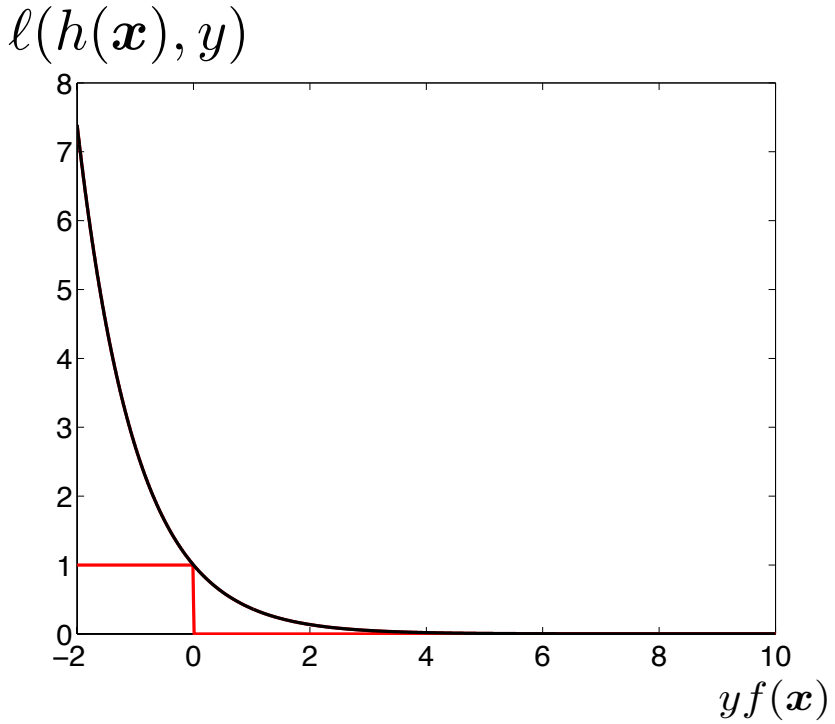
$0-1$ loss function $\ell(h(\boldsymbol{x}), y)$ is non-convex and difficult to optimize.
We can instead use a surrogate loss – what are examples?

# Surrogate loss

$0-1$ loss function $\ell(h(\boldsymbol{x}), y)$ is non-convex and difficult to optimize. We can instead use a surrogate loss – what are examples?

## Exponential Loss

$$\ell^{\text{EXP}}(h(\boldsymbol{x}), y) = e^{-yf(\boldsymbol{x})}$$



# Choosing the $t$-th classifier

Suppose a classifier $f_{t-1}(\boldsymbol{x})$, and want to add a weak learner $h_t(\boldsymbol{x})$

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t h_t(\boldsymbol{x})$$

note: $h_t(\cdot)$ outputs $-1$ or $1$, as does sign $[f_{t-1}(\cdot)]$

# Choosing the $t$-th classifier

Suppose a classifier $f_{t-1}(\boldsymbol{x})$, and want to add a weak learner $h_t(\boldsymbol{x})$

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t h_t(\boldsymbol{x})$$

note: $h_t(\cdot)$ outputs $-1$ or $1$, as does sign $[f_{t-1}(\cdot)]$

How can we 'optimally' choose $h_t(\boldsymbol{x})$ and combination coefficient $\beta_t$?
Adaboost greedily *minimizes the exponential loss function*!

$$(h_t^*(\boldsymbol{x}), \beta_t^*) = \arg\min_{(h_t(\boldsymbol{x}), \beta_t)} \sum_i e^{-y_i f(\boldsymbol{x}_i)}$$

# Choosing the $t$-th classifier

Suppose a classifier $f_{t-1}(\boldsymbol{x})$, and want to add a weak learner $h_t(\boldsymbol{x})$

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t h_t(\boldsymbol{x})$$

note: $h_t(\cdot)$ outputs $-1$ or $1$, as does sign $[f_{t-1}(\cdot)]$

How can we 'optimally' choose $h_t(\boldsymbol{x})$ and combination coefficient $\beta_t$?
Adaboost greedily *minimizes the exponential loss function*!

$$\begin{aligned}(h_t^*(\boldsymbol{x}), \beta_t^*) &= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i e^{-y_i f(\boldsymbol{x}_i)} \\ &= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i e^{-y_i [f_{t-1}(\boldsymbol{x}_i) + \beta_t h_t(\boldsymbol{x}_i)]}\end{aligned}$$

# Choosing the $t$-th classifier

Suppose a classifier $f_{t-1}(\boldsymbol{x})$, and want to add a weak learner $h_t(\boldsymbol{x})$

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t h_t(\boldsymbol{x})$$

note: $h_t(\cdot)$ outputs $-1$ or $1$, as does sign $[f_{t-1}(\cdot)]$

How can we 'optimally' choose $h_t(\boldsymbol{x})$ and combination coefficient $\beta_t$?
Adaboost greedily *minimizes the exponential loss function*!

$$\begin{aligned}
(h_t^*(\boldsymbol{x}), \beta_t^*) &= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i e^{-y_i f(\boldsymbol{x}_i)} \\
&= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i e^{-y_i [f_{t-1}(\boldsymbol{x}_i) + \beta_t h_t(\boldsymbol{x}_i)]} \\
&= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}
\end{aligned}$$

where we have used $w_t(i)$ as a shorthand for $e^{-y_i f_{t-1}(\boldsymbol{x}_i)}$

# The new classifier

We can decompose the *weighted* loss function into two parts

$$\sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}$$

$$= \sum_i w_t(i) e^{\beta_t} \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] + \sum_i w_t(i) e^{-\beta_t} \mathbb{I}[y_i = h_t(\boldsymbol{x}_i)]$$

# The new classifier

We can decompose the *weighted* loss function into two parts

$$\sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}$$

$$= \sum_i w_t(i) e^{\beta_t} \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] + \sum_i w_t(i) e^{-\beta_t} \mathbb{I}[y_i = h_t(\boldsymbol{x}_i)]$$

$$= \sum_i w_t(i) e^{\beta_t} \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] + \sum_i w_t(i) e^{-\beta_t} (1 - \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)])$$

# The new classifier

We can decompose the *weighted* loss function into two parts

$$\begin{aligned}
&\sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)} \\
&= \sum_i w_t(i) e^{\beta_t} \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] + \sum_i w_t(i) e^{-\beta_t} \mathbb{I}[y_i = h_t(\boldsymbol{x}_i)] \\
&= \sum_i w_t(i) e^{\beta_t} \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] + \sum_i w_t(i) e^{-\beta_t} (1 - \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]) \\
&= (e^{\beta_t} - e^{-\beta_t}) \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] + e^{-\beta_t} \sum_i w_t(i)
\end{aligned}$$

We have used the following properties to derive the above

- $y_i h_t(\boldsymbol{x}_i)$ is either 1 or -1 as $h_t(\boldsymbol{x}_i)$ is the output of a binary classifier
- The indicator function $\mathbb{I}[y_i = h_t(\boldsymbol{x}_i)]$ is either 0 or 1, so it equals $1 - \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$

# Finding the optimal weak learner

## Summary

$$(h_t^*(\boldsymbol{x}), \beta_t^*) = \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}$$

$$= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

$$+ e^{-\beta_t} \sum_i w_t(i)$$

**What term(s) must we optimize to choose $h_t(\boldsymbol{x}_i)$?**

# Finding the optimal weak learner

## Summary

$$(h_t^*(\boldsymbol{x}), \beta_t^*) = \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}$$

$$= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

$$+ e^{-\beta_t} \sum_i w_t(i)$$

## What term(s) must we optimize to choose $h_t(\boldsymbol{x}_i)$?

$$h_t^*(\boldsymbol{x}) = \arg\min_{h_t(\boldsymbol{x})} \epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

# Finding the optimal weak learner

## Summary

$$(h_t^*(\boldsymbol{x}), \beta_t^*) = \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}$$

$$= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

$$+ e^{-\beta_t} \sum_i w_t(i)$$

## What term(s) must we optimize to choose $h_t(\boldsymbol{x}_i)$?

$$h_t^*(\boldsymbol{x}) = \arg\min_{h_t(\boldsymbol{x})} \epsilon_t = \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

Minimize weighted classification error as noted in step 1 of Adaboost!

# How to choose $\beta_t$?

## Summary

$$(h_t^*(\boldsymbol{x}), \beta_t^*) = \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}$$

$$= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

$$+ e^{-\beta_t} \sum_i w_t(i)$$

## What term(s) must we optimize?

# How to choose $\beta_t$?

## Summary

$$(h_t^*(\boldsymbol{x}), \beta_t^*) = \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}$$

$$= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

$$+ e^{-\beta_t} \sum_i w_t(i)$$

## What term(s) must we optimize?

We need to minimize the entire objective function with respect to $\beta_t$!

# How to choose $\beta_t$?

## Summary

$$(h_t^*(\boldsymbol{x}), \beta_t^*) = \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_i w_t(i) e^{-y_i \beta_t h_t(\boldsymbol{x}_i)}$$

$$= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_i w_t(i) \mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)]$$

$$+ e^{-\beta_t} \sum_i w_t(i)$$

## What term(s) must we optimize?

We need to minimize the entire objective function with respect to $\beta_t$!

We can do this by taking derivative with respect to $\beta_t$, setting to zero, and solving for $\beta_t$. After some calculation and using $\sum_i w_t(i) = 1$, we find:

$$\beta_t^* = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$

which is precisely step 2 of Adaboost! (*Exercise – verify the solution*)

# Updating the weights

Once we find the optimal weak learner we can update our classifier:

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x})$$

# Updating the weights

Once we find the optimal weak learner we can update our classifier:

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x})$$

We then need to compute the weights for the above classifier as:

$$w_{t+1}(i) = e^{-y_i f(\boldsymbol{x}_i)} = e^{-y_i[f_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x}_i)]}$$

# Updating the weights

Once we find the optimal weak learner we can update our classifier:

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x})$$

We then need to compute the weights for the above classifier as:

$$w_{t+1}(i) = e^{-y_i f(\boldsymbol{x}_i)} = e^{-y_i[f_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x}_i)]}$$

$$= w_t(i) e^{-y_i \beta_t^* h_t^*(\boldsymbol{x}_i)}$$

# Updating the weights

Once we find the optimal weak learner we can update our classifier:

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x})$$

We then need to compute the weights for the above classifier as:

$$\begin{aligned} w_{t+1}(i) &= e^{-y_i f(\boldsymbol{x}_i)} = e^{-y_i[f_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x}_i)]} \\ &= w_t(i) e^{-y_i \beta_t^* h_t^*(\boldsymbol{x}_i)} = \begin{cases} w_t(i) e^{\beta_t^*} & \text{if } y_i \neq h_t^*(\boldsymbol{x}_i) \\ w_t(i) e^{-\beta_t^*} & \text{if } y_i = h_t^*(\boldsymbol{x}_i) \end{cases} \end{aligned}$$

**Intuition** Misclassified data points will get their weights increased, while correctly classified data points will get their weight decreased

# Meta-Algorithm

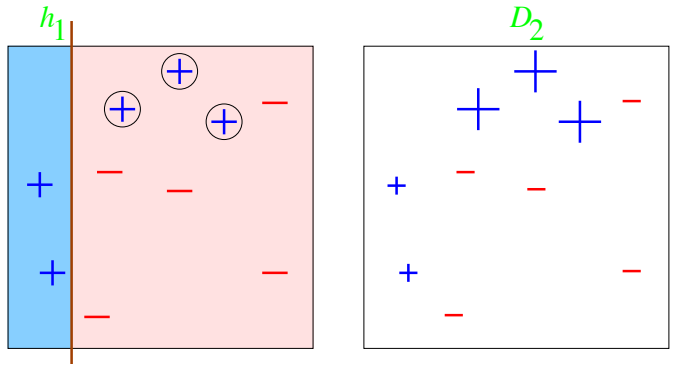
note that the AdaBoost algorithm itself never specifies how we would get $h_t^*(\boldsymbol{x})$ as long as it minimizes the weighted classification error

$$\epsilon_t = \sum_i w_t(i)\mathbb{I}[y_i \neq h_t^*(\boldsymbol{x}_i)]$$

In this aspect, the AdaBoost algorithm is a meta-algorithm and can be used with any type of classifier

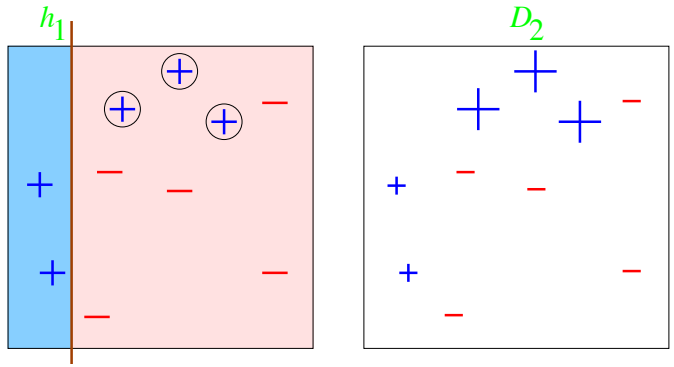
# E.g., Decision Stumps

How do we choose the decision stump classifier given the weights at the second round of the following distribution?



# E.g., Decision Stumps

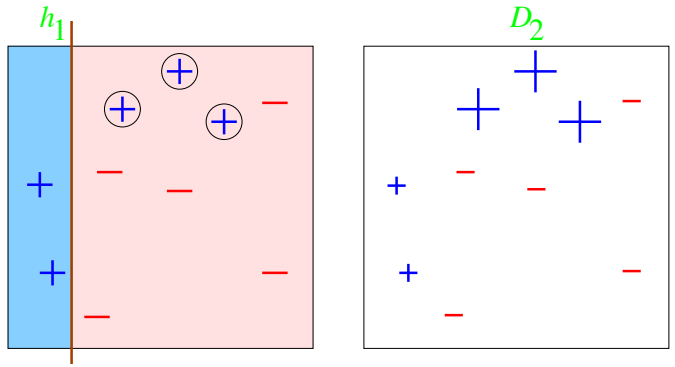
How do we choose the decision stump classifier given the weights at the second round of the following distribution?




We can simply enumerate all possible ways of putting vertical and horizontal lines to separate the data points into two classes and find the one with the smallest weighted classification error! Runtime?

# E.g., Decision Stumps

How do we choose the decision stump classifier given the weights at the second round of the following distribution?




We can simply enumerate all possible ways of putting vertical and horizontal lines to separate the data points into two classes and find the one with the smallest weighted classification error! Runtime?

- Presort data by each feature in $O(dn \log n)$ time
- Evaluate $n + 1$ thresholds for each feature at each round in $O(dn)$ time
- In total $O(dn \log n + dnT)$ time – this efficiency is an attractive quality of boosting!