Pattern Recognition
and Applications Lab

Artificial Intelligence
Ensembles

Ambra Demontis

Outline

- Bias and Variance
- Bias Variance Trade-off
- Bagging
- Random Forest
- Boosting
- AdaBoost

Bias

Suppose you would like to estimate the value of a feature of an object x using an instrument.

The true value of that feature is y.

The value that you obtain from a single measurement with the instrument you have is f(x), and you can repeat this measurement many time and average the result.

The **bias** is the difference between the average estimate and the true value, namely the *error*.

Bias : $E[\, f(x) \,] - y$

Variance

Suppose you would like to estimate the value of a feature of an object x using an instrument.

The true value of that feature is y.

The value that you obtain from a single measurement with the instrument you have is f(x), and you can repeat this measurement many time and average the result.

The **variance** is the variability between the different estimates that you get.

Variance : $E[\,(f(x) - E[f(x)])^2\,]$

High Bias and High Variance

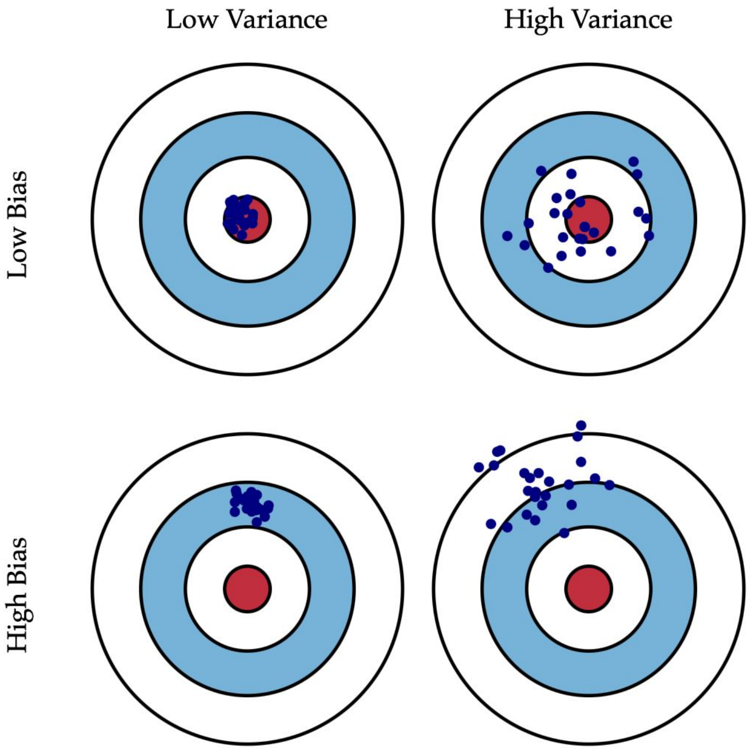
Suppose you would like to measure something with an instrument that has not been calibrated and thus makes a systematic error.

If you make the same measurement many times, you obtain almost the same value. However, unfortunately, its average will be quite different from the true one.

Therefore, the bias will be high and the variance will be low.

If you have an instrument that, for repeated measurement, gives you quite different values, the variance will be high.

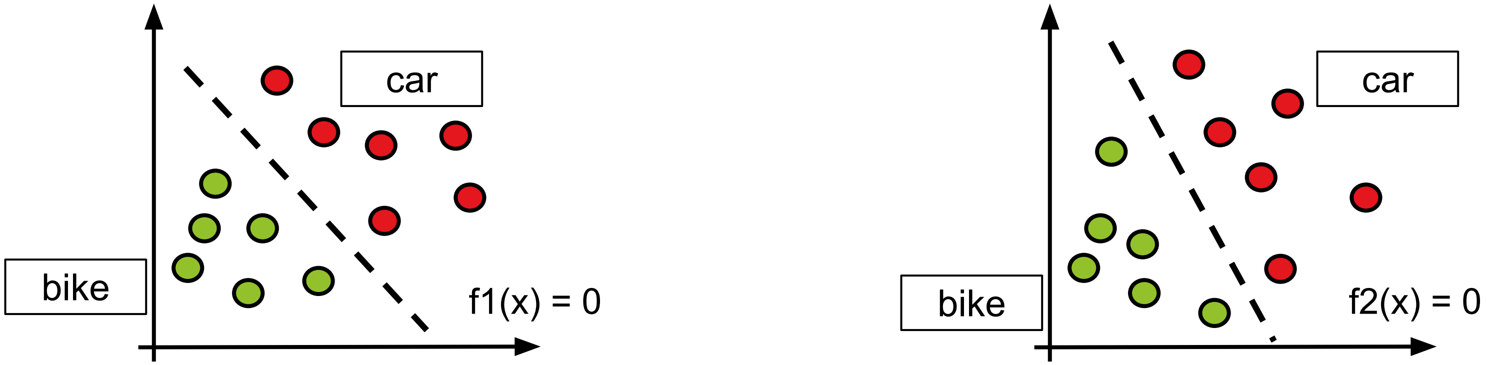
Bias and Variance



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

Recap: Learning Model's Goal

When we devise a machine learning model to perform a task we would like it to work well on different train / test split that we can obtain from the underlying data distribution.



Recap: Model Complexity

Machine learning models have a different complexity.

Depending on the model type and learning algorithms, this complexity can be bounded in different ways:

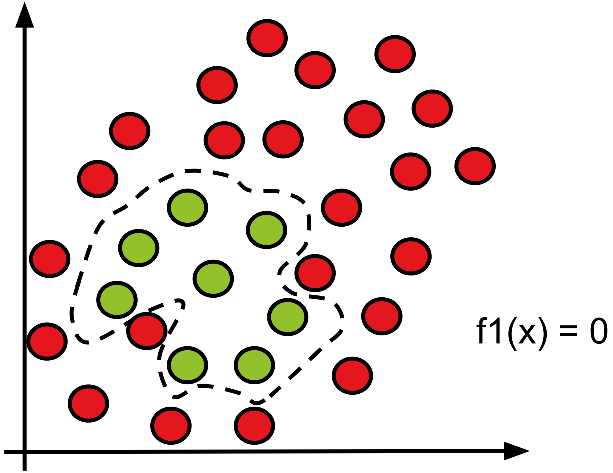
- Decision trees, e.g., bounding the depth
- Artificial Neural Network, e.g., using a small number of neurons.

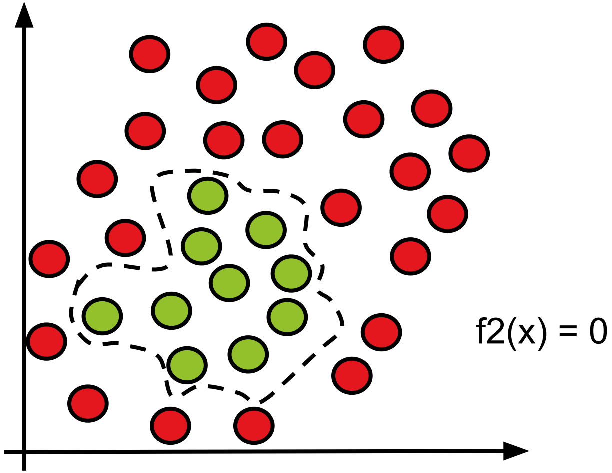
Models with different complexity behave differently on different train-test splits.

The Behavior of High-Complexity Models

Models that can learn complex functions usually works better on the training data but they tend to **overfit** and have a **high variance**. Where the variance is the difference in the classifier's output when it is trained on different training dataset coming from the same distribution.

The image below shows two highly-complex classifiers trained on different training subset (the point showed.)

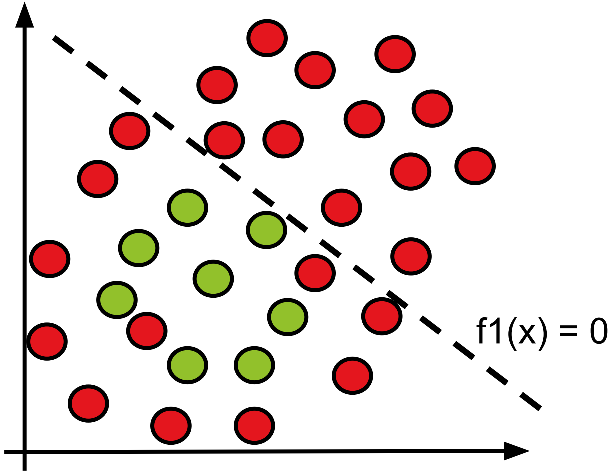


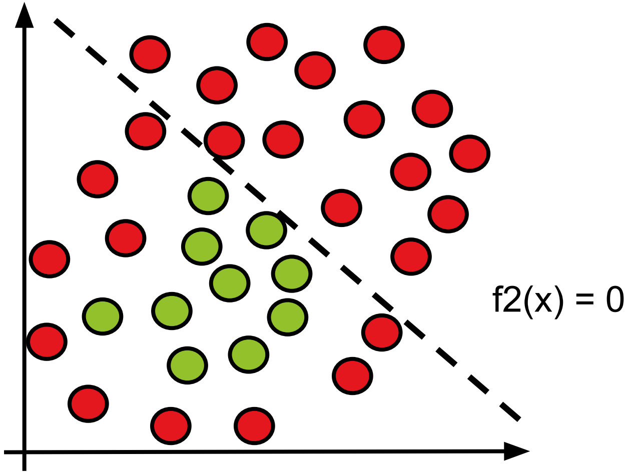


The Behavior of Low-Complexity Models

Instead, model with a low complexity usually have a **low variance**.

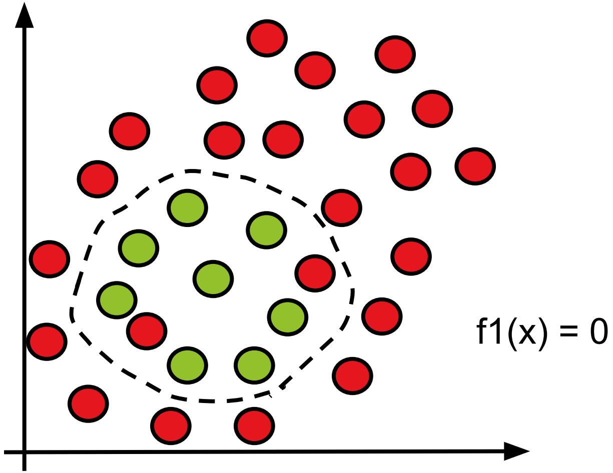
However, they may incur in **underfitting**.

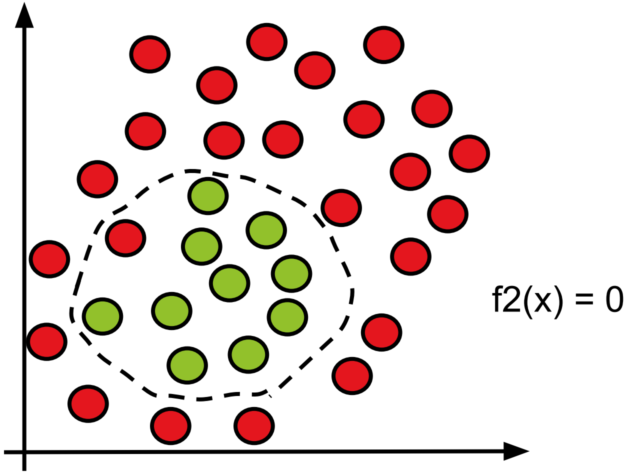




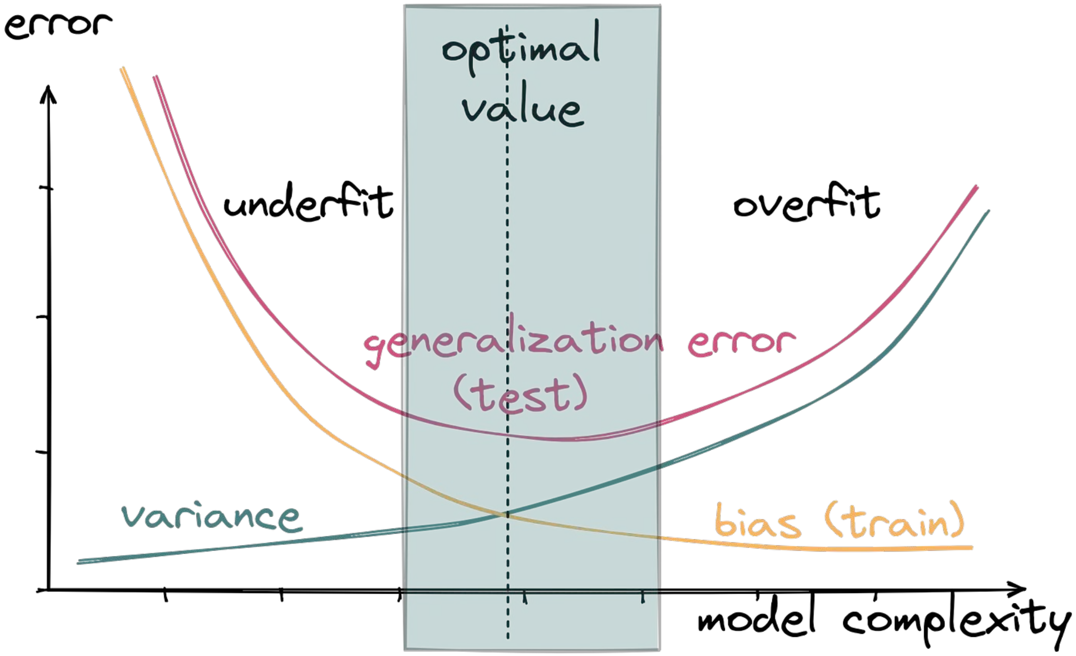
The Behavior of Low-Complexity Models

A model with the right level of complexity will have a **lower variance** and a **lower bias**.





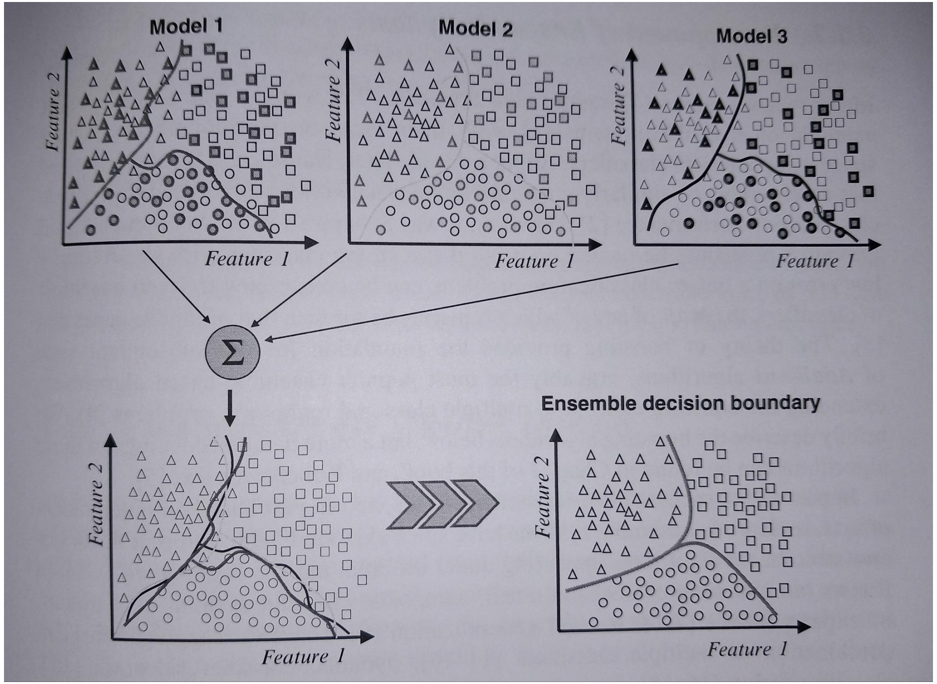
The Bias Variance Trade-off



https://medium.com/@ivanreznikov/stop-using-the-same-image-in-bias-variance-trade-off-explanation-691997a94a54

Ensemble Learning

Combining different classifiers we can obtain better decisions.



Zhang, *Ensemble Machine Learning*, 2012

Bagging (Bootstrap Aggregation)

Aims to *reduce the variance* without increasing the bias.

Main idea: Averaging reduces the variance.

Bagging:

- **Creates k version of the training dataset using bootstrap** (random sampling with replacement);
- Train k distinct classifiers, each on one version of the training dataset;
- To classify a sample, obtain the predictions of all the classifiers and choose the one chosen by the majority (**majority voting**).

Bagging is useful if we have good base classifiers with unstable decisions.

Breiman, *Bagging Predictors*, 1996, Machine Learning

Random Forest

Employ bagging with **decision trees** as the base classifiers. For each decision tree:

- Create a bootstrap dataset of the same size as the original dataset;
- Create a decision tree using the bootstrap dataset and, for each node:
  1) *randomly sample a **subset of attribute***
  2) choose the attribute that provides the best splits between them.

(Otherwise, the variance between the generated decision trees will be too low, and bagging will not provide advantages).

To evaluate the accuracy, for each dataset sample, compute the majority voting of the decision trees considering only those for which that sample was not part of the training dataset (***out-of-bag predictions***).

This procedure allows to obtain a confusion matrix.

Breiman, *Bagging Predictors*, 1996, Machine Learning

Weak Learners

Consider a classification rule h: $\mathcal{X}$ -> {-1, +1} such that h $\in \mathcal{H}$, where $\mathcal{H}$ is a class of functions from **x** to {-1, +1}.

Consider a set of samples $\{(\mathbf{x}_i, y_i), i = 1,..,N\}$ belonging to a distribution $\mathcal{D}$ such that $y_i = h(\mathbf{x}_i)$.

A classifier is defined **weak learner** if given a particular pair of $\varepsilon \geq 0$, and $\delta \leq 1/2$ it outputs with probability lower than 1 - δ a classifier f : $\mathcal{X}$ -> {-1, +1} satisfying $P_{\mathcal{D}}\,[f(\mathbf{x}) \neq h(\mathbf{x})] \leq \varepsilon$.

Informally, they are classifiers that perform slightly better than the naive model (that for classification would have the 50% of accuracy).

Can ensemble be useful if the base classifiers are weak learners?

Boosting

Aims to *reduce the bias* of week learners trying to focus on the misclassified samples.

Main idea: focusing on the errors

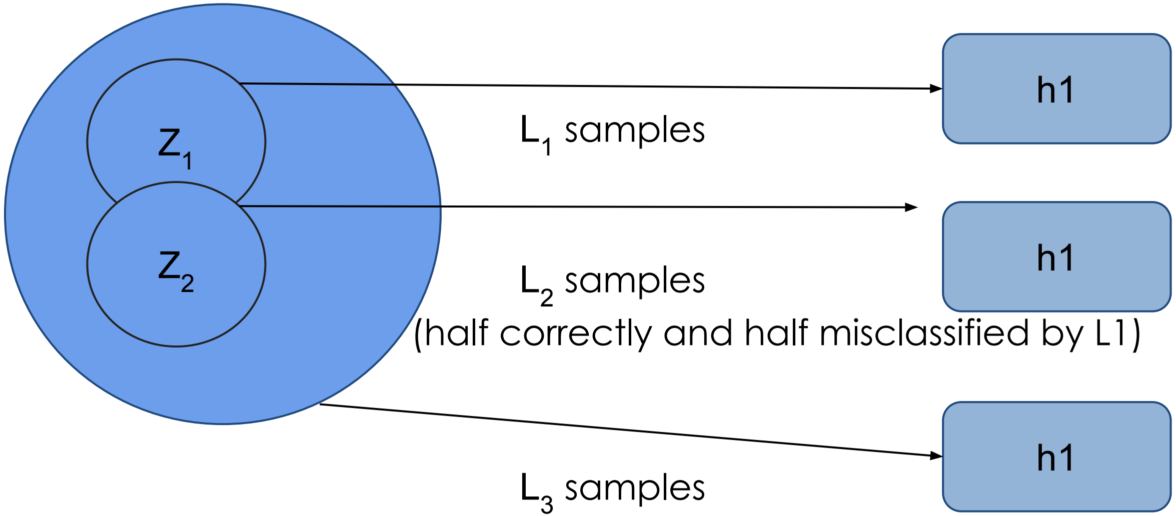
Creates three classifiers:

- h1 trained on a subset of the training dataset sampled without replacement;
- h2 is trained on a different subset of the training dataset, half of which is made by samples misclassified by h1
- h3 is trained on all the instances of the training dataset on which h1 and h2 disagree.

The decision of these classifier are then combined using majority vote.

Shapire, *The Strength of weak learnability*, 1990, Machine Learning

Boosting



L_1 samples

L_2 samples
(half correctly and half misclassified by L1)

L_3 samples
(the ones on which h1 and h2 disagree).

Ferreira et al., *Ensemble Machine Learning*, 2012

AdaBoost (Adaptive Boosting)

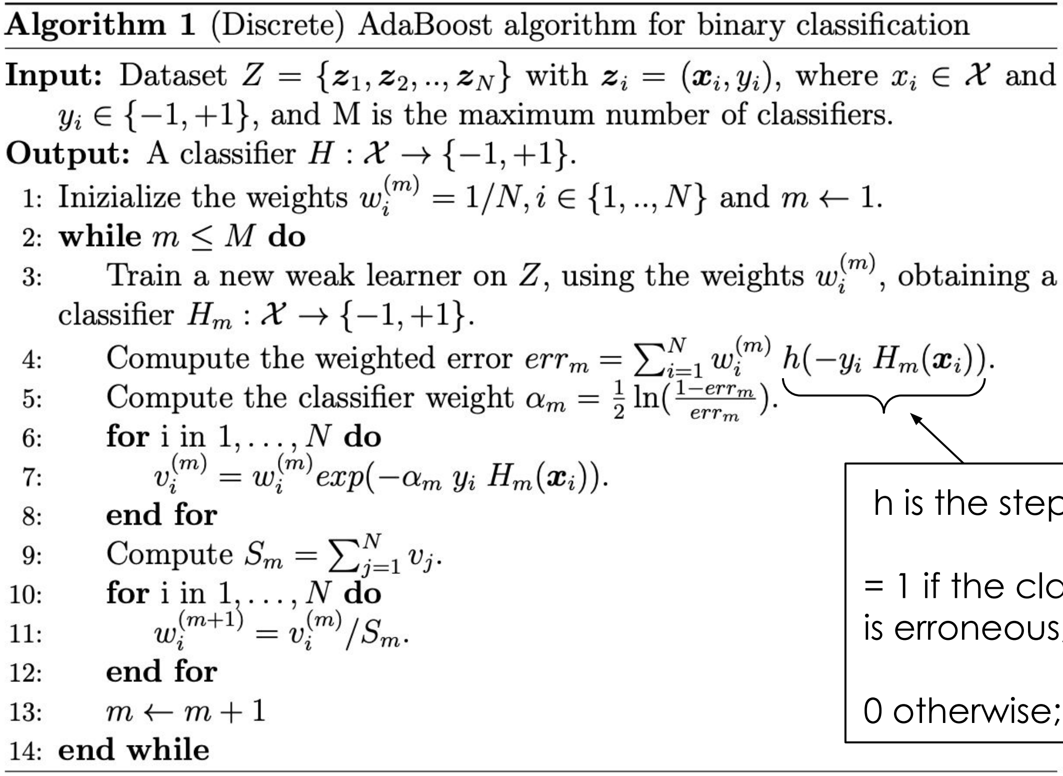
1. **Train a new weak classifier** on the weighted* samples and add it to the set of classifiers;
2. **Assign a weight to the new classifier's decisions** with a formula that allows *giving more weight to the decisions of the most accurate classifiers*.
3. **Weight the training samples** by *assigning larger weights to those that are misclassified* by the ensemble (at the first iteration a single, weak classifier);

Repeat the process.

* At the first iteration the samples will have equal weights.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)



Algorithm 1 (Discrete) AdaBoost algorithm for binary classification

```
Input: Dataset $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, .., \boldsymbol{z}_N\}$ with $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$, where $x_i \in \mathcal{X}$ and
    $y_i \in \{-1, +1\}$, and M is the maximum number of classifiers.
Output: A classifier $H : \mathcal{X} \to \{-1, +1\}$.
Inizialize the weights $w_i^{(m)} = 1/N, i \in \{1, .., N\}$ and $m \leftarrow 1$.
while $m \leq M$ do
    Train a new weak learner on $Z$, using the weights $w_i^{(m)}$, obtaining a classifier $H_m : \mathcal{X} \to \{-1, +1\}$.
    Comupute the weighted error $err_m = \sum_{i=1}^{N} w_i^{(m)} \underbrace{h(-y_i\, H_m(\boldsymbol{x}_i))}$.
    Compute the classifier weight $\alpha_m = \frac{1}{2} \ln(\frac{1-err_m}{err_m})$.
    for i in $1, \ldots, N$ do
        $v_i^{(m)} = w_i^{(m)} exp(-\alpha_m\, y_i\, H_m(\boldsymbol{x}_i))$.
    end for
    Compute $S_m = \sum_{j=1}^{N} v_j$.
    for i in $1, \ldots, N$ do
        $w_i^{(m+1)} = v_i^{(m)} / S_m$.
    end for
    $m \leftarrow m + 1$
end while
```

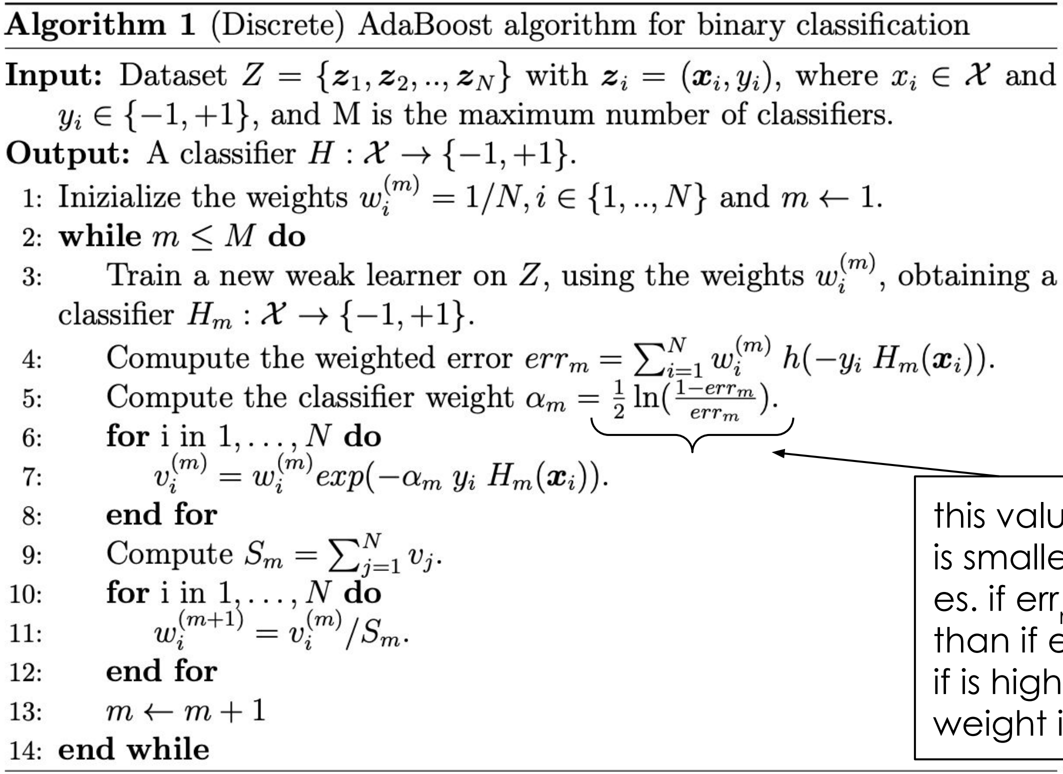
h is the step function

= 1 if the classification is erroneous;

0 otherwise;

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)



Algorithm 1 (Discrete) AdaBoost algorithm for binary classification

Input: Dataset $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, .., \boldsymbol{z}_N\}$ with $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$, and M is the maximum number of classifiers.

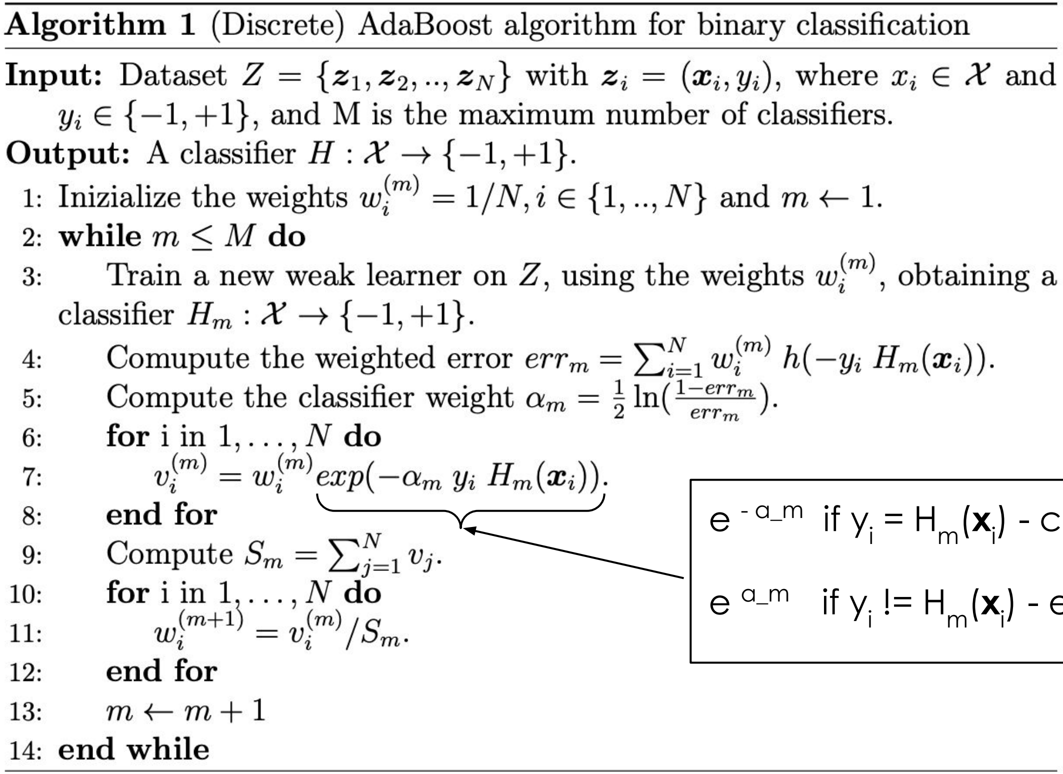
Output: A classifier $H : \mathcal{X} \rightarrow \{-1, +1\}$.

```
Inizialize the weights $w_i^{(m)} = 1/N, i \in \{1, .., N\}$ and $m \leftarrow 1$.
while $m \leq M$ do
    Train a new weak learner on $Z$, using the weights $w_i^{(m)}$, obtaining a classifier $H_m : \mathcal{X} \rightarrow \{-1, +1\}$.
    Comupute the weighted error $err_m = \sum_{i=1}^{N} w_i^{(m)}\ h(-y_i\ H_m(\boldsymbol{x}_i))$.
    Compute the classifier weight $\alpha_m = \underbrace{\frac{1}{2}\ln(\frac{1-err_m}{err_m})}$.
    for i in $1, \ldots, N$ do
        $v_i^{(m)} = w_i^{(m)} exp(-\alpha_m\ y_i\ H_m(\boldsymbol{x}_i))$.
    end for
    Compute $S_m = \sum_{j=1}^{N} v_j$.
    for i in $1, \ldots, N$ do
        $w_i^{(m+1)} = v_i^{(m)} / S_m$.
    end for
    $m \leftarrow m + 1$
end while
```

this value is higher if err_m is smaller
es. if $err_{m\,=}$0.40 is lower than if $err_{m\,=}$0.30.
if is higher than 0.50 the weight is negative

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)



Algorithm 1 (Discrete) AdaBoost algorithm for binary classification

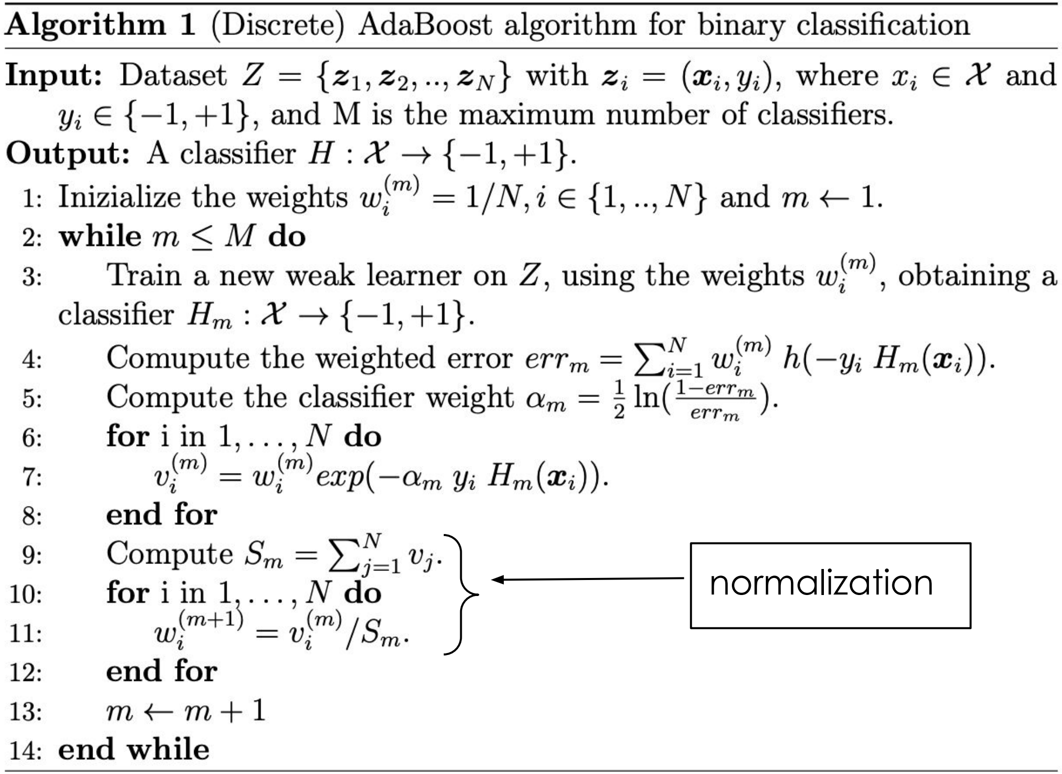
```
Input: Dataset $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, .., \boldsymbol{z}_N\}$ with $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$, where $x_i \in \mathcal{X}$ and
    $y_i \in \{-1, +1\}$, and M is the maximum number of classifiers.
Output: A classifier $H : \mathcal{X} \rightarrow \{-1, +1\}$.
Inizialize the weights $w_i^{(m)} = 1/N, i \in \{1, .., N\}$ and $m \leftarrow 1$.
while $m \leq M$ do
    Train a new weak learner on $Z$, using the weights $w_i^{(m)}$, obtaining a
    classifier $H_m : \mathcal{X} \rightarrow \{-1, +1\}$.
    Comupute the weighted error $err_m = \sum_{i=1}^{N} w_i^{(m)} \; h(-y_i \; H_m(\boldsymbol{x}_i))$.
    Compute the classifier weight $\alpha_m = \frac{1}{2} \ln(\frac{1-err_m}{err_m})$.
    for i in $1, \ldots, N$ do
        $v_i^{(m)} = w_i^{(m)} \underbrace{exp(-\alpha_m \; y_i \; H_m(\boldsymbol{x}_i))}$.
    end for
    Compute $S_m = \sum_{j=1}^{N} v_j$.
    for i in $1, \ldots, N$ do
        $w_i^{(m+1)} = v_i^{(m)} / S_m$.
    end for
    $m \leftarrow m + 1$
end while
```

$e^{\text{- a\_m}}$ if $y_i = H_m(\mathbf{x}_i)$ - correct

$e^{\text{a\_m}}$ if y_i != $H_m(\mathbf{x}_i)$ - error

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)



Algorithm 1 (Discrete) AdaBoost algorithm for binary classification

Input: Dataset $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, .., \boldsymbol{z}_N\}$ with $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$, and M is the maximum number of classifiers.

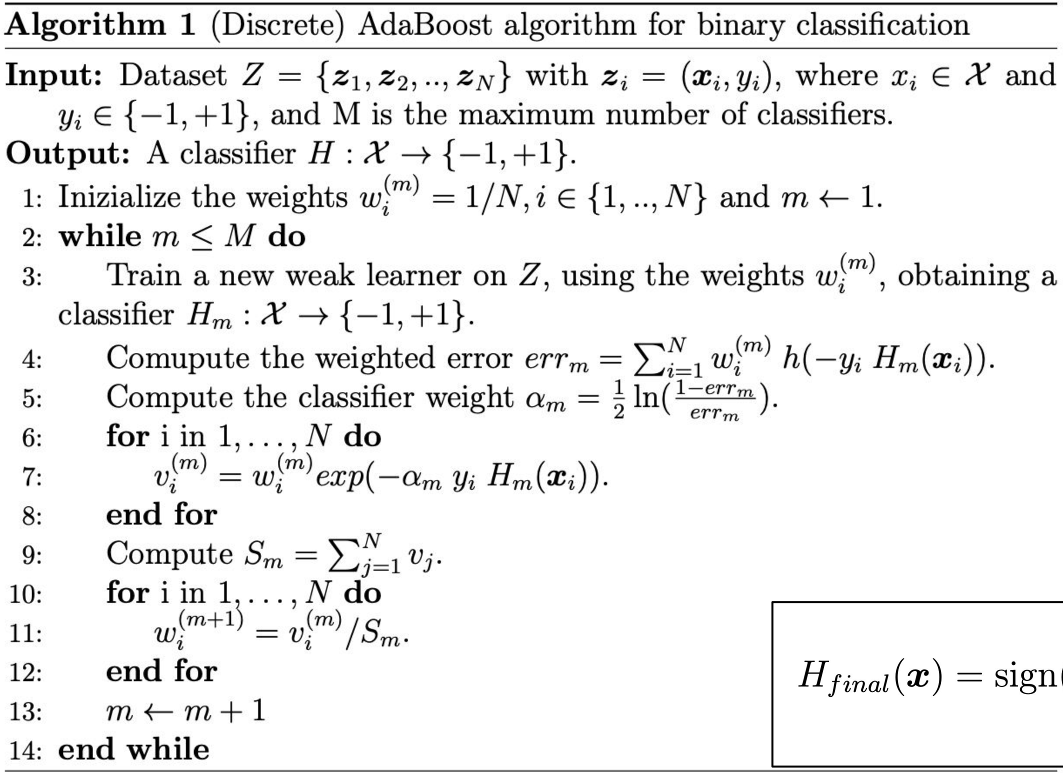
Output: A classifier $H : \mathcal{X} \rightarrow \{-1, +1\}$.

```
Inizialize the weights $w_i^{(m)} = 1/N, i \in \{1, .., N\}$ and $m \leftarrow 1$.
while $m \leq M$ do
    Train a new weak learner on $Z$, using the weights $w_i^{(m)}$, obtaining a classifier $H_m : \mathcal{X} \rightarrow \{-1, +1\}$.
    Comupute the weighted error $err_m = \sum_{i=1}^{N} w_i^{(m)} \; h(-y_i \; H_m(\boldsymbol{x}_i))$.
    Compute the classifier weight $\alpha_m = \frac{1}{2} \ln(\frac{1-err_m}{err_m})$.
    for i in $1, \ldots, N$ do
        $v_i^{(m)} = w_i^{(m)} exp(-\alpha_m \; y_i \; H_m(\boldsymbol{x}_i))$.
    end for
    Compute $S_m = \sum_{j=1}^{N} v_j$.
    for i in $1, \ldots, N$ do
        $w_i^{(m+1)} = v_i^{(m)} / S_m$.
    end for
    $m \leftarrow m + 1$
end while
```

normalization

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)



Algorithm 1 (Discrete) AdaBoost algorithm for binary classification

Input: Dataset $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, .., \boldsymbol{z}_N\}$ with $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$, and M is the maximum number of classifiers.

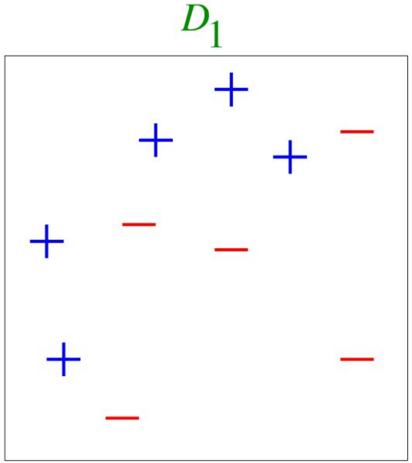
Output: A classifier $H : \mathcal{X} \rightarrow \{-1, +1\}$.

```
Inizialize the weights $w_i^{(m)} = 1/N, i \in \{1, .., N\}$ and $m \leftarrow 1$.
while $m \leq M$ do
    Train a new weak learner on $Z$, using the weights $w_i^{(m)}$, obtaining a classifier $H_m : \mathcal{X} \rightarrow \{-1, +1\}$.
    Comupute the weighted error $err_m = \sum_{i=1}^{N} w_i^{(m)} \; h(-y_i \; H_m(\boldsymbol{x}_i))$.
    Compute the classifier weight $\alpha_m = \frac{1}{2} \ln(\frac{1-err_m}{err_m})$.
    for i in $1, \ldots, N$ do
        $v_i^{(m)} = w_i^{(m)} exp(-\alpha_m \; y_i \; H_m(\boldsymbol{x}_i))$.
    end for
    Compute $S_m = \sum_{j=1}^{N} v_j$.
    for i in $1, \ldots, N$ do
        $w_i^{(m+1)} = v_i^{(m)} / S_m$.
    end for
    $m \leftarrow m + 1$
end while
```

$$\boxed{H_{final}(\boldsymbol{x}) = \text{sign}(\sum_{j=1}^{M} \alpha_j \; H_j(\boldsymbol{x}))}$$

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

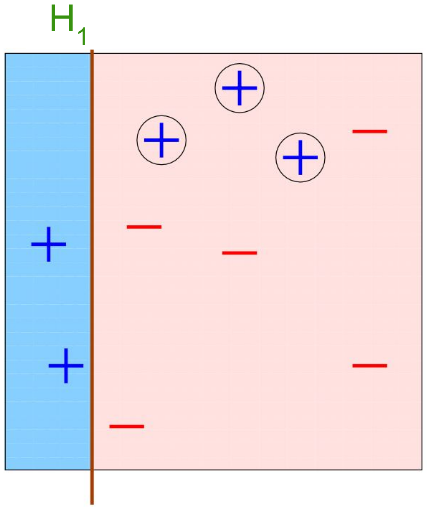
AdaBoost (Adaptive Boosting)



Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

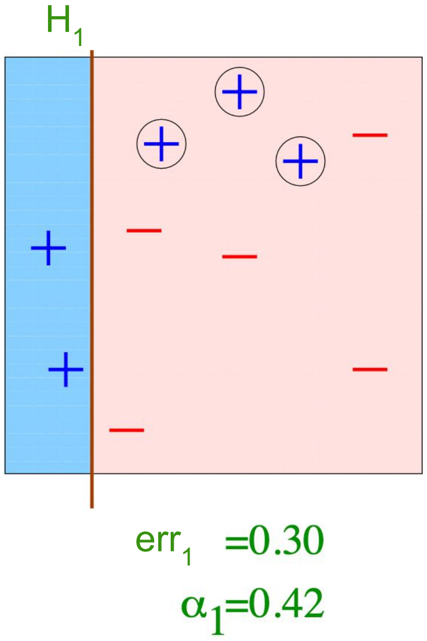
AdaBoost (Adaptive Boosting)



Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

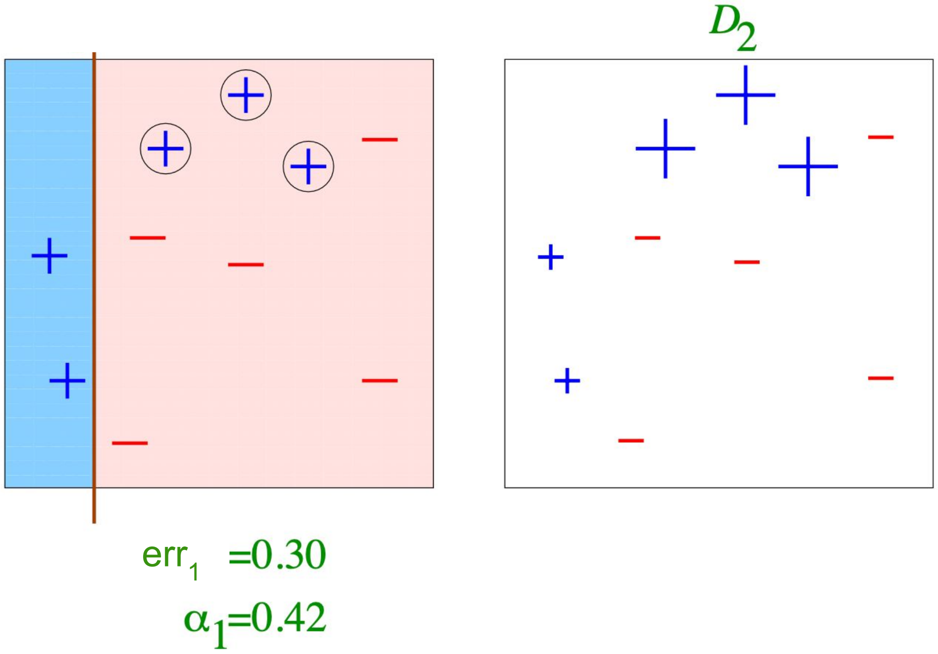
AdaBoost (Adaptive Boosting)



Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

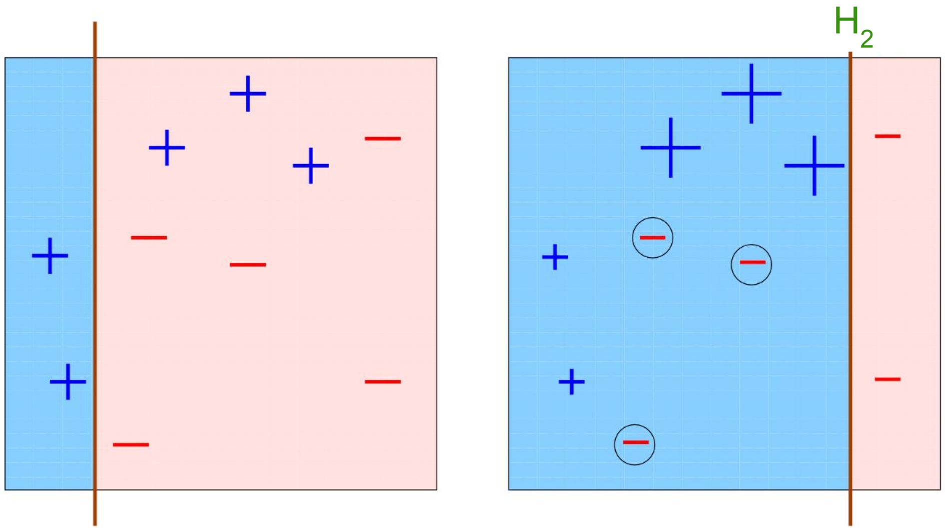
AdaBoost (Adaptive Boosting)



Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

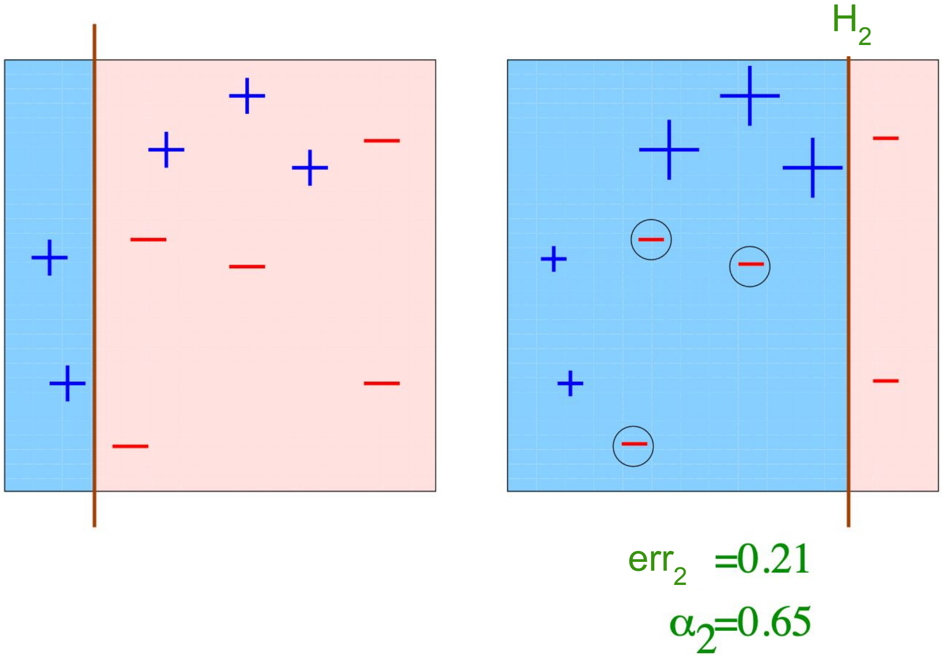
AdaBoost (Adaptive Boosting)



Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

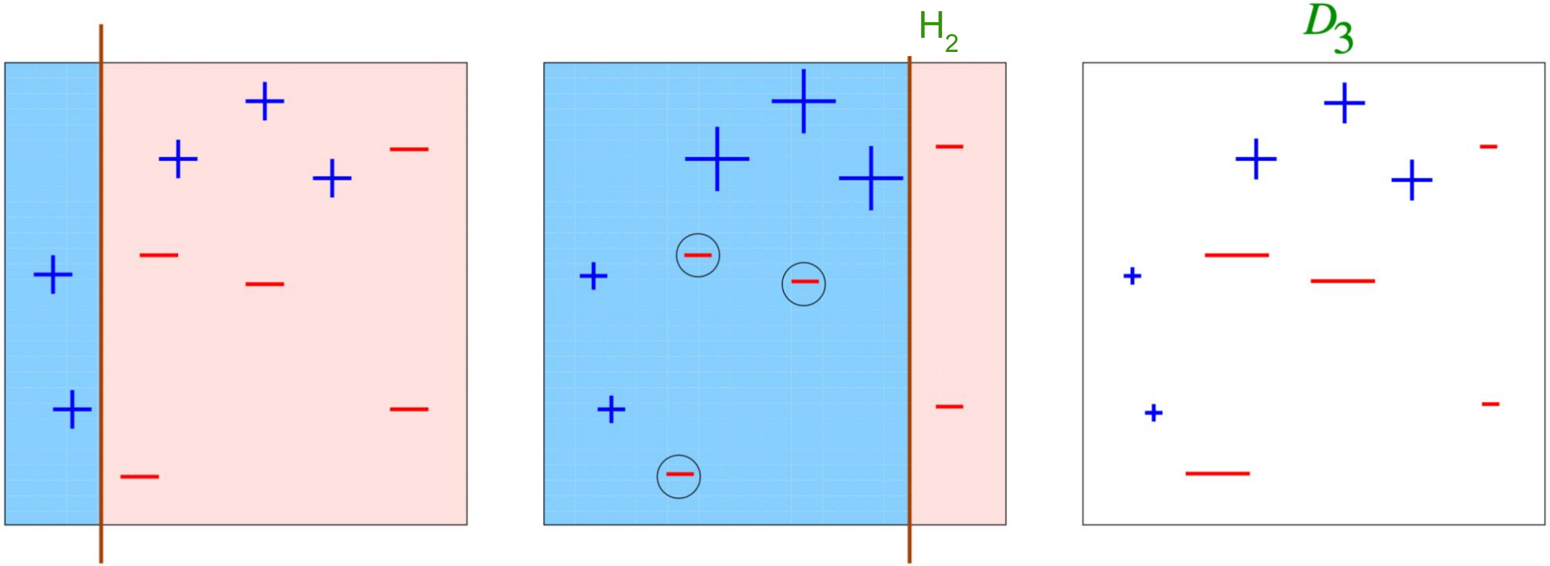
AdaBoost (Adaptive Boosting)



Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)



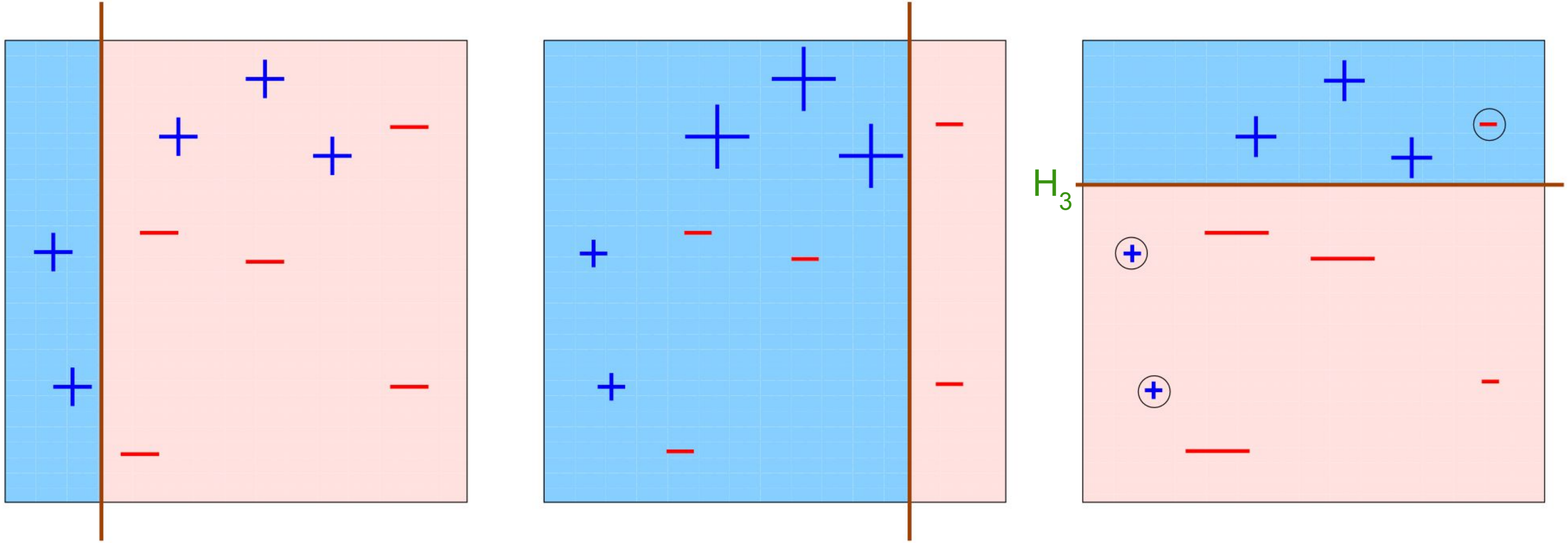
$err_2 = 0.21$

$\alpha_2 = 0.65$

Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)

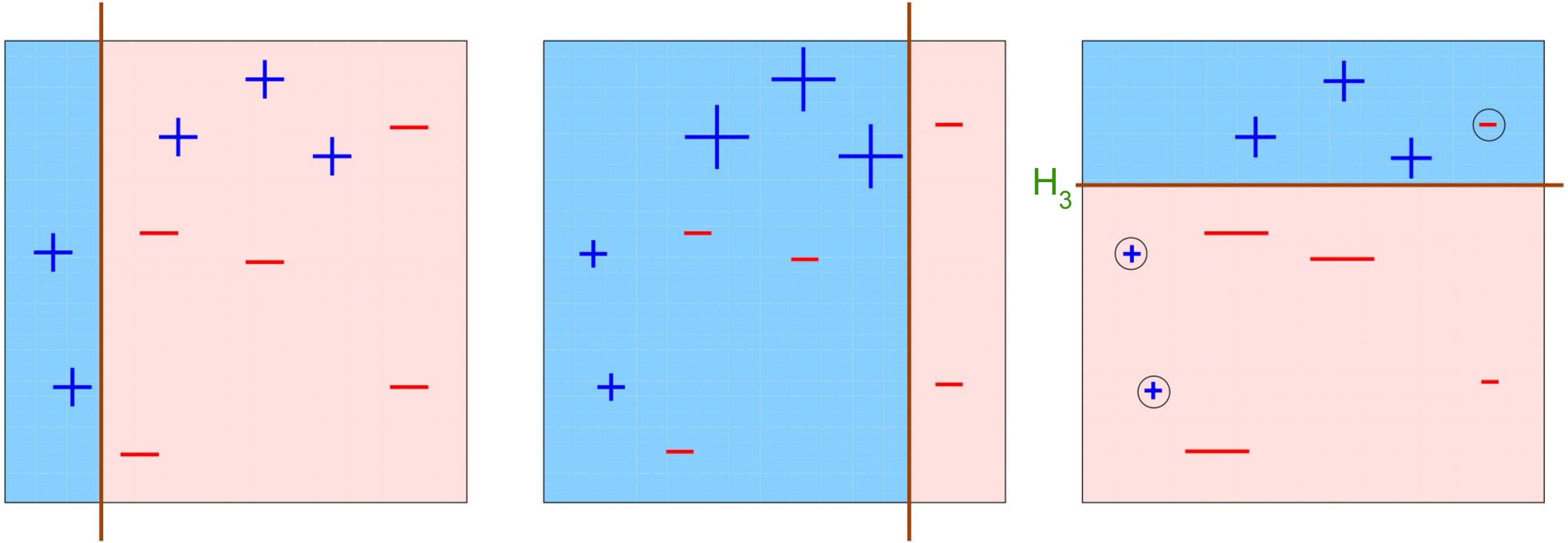


Slides from Shapire.

AI – https://unica-ai.github.io

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)



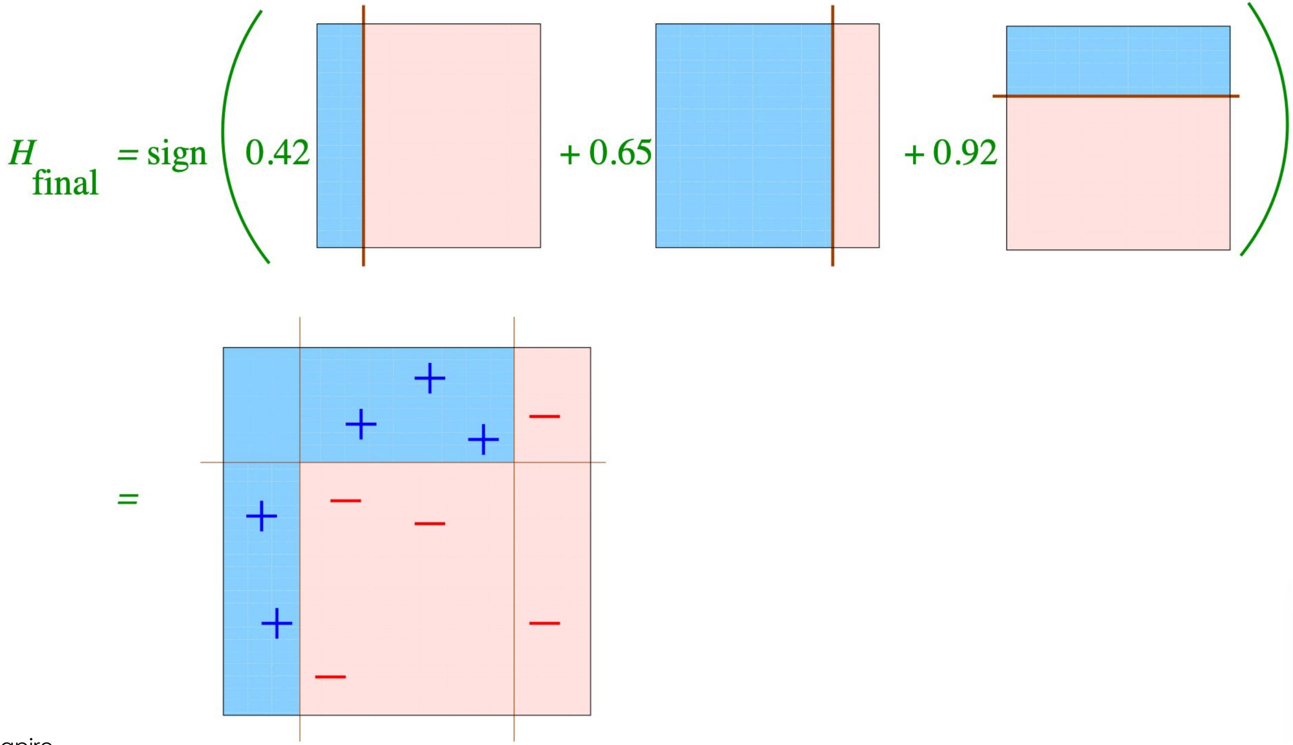
$err_3 = 0.14$

$\alpha_3 = 0.92$

Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)



Slides from Shapire.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science

AdaBoost (Adaptive Boosting)

Would a classifier that is not weak be better for AdaBoost? No!

Each iteration focuses on the instances misclassified by the previous classifiers.

If the base classifier is too strong, it may achieve high accuracy, leaving only outlier and noisy instances with significant weight to be learned in the following rounds.

Then, the classifier learned in the following round may have a high weight (because it is computed on the errors), but they may make a lot of mistakes on the samples the original classifier was able to classify correctly.

Freund and Shapire, *Decision-theoretic generalization of on-line learning and an application to boosting*, 1997, Journal of Computer and System Science