Ensembles

Dr. Daniele Cattaneo, Prof. Dr. Josif Grabocka

Machine Learning Course
Winter Semester 2023/2024

Albert-Ludwigs-Universität Freiburg

cattaneo@informatik.uni-freiburg.de, grabocka@informatik.uni-freiburg.de

04.12.2023

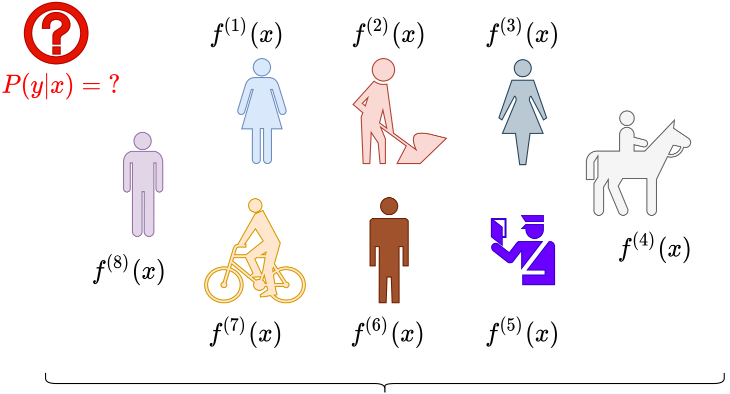
Overview

1. Bagging
2. Boosting

Table of Contents

Motivation

- **Ensembles** are aggregations of multiple **base** models



$$P(y|x) = \text{aggregate}(f^{(1)}(x), \ldots, f^{(K)}(x)), \quad \text{e.g. } \hat{y} = \frac{1}{K}\sum_{k=1}^{K} f^{(k)}(x)$$

Variance Reduction

The Bias-Variance Decomposition of $E_{x,y,D}\left[\left(\hat{f}(x;D)-y\right)^2\right]=$

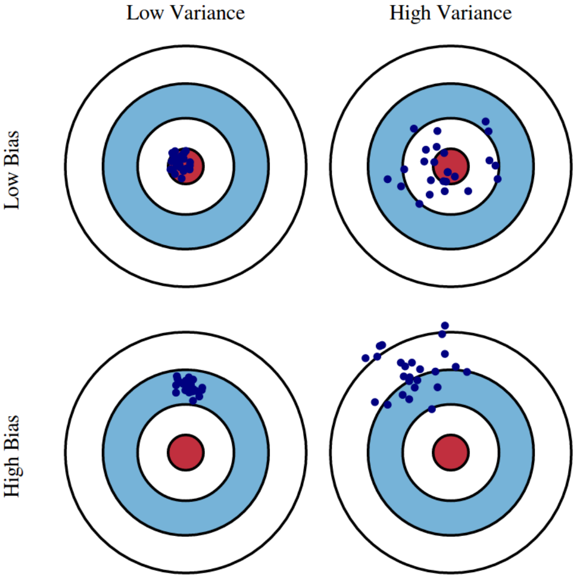
$$\underbrace{E_{x,D}\left[\left(\hat{f}(x;D)-\bar{f}(x)\right)^2\right]}_{\text{Variance}}+\underbrace{E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)^2\right]}_{\text{Bias}^2}+\underbrace{E_{x,y}\left[\left(\bar{y}(x)-y\right)^2\right]}_{\text{Noise}}$$

where the expected prediction model $\bar{f}(x)$ is:

$$\bar{f}(x):=E_{D\sim\mathcal{P}^N}\left[\hat{f}(x;D)\right]=\int_D \hat{f}(x;D)\,p(D)\,dD$$

Therefore, variance is reduced if: $\hat{f}(x;D)\rightarrow\bar{f}(x),\ \forall D$

Bias-Variance Decomposition Illustration



Ensemble of multiple I.I.D. training sets

- Given $D^{(1)}, \dots, D^{(K)}$ multiple (i.i.d.) training sets

$$D^{(k)} \in (\mathcal{X} \times \mathcal{Y})^N, \forall k \in \{1, \dots, K\}$$

- Train one prediction model on each training set

$$\hat{f}^{(k)}(x) = \hat{f}(x, \theta^{(k)}) \quad \text{s.t.} \quad \theta^{(k)} = \operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(y_i^{(k)}, \hat{f}(x_i^{(k)}, \theta)\right)^2$$

- Compute an **ensemble** model:

$$\hat{f}(x) = \frac{1}{K} \sum_{k=1}^{K} \hat{f}^{(k)}(x)$$

Ensemble reduces variance

- Following the law of large numbers $\bar{z} = \frac{z_1+z_2+\cdots+z_K}{K} = E[z]$

$$\left[\hat{f}(x) = \frac{1}{K}\sum_{k=1}^{K} \hat{f}^{(k)}(x)\right] \rightarrow \left[\bar{f}(x) := E_{D\sim\mathcal{P}^N}\left[\hat{f}(x;D)\right]\right]$$

- Replacing the single model with an average ensemble reduces the variance:

$$E_{x,D}\left[\left(\hat{f}(x;D) - \bar{f}(x)\right)^2\right] = E_x\left[\left(\hat{f}(x) - \bar{f}(x)\right)^2\right] \rightarrow 0$$

Ensemble reduces variance

- Following the law of large numbers $\bar{z} = \frac{z_1+z_2+\cdots+z_K}{K} = E[z]$

$$\left[\hat{f}(x) = \frac{1}{K}\sum_{k=1}^{K}\hat{f}^{(k)}(x)\right] \to \left[\bar{f}(x) := E_{D\sim\mathcal{P}^N}\left[\hat{f}(x;D)\right]\right]$$

- Replacing the single model with an average ensemble reduces the variance:

$$E_{x,D}\left[\left(\hat{f}(x;D) - \bar{f}(x)\right)^2\right] = E_x\left[\left(\hat{f}(x) - \bar{f}(x)\right)^2\right] \to 0$$

- But in reality we do not have K different i.i.d. training sets, we just have **one**!

Bootstrap Aggregation (Idea)

Generate K *diverse* training sets by sampling from one training set.

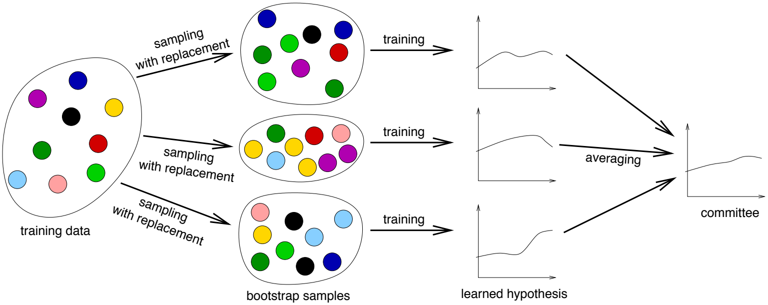


Figure 1: Source: Criminisi et al., 2013

Bootstrap Aggregation (Bagging)

- $D^{(1)}, \ldots, D^{(K)}$ **with replacement** from $D^{(Train)} \in (\mathcal{X} \times \mathcal{Y})^N$
  - Distribution of sampling one instance $\mathcal{I}((x, y) \mid D^{(Train)}) = \frac{1}{N}$
  - Sample a bootstrap $D^{(k)} \sim \mathcal{I}^n$ with repl., $n = |D^{(k)}|, \ n \leq N$
- Train one model per-bootstrap and aggregate an ensemble:

$$\hat{f}(x) = \frac{1}{K} \sum_{k}^{K} \hat{f}^{(k)}(x)$$

- Notice that $D^{(k)}, \forall k$ are not i.i.d. since they share instances:

$$\hat{f}(x) \not\to \bar{f}, \ \ E_x \left[\left(\hat{f}(x) - \bar{f}(x) \right)^2 \right] \not\to 0$$

Bagging requires uncorrelated models

- Models make errors $\epsilon_k, k = 1, \ldots, K$ in a regression task:
  - Assume ϵ_k is drawn from a multivariate normal distribution with mean 0, variance $E\left[\epsilon_k^2\right] = v$ and covariances $E\left[\epsilon_k \epsilon_\ell\right] = c$
- The overall error of an ensemble is $\frac{1}{K}\sum_{k=1}^{K} \epsilon_k$
- The expected squared error of the ensemble is:

$$E\left[\left(\frac{1}{K}\sum_{k=1}^{K}\epsilon_k\right)^2\right] = \frac{1}{K^2}E\left[\sum_{k=1}^{K}\left(\epsilon_k^2 + \sum_{k\neq\ell}\epsilon_k\epsilon_\ell\right)\right] = \frac{1}{K}v + \frac{K-1}{K}c$$

- (A) If errors are correlated, $c = v$ then squared error is v
- (B) If errors are uncorrelated, $c = 0$ then squared error is $\frac{v}{k}$
- In (A) ensemble doesn't help and in (B) the error is reduced linearly

Bagged Trees

Bagging needs:

- Low bias individual models
- Uncorrelated individual models

Trees are low bias but might be correlated. Further decorrelation is achieved by randomizing the choice of features to split upon.

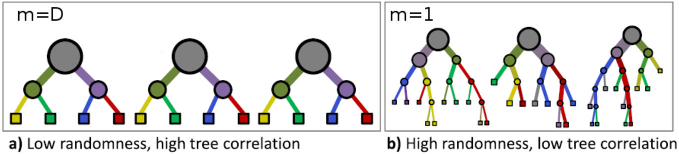


Figure 2: Source: Criminisi et al., 2013

Random Forests

Random Forests are Bagged Trees with randomness in terms of the split conditions.

Algorithm 1: Random Forest

1: **for** $k = 1, \ldots, K$ **do**

2: Draw bootstrap $D^{(k)}$ with replacement from $D^{(\text{Train})}$

3: Train tree $f^{(k)}(x)$ through CART with a randomized split search for every node:

- a) Select m features at random from the M features
- b) Pick the best decision split among the m features

4: $\hat{f}(x) = \frac{1}{K} \sum_k^K \hat{f}^{(k)}(x)$

Random Forests Illustration

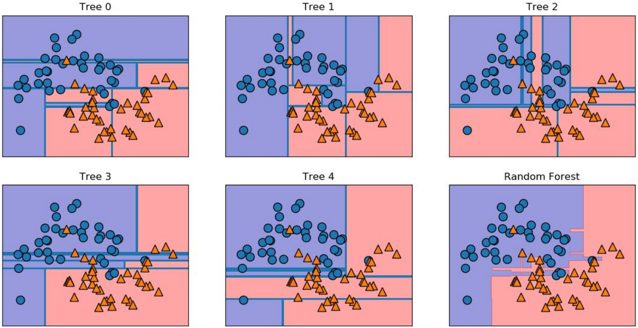


Figure 3: Source: Mueller et al., 2017

Uncertainty and Out-of-Bag Error

- The variance of an ensemble can be used as uncertainty:

$$\sigma(x) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(\hat{f}^{(k)}(x) - \hat{f}(x)\right)^2}$$

- Bagged ensembles produce an estimate of the **test error** via the **out-of-bag training error**:

$$S(x) = \left\{k \in \{1, \ldots, K\} \mid x \notin \pi_x\left(D^{(k)}\right)\right\}$$

$$\epsilon_{OOB} = \frac{1}{N}\sum_{n=1}^{N}\ell\left(y_n, \frac{1}{|S(x_n)|}\sum_{k \in S(x_n)} f^{(k)}(x_n)\right)$$

Table of Contents

Boosting as Additive Models

- Boosting sequentially aggregates **high-bias models** to create a **low-bias ensemble**
- Ensemble $F^{(k)}$ is a weighted sum of $f^{(k)}$, $\forall k \in \{1, \dots, K\}$

$$F^{(K)}(x) = \sum_{k=1}^{K} \alpha f^{(k)}(x)$$

- We add one model at a time sequentially:

$$F^{(K+1)}(x) = \sum_{k=1}^{K+1} \alpha f^{(k)}(x) = F^{(K)}(x) + \alpha f^{(K+1)}(x)$$

- Train the parameters of the next $(K+1)$-th model by keeping the past models fixed:

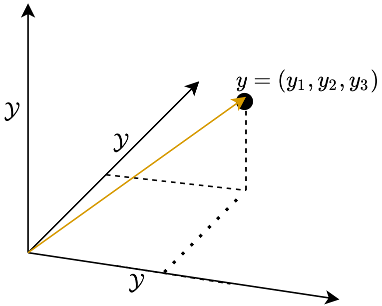
$$\theta^{(K+1)} =: \operatorname*{argmin}_{\theta} \sum_{n=1}^{N} \ell\left(y_n, F^{(K)}(x_n) + \alpha f(x_n; \theta)\right)$$

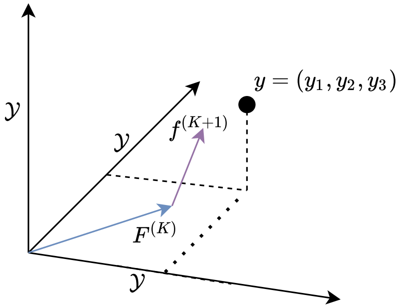
Boosting in Functional Space

- Rewriting the objective as $\ell(F^{(K)}) = \sum_{n=1}^{N} \ell\left(y_n, F^{(K)}(x_n)\right)$ yields:

$$f^{(K+1)} = \underset{f \in \mathbb{H}}{\text{argmin}}\ \ell\left(F^{(K)} + \alpha f\right)$$

- The objective has a geometric intepretation:





Gradient Descent and the Taylor Approximation

- We want to find the update value Δw to weights w
- The loss can be linearly approximated via the first-order Taylor expansion:

$$\ell(w + \Delta w) \approx \ell(w) + \frac{d\ell(w)}{dw}\Delta w$$

- If we set $\Delta w = -\eta \frac{d\ell(w)}{dw}$, then:

$$\ell(w + \Delta w) = \ell(w) - \eta \left(\frac{d\ell(w)}{dw}\right)^2 \leq \ell(w)$$

- In case of a multivariate w:

$$\ell(w + \Delta w) = \ell(w) + \langle \nabla_w \ell(w), \Delta w \rangle$$

Taylor Approximation of the Ensemble Loss

- Expand using the first-order Taylor approximation

$$\ell(F^{(K)} + \alpha f) = \ell(F^{(K)}) + \alpha \langle \nabla \ell(F^{(K)}), f \rangle$$

- Leading to the following optimization:

$$\begin{aligned} f^{(K+1)} &= \underset{f \in \mathbb{H}}{\operatorname{argmin}}\, \ell(F^{(K)} + \alpha f) = \underset{f \in \mathbb{H}}{\operatorname{argmin}} \langle \nabla \ell(F^{(K)}), f \rangle \\ &= \underset{f \in \mathbb{H}}{\operatorname{argmin}} \sum_{n=1}^{N} \frac{\partial \ell(F^{(K)})}{\partial F^{(k)}(x_n)} f(x_n) \end{aligned}$$

- Where:

$$\frac{\partial \ell(F^{(K)})}{\partial F^{(k)}(x_n)} = \frac{\partial \left(\sum_{i=1}^{N} \ell \left(y_i, F^{(k)}(x_i) \right) \right)}{\partial F^{(k)}(x_n)} = \frac{\partial \ell \left(y_n, F^{(k)}(x_n) \right)}{\partial F^{(k)}(x_n)}$$

Converting to Regression of the Negative Gradients

- Introducing the notation $z_n = -\frac{\partial \ell\left(y_n, F^{(k)}(x_n)\right)}{\partial F^{(k)}(x_n)}$:

$$f^{(K+1)} = \operatorname*{argmin}_{f \in \mathbb{H}} \sum_{n=1}^{N} -z_n\, f(x_n)$$

- Multiply with 2 and add the constant $\sum_{n=1}^{N} z_n^2 = c$:

$$f^{(K+1)} = \operatorname*{argmin}_{f \in \mathbb{H}} \sum_{n=1}^{N} -2z_n\, f(x_n) + z_n^2$$

- Assuming $\sum_{n=1}^{N} (f(x_n))^2 = c$ we achieve:

$$\begin{aligned} f^{(K+1)} &= \operatorname*{argmin}_{f \in \mathbb{H}} \sum_{n=1}^{N} (f(x_n))^2 - 2z_n\, f(x_n) + z_n^2 \\ &= \operatorname*{argmin}_{f \in \mathbb{H}} \sum_{n=1}^{N} (f(x_n) - z_n)^2 \end{aligned}$$

Gradient Boosting Pseudo-code

Algorithm 2: Gradient Boosting

1: $\forall n \in \{1, \ldots, N\} : F^{(0)}(x_n) = 0$

2: **for** $k = 1, \ldots, K-1$ **do**

3: $\quad \forall n \in \{1, \ldots, N\} : z_n = -\frac{\partial \ell\left(y_n, F^{(k)}(x_n)\right)}{\partial F^{(k)}(x_n)}$

4: $\quad$ Train $f^{(k)} := \text{argmin}_{f \in \mathbb{H}} \sum_{n=1}^{N} \left(f(x_n) - z_n\right)^2$

5: $\quad \forall n \in \{1, \ldots, N\} : F^{(k+1)}(x_n) = F^{(k)}(x_n) + \alpha^{(k)}\ f^{(k)}(x_n)$

6: **return** $F^{(K)}$

Inference on a test instance x' is $F^{(k)}(x') = \sum_{k=1}^{K} \alpha^{(k)}\ f^{(k)}(x')$

AdaBoost [Freund and Schapire, 1995]

Iterative $f^{(k+1)} \in \{-1,+1\}^N$ fixing the errors of $F^{(k)} \in \{-1,+1\}^N$

- One weight w_i for each data point $(x_i, y_i), \forall i \in \{1, \dots, N\}$
  - w_i measures how hard data point x_i is to predict
  - Start with $w_i^{(1)} = \frac{1}{N}$ and update $w_i^{(k)}$, $\forall k \in \{1, \dots, K\}$
  - In each iteration increase weights of missclassified instances:

$$
\begin{aligned}
w_i^{(k+1)} \quad &:= \quad \begin{cases} w_i^{(k)} \times \exp(\alpha_k) & , \quad \text{if } y_i \neq f^{(k)}(x_i) \\ w_i^{(k)} & , \quad \text{otherwise} \end{cases} \\
&= \quad w_i^{(k)} \exp(\alpha_k \, \mathbb{I}(y_i \neq f^{(k)}(x_i))
\end{aligned}
$$

- We'll also compute a weight α_k for each submodel $f^{(k)}$
  - This depends on how good the model is
  - Use these weights α_k for a weighted majority ensemble

AdaBoost: Weighted Ensemble

- Ensemble $F^{(K)}(x)$ combines $f^{(1)}, \dots, f^{(K)}$ through a weighted majority vote with $\alpha_1, \dots, \alpha_K$:

$$F^{(K)}(x) = \mathrm{sign}\left(\sum_{k=1}^{K} \alpha_k f^{(k)}(x)\right)$$

- $f^{(k)}$ are weighted depending on their error rates Err_k:

$$\alpha_k := \log \frac{1 - \mathrm{Err}_k}{\mathrm{Err}_k}$$

  - E.g., $\mathrm{Err}_k = 0.1 \rightarrow \alpha_k = 2.197$
  - E.g., $\mathrm{Err}_k = 0.4 \rightarrow \alpha_k = 0.41$
  - E.g., $\mathrm{Err}_k = 0.5 \rightarrow \alpha_k = 0$
  - (log refers to the natural logarithm)

Weighted Error Rate

- The (unweighted) training error rate of a submodel $f^{(k)}$ is

$$\overline{\mathrm{Err}}_k = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(y_i \neq f^{(k)}(x_i)).$$

- The weighted training error rate of submodel $f^{(k)}$ is

$$\mathrm{Err}_k = \frac{\sum_{i=1}^{N} w_i^{(k)} \ \mathbb{I}(y_i \neq f^{(k)}(x_i))}{\sum_{i=1}^{N} w_i^{(k)}}$$

- As mentioned, these weighted training error rates are used for computing the model weights:

$$\alpha_k := \log \frac{1-\mathrm{Err}_k}{\mathrm{Err}_k}$$

Training a New Model

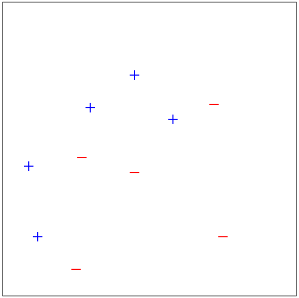
- AdaBoost uses an instance-penalty loss:

$$
\begin{aligned}
f^{(k)} &:= \operatorname*{argmin}_{f \in \mathbb{H}} \frac{\sum_{i=1}^{N} w_i^{(k)} \, \mathbb{I}(y_i \neq f(x_i))}{\sum_{i=1}^{N} w_i^{(k)}} \\
&\approx \operatorname*{argmin}_{f \in \mathbb{H}} \sum_{i=1}^{N} w_i^{(k)} \, \ell(y_i, f(x_i)) \\
&\approx \operatorname*{argmin}_{f \in \mathbb{H}} \sum_{i=1}^{N} w_i^{(k)} \, e^{-y_i f(x_i)}
\end{aligned}
$$

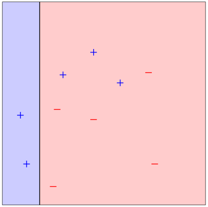
- Notice that $w^{(k)}$ depend on the performance of $F^{(k)}$

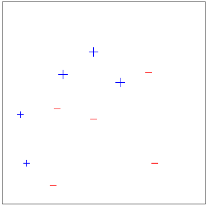
AdaBoost Example (step $k = 1$)

Example taken from [Schapire, 2003]
Model class: simple axis-aligned splits (decision stumps)



AdaBoost Example (step $k = 1$)





Train a decision stump:

$$f^{(1)} := \underset{f \in \mathbb{H}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(1)} \; e^{-y_i f(x_i)}$$

$$:= \frac{1}{N} \underset{f \in \mathbb{H}}{\operatorname{argmin}} \sum_{i=1}^{N} \; e^{-y_i f(x_i)}$$

AdaBoost Example (step $k = 1$ details)

- Model error and model weight:

$$\mathrm{Err}_1 = \sum_{i=1}^{N} w_i^{(1)}\, \mathbb{I}(f^{(1)}(x_i) \neq y_i) = \frac{1}{10} \times 3 = 0.3$$

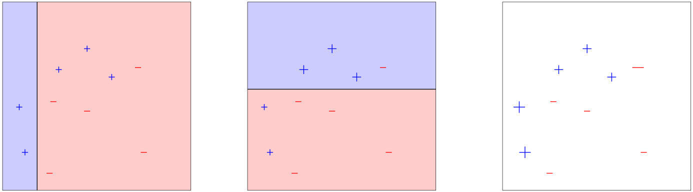
$$\alpha_1 = \log \frac{1 - \mathrm{Err}_1}{\mathrm{Err}_1} = \log \frac{1 - 0.3}{0.3} \approx 0.847$$

- Weight adaptation for data points:

$$w_i^{(k+1)} = w_i^{(k)} \exp(\alpha_k\, \mathbb{I}(y_i \neq f^{(k)}(x_i))$$

  - Misclassified data point: $w_i^{(2)} \leftarrow w_i^{(1)} \exp(\alpha_1) \approx 0.1 \exp(0.847) \approx 0.233$
  - Correctly classified data points: $w_i^{(2)} \leftarrow w_i^{(1)} = 0.1$

AdaBoost Example (step $k = 2$)



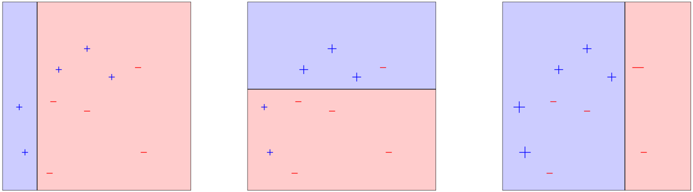
Train a second decision stump:

$$f^{(2)} := \operatorname*{argmin}_{f \in \mathbb{H}} \sum_{i=1}^{N} w_i^{(2)} \, e^{-y_i f(x_i)}$$

Update: Err_2, α_2 and $w^{(3)}$ (omitted):

$$\mathrm{Err}_2 = \frac{\sum_{i=1}^{N} w_i^{(2)} \, \mathbb{I}(f^{(2)}(x_i) \neq y_i)}{\sum_{i=1}^{N} w_i^{(2)}} \approx 0.21, \;\; \alpha_2 = \log \frac{1 - \mathrm{Err}_2}{\mathrm{Err}_2} \approx 1.3$$

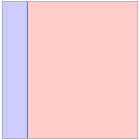
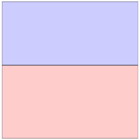
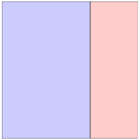
AdaBoost Example (step $k = 3$)

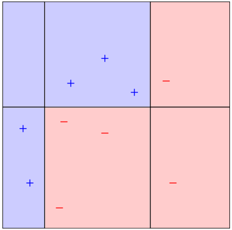


$\mathrm{Err}_3 \approx 0.14, \alpha_3 \approx 1.84$

Ensemble

Final classifier:

G = sign(+0.84  +1.3  +1.84 )

= 

Algorithm

Algorithm 3: AdaBoost

1: $\forall i \in \{1, \ldots, N\} : w_i = \frac{1}{N}$

2: **for** $k = 1, \ldots, K$ **do**

3: Train: $f^{(k)} := \operatorname{argmin}_{f \in \mathbb{H}} \sum_{i=1}^{N} w_i^{(k)} \; e^{-y_i f(x_i)}$

4: Compute: $\mathrm{Err}_k = \frac{\sum_{i=1}^{N} w_i \;\; \mathbb{I}(y_i \neq f^{(k)}(x_i))}{\sum_{i=1}^{N} w_i}$

5: Compute: $\alpha_k := \log \frac{1 - \mathrm{Err}_k}{\mathrm{Err}_k}$

6: Update: $\forall i \in \{1, \ldots, N\} : w_i \leftarrow w_i \; \exp(\alpha_k \, \mathbb{I}(y_i \neq f^{(k)}(x_i))$

7: **return** $F^{(K)}(x) = \operatorname{sign}\left(\sum_{k=1}^{K} \alpha_k \, f^{(k)}(x)\right)$