



STK-IN4300

Statistical Learning Methods in Data Science

Riccardo De Bin

`debin@math.uio.no`

Outline of the lecture

- AdaBoost
 - Introduction
 - algorithm
- Statistical Boosting
 - Boosting as a forward stagewise additive modelling
 - Why exponential loss?
 - Steepest Descent
 - Gradient boosting

AdaBoost: introduction

L. Breiman: “[*Boosting is*] the best off-shelf classifier in the world”.

- originally developed for classification;
- as a pure machine learning black-box;
- translated into the statistical world (Friedman et al., 2000);
- extended to every statistical problem (Mayr et al., 2014),
 - ▶ regression;
 - ▶ survival analysis;
 - ▶ ...
- interpretable models, thanks to the statistical view;
- extended to work in high-dimensional settings (Bühlmann, 2006).

AdaBoost: introduction

Starting challenge:

“Can [a committee of blockheads] somehow arrive at **highly reasoned decisions**, despite the **weak judgement** of the individual members?” (Schapire & Freund, 2014)

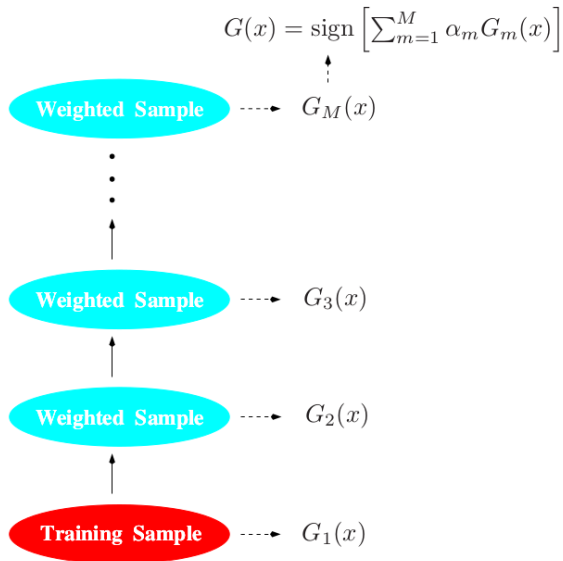
Goal: create a good classifier by **combining several weak** classifiers,

- in classification, a “weak classifier” is a classifier which is able to produce results **only slightly better** than a random guess;

Idea: apply repeatedly (**iteratively**) a weak classifier to **modifications of the data**,

- at each iteration, give **more weight** to the misclassified observations.

AdaBoost: introduction



AdaBoost: algorithm

Consider a **two-class** classification problem, $y_i \in \{-1, 1\}$, $x_i \in \mathbb{R}^p$.

AdaBoost algorithm:

1. **initialize** the weights, $w^{[0]} = (1/N, \dots, 1/N)$;
2. for m from 1 to m_{stop} ,
 - (a) **fit the weak estimator** $G(\cdot)$ to the weighted data;
 - (b) compute the weighted in-sample **misclassification rate**,
$$\text{err}^{[m]} = \sum_{i=1}^N w_i^{[m-1]} \mathbb{1}(y_i \neq \hat{G}^{[m]}(x_i));$$
 - (c) compute the **voting weights**, $\alpha^{[m]} = \frac{1}{2} \log((1 - \text{err}^{[m]})/\text{err}^{[m]})$;
 - (d) **update** the weights
 - ▶ $\tilde{w}_i = w_i^{[m-1]} \exp\{\alpha^{[m]} \mathbb{1}(y_i \neq \hat{G}^{[m]}(x_i))\}$;
 - ▶ $w_i^{[m]} = \tilde{w}_i / \sum_{i=1}^N \tilde{w}_i$;
3. compute the **final result**,

$$\hat{G}_{\text{AdaBoost}} = \text{sign}\left(\sum_{m=1}^{m_{\text{stop}}} \alpha^{[m]} \hat{G}^{[m]}(x)\right)$$

AdaBoost: example

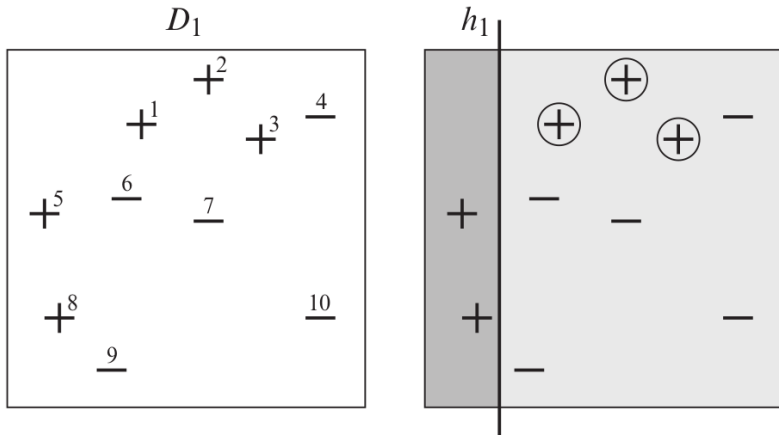


figure from Schapire & Freund (2014)

AdaBoost: example

First iteration:

- apply the classifier $G(\cdot)$ on observations with weights:

	1	2	3	4	5	6	7	8	9	10
w_i	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10

- observations 1, 2 and 3 are **misclassified** $\Rightarrow \text{err}^{[1]} = 0.3$;
- compute $\alpha^{[1]} = 0.5 \log((1 - \text{err}^{[1]})/\text{err}^{[1]}) \approx 0.42$;
- set $w_i = w_i \exp\{\alpha^{[1]} \mathbb{1}(y_i \neq \hat{G}^{[1]}(x_i))\}$:

	1	2	3	4	5	6	7	8	9	10
w_i	0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07

AdaBoost: example

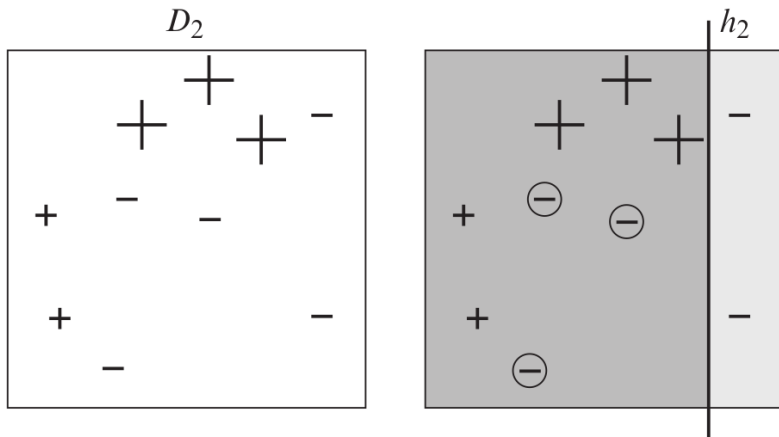


figure from Schapire & Freund (2014)

AdaBoost: example

Second iteration:

- apply classifier $G(\cdot)$ on **re-weighted** observations ($w_i / \sum_i w_i$):

	1	2	3	4	5	6	7	8	9	10
w_i	0.17	0.17	0.17	0.07	0.07	0.07	0.07	0.07	0.07	0.07

- observations 6, 7 and 9 are **misclassified** $\Rightarrow \text{err}^{[2]} \approx 0.21$;
- compute $\alpha^{[2]} = 0.5 \log((1 - \text{err}^{[2]})/\text{err}^{[2]}) \approx 0.65$;
- set $w_i = w_i \exp\{\alpha^{[2]} \mathbb{1}(y_i \neq \hat{G}^{[2]}(x_i))\}$:

	1	2	3	4	5	6	7	8	9	10
w_i	0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04

AdaBoost: example

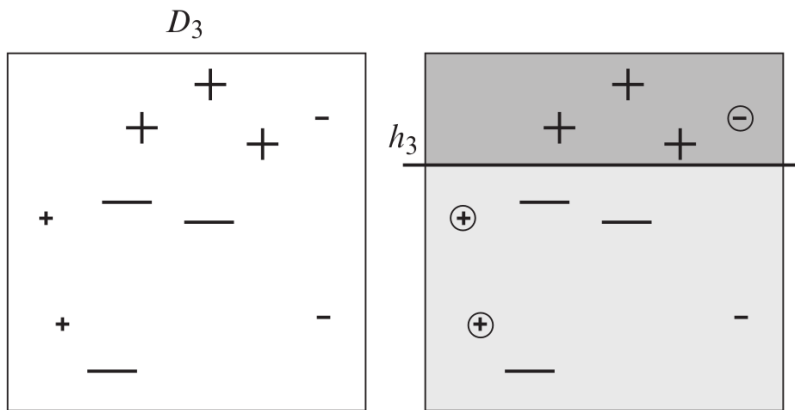


figure from Schapire & Freund (2014)

AdaBoost: example

Third iteration:

- apply classifier $G(\cdot)$ on **re-weighted** observations ($w_i / \sum_i w_i$):

	1	2	3	4	5	6	7	8	9	10
w_i	0.11	0.11	0.11	0.05	0.05	0.17	0.17	0.05	0.17	0.05

- observations 4, 5 and 8 are **misclassified** $\Rightarrow \text{err}^{[3]} \approx 0.14$;
- compute $\alpha^{[3]} = 0.5 \log((1 - \text{err}^{[3]}) / \text{err}^{[3]}) \approx 0.92$;
- set $w_i = w_i \exp\{\alpha^{[3]} \mathbb{1}(y_i \neq \hat{G}^{[3]}(x_i))\}$:

	1	2	3	4	5	6	7	8	9	10
w_i	0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02

AdaBoost: example

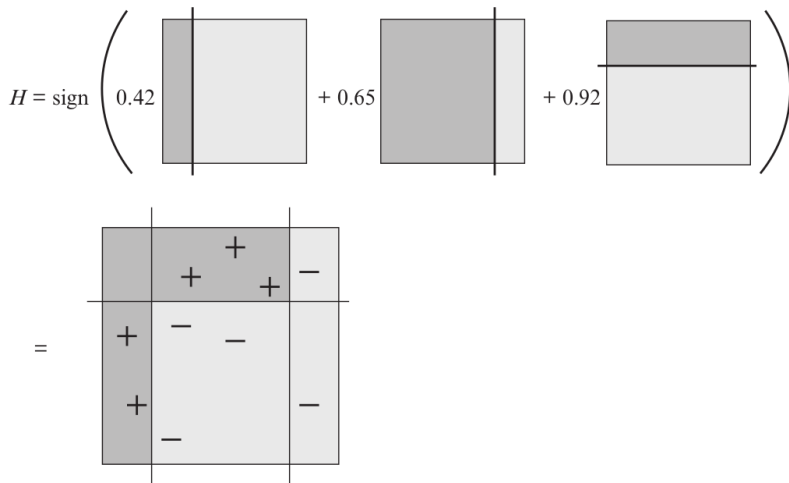
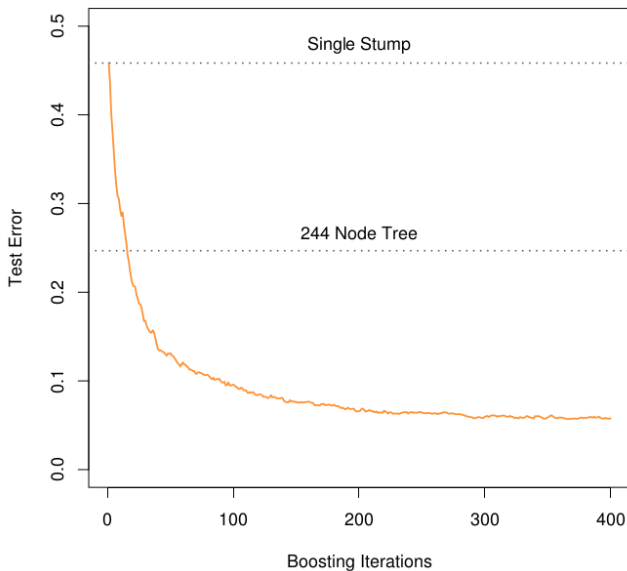


figure from Schapire & Freund (2014)

AdaBoost: example



Statistical Boosting: Boosting as a forward stagewise additive modelling

The **statistical view** of boosting is based on the concept of **forward stagewise additive modelling**:

- minimizes a **loss function** $L(y_i, f(x_i))$;
- using an **additive model**,

$$f(x) = \sum_{m=1}^M \alpha_m b(x; \gamma_m);$$

- ▶ $b(x; \gamma_m)$ is the **basis**, or **weak learner**;
- at **each step**,

$$(\alpha_m, \gamma_m) = \operatorname{argmin}_{\alpha, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \alpha b(x_i; \gamma));$$

- the estimate is updated as $f_m(x) = f_{m-1}(x) + \alpha_m b(x; \gamma_m)$
- e.g., in AdaBoost, $\alpha_m = \alpha_m$, $b(x; \gamma_m) = G(x)$;

Statistical Boosting: Boosting as a forward stagewise additive modelling

(see notes)

Statistical Boosting: Why exponential loss?

The statistical view of boosting:

- allows to **interpret** the results;
- by studying the **properties of the exponential loss**;

It is easy to show that

$$f^*(x) = \operatorname{argmin}_{f(x)} E_{Y|X=x}[e^{-Yf(x)}] = \frac{1}{2} \log \frac{\Pr(Y = 1|x)}{\Pr(Y = -1|x)},$$

i.e.

$$\Pr(Y = 1|x) = \frac{1}{1 + e^{-2f^*(x)}};$$

therefore AdaBoost estimates **1/2 the log-odds** of $\Pr(Y = 1|x)$.

Statistical Boosting: Why exponential loss?

Note:

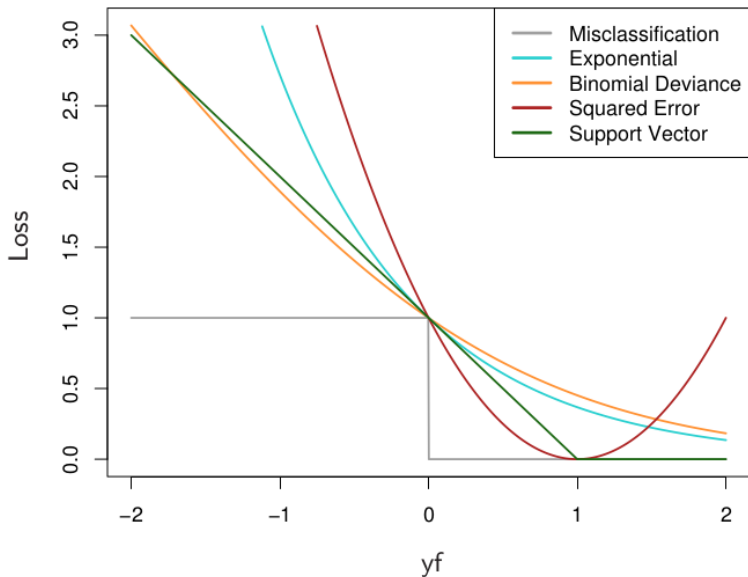
- the exponential loss is **not** the only possible loss-function;
- deviance (cross/entropy): **binomial negative log-likelihood**,

$$-\ell(\pi_x) = -y' \log(\pi_x) - (1 - y') \log(1 - \pi_x),$$

where:

- ▶ $y' = (y + 1)/2$, i.e., $y' \in \{0, 1\}$;
- ▶ $\pi_x = \Pr(Y = 1|X = x) = \frac{e^{f(x)}}{e^{-f(x)} + e^{f(x)}} = \frac{1}{1 + e^{-2f(x)}}$;
- equivalently,
$$-\ell(\pi_x) = \log(1 + e^{-2yf(x)}).$$
- same population minimizers for $E[-\ell(\pi_x)]$ and $E[e^{-Yf(x)}]$.

Statistical Boosting: Why exponential loss?



Statistical Boosting: steepest descent

We saw that AdaBoost iteratively **minimizes a loss function**.

In general, consider

- $L(f) = \sum_{i=1}^N L(y_i, f(x_i))$;
- $\hat{f} = \operatorname{argmin}_f L(f)$;
- the **minimization problem** can be solved by considering

$$f_{m_{\text{stop}}} = \sum_{m=0}^{m_{\text{stop}}} h_m$$

where:

- ▶ $f_0 = h_0$ is the **initial guess**;
- ▶ each f_m **improves** the previous f_{m-1} through h_m ;
- ▶ h_m is called "**step**".

Statistical Boosting: steepest descent

The **steepest descent** chooses

$$h_m = -\rho_m g_m$$

where

- $g_m \in \mathbb{R}^N$ is the **gradient descent** of $L(f)$ evaluated at $f = f_{m-1}$ and represents the **direction** for the minimization,

$$g_{im} = \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \Big|_{f(x_i)=f_{m-1}(x_i)}$$

- ρ_m is a scalar and tells “**how much**” to minimize

$$\rho_m = \operatorname{argmin}_{\rho} L(f_{m-1} - \rho g_m).$$

Statistical Boosting: example

Consider the linear Gaussian regression case:

- $L(y, f(x)) = \frac{1}{2} \sum_{i=1}^N (y_i - f(x_i))^2$;
- $f(x) = X^T \beta$;
- initial guess: $\beta \equiv 0$.

Therefore:

- $g = \frac{\partial \frac{1}{2} \sum_{i=1}^N (y_i - f(x_i))^2}{\partial f(x_i)} = -(y - X^T \beta)$;
- $g = -(y - X^T \beta)|_{\beta=0} = -y$;
- $\rho = \operatorname{argmin}_{\rho} \frac{1}{2} (y - \rho y)^2 \rightarrow \rho = X(X^T X)^{-1} X^T$.

Note:

- overfitting!

Statistical Boosting: weak learner

If we only needed to minimize the loss on the training data:

- Done! Use steepest descent.

But:

- the goal is to find an $\hat{f}$ that works in general (new data);
- we can only compute the gradient descent on the train data.

Possible solution:

- fit a base learner which approximates the gradient:
 - ▶ line which minimize the sum of squares (linear regression);
 - ▶ a regression (or classification) tree;
 - ▶ ...
- e.g.: $b(g_m, X) = X(X^T X)^{-1} X^T g_m$;

Statistical Boosting: shrinkage

To **regularize** the procedure, a **shrinkage factor** is introduced,

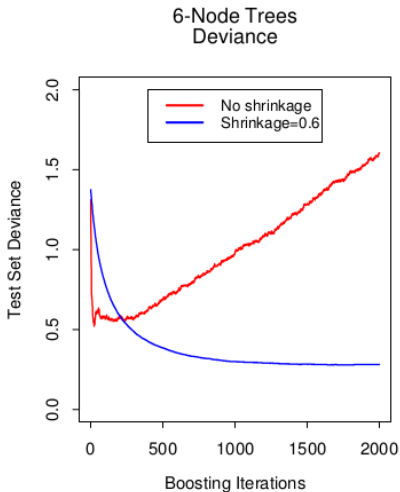
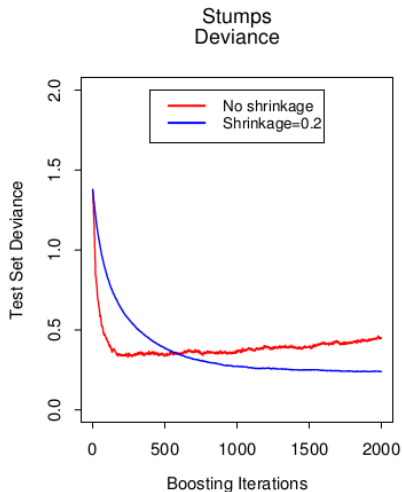
$$f_m(x) = f_{m-1}(x) + \nu h_m$$

where $0 < \nu < 1$.

The **tuning parameter** ν :

- is often called **boosting step size**;
- control the learning rate:
 - ▶ smaller value, more shrinkage in the single step;
- empirically shown that small values ($\nu \leq 0.1$) work better;
- often set equal to 0.1;
- relation with **number of steps** m_{stop} ,
 - ▶ smaller value, more steps.

Statistical Boosting: shrinkage



Statistical Boosting: Gradient boosting

Gradient boosting algorithm:

1. initialize the estimate, e.g., $f_0(x) = 0$;
2. for $m = 1, \dots, m_{\text{stop}}$,
 - 2.1 compute the **negative gradient** vector,
$$u_m = - \left. \frac{\partial L(y, f(x))}{\partial f(x)} \right|_{f(x) = \hat{f}_{m-1}(x)};$$
 - 2.2 fit the **base learner** to the negative gradient vector, $b_m(u_m, x)$;
 - 2.3 **update** the estimate, $f_m(x) = f_{m-1}(x) + \nu b_m(u_m, x)$.
3. final estimate, $\hat{f}_{m_{\text{stop}}}(x) = \sum_{m=1}^{m_{\text{stop}}} \nu b_m(u_m, x)$

Note:

- $u_m = -g_m$
- $\hat{f}_{m_{\text{stop}}}(x)$ is a **GAM**.

Statistical Boosting: example

Consider again the linear Gaussian regression case:

- $L(y, f(X)) = \frac{1}{2} \sum_{i=1}^N (y_i - f(x_i, \beta))^2$, $f(x_i, \beta) = x_i \beta$;
- $h(y, X) = X(X^T X)^{-1} X^T y$.

Therefore:

- initialize the estimate, e.g., $\hat{f}_0(X, \beta) = 0$;
- $m = 1$,
 - ▶ $u_1 = - \left. \frac{\partial L(y, f(X, \beta))}{\partial f(X, \beta)} \right|_{f(X, \beta) = \hat{f}_0(X, \beta)} = (y - 0) = y$;
 - ▶ $b_1(u_1, X) = X(X^T X)^{-1} X^T y$;
 - ▶ $\hat{f}_1(x) = 0 + \nu X(X^T X)^{-1} X^T y$.
- $m = 2$,
 - ▶ $u_2 = - \left. \frac{\partial L(y, f(X, \beta))}{\partial f(X, \beta)} \right|_{f(X, \beta) = \hat{f}_1(X, \beta)} = (y - X^T(\nu \hat{\beta}))$;
 - ▶ $b_2(u_2, X) = X(X^T X)^{-1} X^T (y - X^T(\nu \hat{\beta}))$;
 - ▶ update the estimate, $\hat{f}_2(X, \beta) = \nu X(X^T X)^{-1} X^T y + \nu X(X^T X)^{-1} X^T (y - X^T(\nu \hat{\beta}))$.

Statistical Boosting: remarks

Note that:

- we do **not** need to have a linear effects,
 - $b(y, X)$ can be, e.g., a **spline**;
- using $f(X, \beta) = X^T \beta$, it makes more sense to **work with β** :
 1. initialize the estimate, e.g., $\hat{\beta}_0 = 0$;
 2. for $m = 1, \dots, m_{\text{stop}}$,
 - 2.1 compute the negative gradient vector,
$$u_m = - \left. \frac{\partial L(y, f(X, \beta))}{\partial f(X, \beta)} \right|_{\beta = \hat{\beta}_{m-1}};$$
 - 2.2 fit the base learner to the negative gradient vector,
$$b_m(u_m, X) = (X^T X)^{-1} X^T u_m;$$
 - 2.3 update the estimate, $\hat{\beta}_m = \hat{\beta}_{m-1} + \nu b_m(u_m, x)$.
 3. final estimate, $\hat{f}_{m_{\text{stop}}}(x) = X^T \hat{\beta}_m$.

Statistical Boosting: remarks

Further remarks:

- for $m_{\text{stop}} \rightarrow \infty$, $\hat{\beta}_{m_{\text{stop}}} \rightarrow \hat{\beta}_{OLS}$;
- the shrinkage is controlled by both m_{stop} and ν ;
- usually ν is fixed, $\nu = 0.1$
- m_{stop} is computed by cross-validation:
 - ▶ it controls the model complexity;
 - ▶ we need an early stop to avoid overfitting;
 - ▶ if it is too small $\rightarrow$ too much bias;
 - ▶ if it is too large $\rightarrow$ too much variance;
- the predictors must be centred, $E[X_j] = 0$.

References I

- BÜHLMANN, P. (2006). Boosting for high-dimensional linear models. *The Annals of Statistics* **34**, 559–583.
- FRIEDMAN, J., HASTIE, T. & TIBSHIRANI, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics* **28**, 337–407.
- MAYR, A., BINDER, H., GEFELLER, O. & SCHMID, M. (2014). Extending statistical boosting. *Methods of Information in Medicine* **53**, 428–435.
- SCHAPIRE, R. E. & FREUND, Y. (2014). *Boosting: Foundations and Algorithms*. MIT Press, Cambridge.