Boosting algorithms

Vincent Divol

These notes are based on Schapire and Freund's excellent textbook on boosting methods [Schapire and Freund, 2012].

1 What is boosting?

How can a crowd of uninformed people take good decisions by working collectively? The simplest instance of such a phenomenon arises when n "uninformed" people answer a question. Assuming that they perform slightly better than random (say they answer correctly with probability 55%) and that their answers are independent, then the correct answer can be found with high probability by taking a vote.

Lemma 1.1 (Hoeffding inequality)**.** *Let $X_1, \dots, X_n$ be independent random variables in $[0,1]$, and let $A_n = \frac{1}{n}\sum_{i=1}^n X_i$. Then, for any $r > 0$,*

$$\mathbb{P}(A_n > \mathbb{E}[A_n] + r) \leq e^{-2nr^2}. \tag{1}$$

In our example, we let $X_i = 1$ if the ith answer is false, and 0 otherwise. Then, the majority is right if $A_n = \frac{1}{n}\sum_{i=1}^n X_i$ is smaller than $1/2$, whereas $\mathbb{E}[A_n] = 0.45$. Therefore,

$$\mathbb{P}(\text{the majority is wrong}) = \mathbb{P}(A_n > 1/2) = \mathbb{P}(A_n > \mathbb{E}[A_n] + 0.05) \leq e^{-5\cdot 10^{-3} n}. \tag{2}$$

For a crowd of $n = 1000$ people, the probability that the majority is wrong is already extremely small, smaller than $e^{-5} \simeq 0.6\%$.

Boosting algorithms aim at aggregating weak classifiers, that act as "rules of thumbs" that perform slightly better than random, to create a strong classifier with excellent generalization error. Consider for instance the problem of spam detection. Simple rules of thumbs may consist in automatically classifying an email as spam if it contains a suspect word such as "Viagra". Of course, not all emails containing "Viagra" are spam, but such a rule of thumb will definitely perform better than random. Other rules of thumb can be defined by creating a list of forbidden words, or by defining a list of "safe" email addresses. Individually, each rule will only have a mediocre performance, although we will see how aggregating such rules in a carefully designed way is enough to obtain a small error.

Formally, consider a training set $(X_1, Y_1), \dots, (X_n, Y_n)$ of n labelled examples, with $X_i \in \mathcal{X}$ being an example (e.g. an email) whereas $Y_i \in \{-1, +1\}$ is the associated label (e.g. spam or not spam). We assume that both X_i and Y_i are random, and that our training examples are independent and identically distributed from some (unknown) probability distribution P on $\mathcal{X} \times \{-1, +1\}$. A *classifier* is a function $h : \mathcal{X} \to \{-1, +1\}$, whereas its generalization error is defined as

$$\mathrm{err}(h) = P(h(X) \neq Y), \tag{3}$$

that is the generalization error is the probability of making mistakes on new unlabelled data.

Boosting algorithms take as input a family $\mathcal{H}$ of weak classifiers. Naively, the "best" weak classifier should be the one that minimizes the training error, defined by

$$\widehat{\mathrm{err}}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{h(X_i) \neq Y_i\}. \tag{4}$$

In boosting algorithms, the final classifier H is obtained by making many different carefully selected weak classifiers participate in a vote. That is to say H will take the form of

$$H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right), \tag{5}$$

where $h_1, \ldots, h_T$ are the selected weak classifiers and α_t is the voting weight assigned to the classifier h_t. When all the weights α_t are equal, $H(x)$ will be equal to 1 if the majority of the $h_t(x)$s are equal to 1, and equal to -1 otherwise. We allow to put higher weights ($\alpha_t \gg 1$) on some classifiers in our voting scheme if we believe that they have better accuracy.

If we ask for the best weak classifier repeatedly while using the same dataset, our algorithm will simply return the same classifier again and again. Thus, a key component of boosting algorithms consists in weighting the observations of the training set, to signify that making a mistake on observations with higher weights is more critical. To do so, consider a vector of weights $D = (D(1), \ldots, D(n))$, with $D(i) \geq 0$ and $\sum_{i=1}^{n} D(i) = 1$. The D-weighted training error is defined as

$$\widehat{\mathrm{err}}_D(h) = \sum_{i=1}^{n} D(i)\mathbf{1}\{h(X_i) \neq Y_i\}. \tag{6}$$

When all the $D(i)$s are equal to $1/n$, the weighted error is equal to the usual training error. We are now in position to describe a popular boosting algorithm, called AdaBoost.

Algorithm 1 The AdaBoost algorithm

Input: Dataset $(X_1, Y_i), \ldots, (X_n, Y_n)$, family of weak classifiers $\mathcal{H}$, number of rounds T

Initialize: $D_1 = (1/n, \ldots, 1/n)$

for $t = 1, \ldots, T$ **do**

Choose $h_t \in \mathcal{H}$ that minimizes the weighted error

$$\widehat{\mathrm{err}}_{D_t}(h) = \sum_{i=1}^{n} D_t(i)\mathbf{1}\{h(X_i) \neq Y_i\}. \tag{7}$$

Let $\varepsilon_t = \widehat{\mathrm{err}}_{D_t}(h_t)$ and $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.

Let

$$\forall i = 1, \ldots, n, \qquad D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{\alpha_t} & \text{if } h_t(X_i) \neq Y_i, \\ e^{-\alpha_t} & \text{if } h_t(X_i) = Y_i, \end{cases} \tag{8}$$

where $Z_t = \sum_{i=1}^{n} D_{t+1}(i)$ is a normalization constant.

end for

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

Several preliminary remarks are in order to understand the intuitive behavior of the AdaBoost algorithm. First, if h_t has a small weighted error ε_t, then α_t is large, so that h_t will have a huge voting power in the final vote, which is a desirable property. Second, if h_t misclassifies the ith observation, then $D_{t+1}(i) \geq D_t(i)$, so that we put more importance on having the ith

observation right on the next round. To put it another way, the reweighting scheme forces us to find weak classifiers that perform well on *hard* examples, that is examples on which previous classifiers tended to make mistakes.

2 AdaBoost's training error

In this section, we show that the training error $\widehat{\mathrm{err}}(H)$ of AdaBoost decreases exponentially fast to 0 with the number of rounds T. Note that, by itself, having a small training error says nothing about the generalization error, due to *overfitting* phenomena. We will however explain in the next section that the generalization error can be equally well controlled.

Theorem 2.1. *Let $\gamma_t = \frac{1}{2} - \varepsilon_t$ be the edge of h_t, measuring how much h_t performs better than random (for the weighting scheme D_t). It holds that*

$$\widehat{\mathrm{err}}(H) \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right). \tag{9}$$

Proof. Let $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. Note that we can write $D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t Y_i h_t(X_i)}}{Z_t}$. Therefore,

$$\begin{aligned} D_{t+1}(i) &= D_1(i) \times \frac{e^{-\alpha_1 Y_i h_1(X_i)}}{Z_1} \times \cdots \times \frac{e^{-\alpha_T Y_i h_T(X_i)}}{Z_T} \\ &= \frac{1}{n} \frac{\exp(-Y_i F(X_i))}{\prod_{t=1}^{T} Z_t}. \end{aligned}$$

Remark that we can rewrite the condition $H(x) = y$ as $yF(x) > 0$. Furthermore, the $0-1$ function $t \mapsto \mathbf{1}\{t > 0\}$ is smaller than the function $t \mapsto e^{-t}$. This implies that the following inequality holds:

$$\mathbf{1}\{H(X_i) \neq Y_i\} \leq \exp(-Y_i F(X_i)). \tag{10}$$

Therefore,

$$\begin{aligned} \widehat{\mathrm{err}}(H) &= \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{H(X_i) \neq Y_i\} \\ &\leq \frac{1}{n} \sum_{i=1}^{n} \exp(-Y_i F(X_i)) = \sum_{i=1}^{n} D_{T+1}(i) \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t, \end{aligned}$$

where we use that $\sum_{i=1}^{n} D_{T+1}(i) = 1$. Furthermore,

$$\begin{aligned} Z_t &= \sum_{i:\ Y_i = h_t(X_i)} D_t(i) e^{-\alpha_t} + \sum_{i:\ Y_i \neq h_t(X_i)} D_t(i) e^{\alpha_t} \\ &= e^{-\alpha_t}(1 - \varepsilon_t) + e^{\alpha_t} \varepsilon_t \\ &= e^{-\alpha_t}\left(\frac{1}{2} + \gamma_t\right) + e^{\alpha_t}\left(\frac{1}{2} - \gamma_t\right) \\ &= \sqrt{1 - 4\gamma_t^2}. \end{aligned}$$

The second inequality is obtained using that $1 - x \leq e^{-x}$ for $x \geq 0$. □

This control on the training error becomes useful only when the edges γ_t are large enough, that is all the h_ts perform substantively better than random. For instance, if $\gamma_t \geq 10\%$ for all t, then the inequality gives $\widehat{\mathrm{err}}(H) \leq (0.98)^T$: the training error decreases exponentially fast to 0.

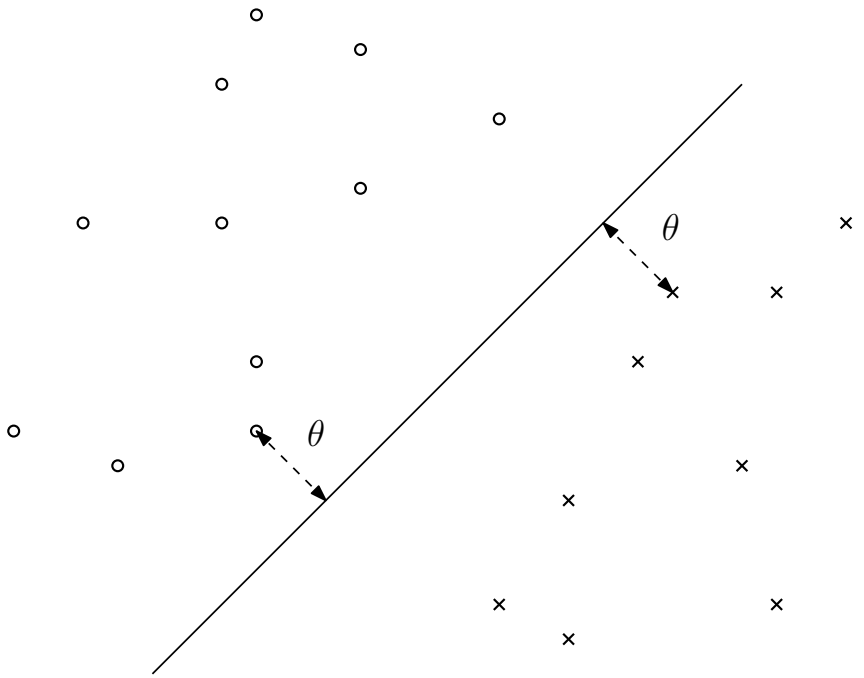

Figure 1: A dataset of points in $\mathbb{R}^2$, labelled either as crosses, or circles. The set of linear classifiers linearly separates this dataset, with margin θ given by the distance between the best separating line and the dataset.

When can we ensure such uniform control over the edges? Intuitively, this is possible when the family $\mathcal{H}$ of weak classifiers is rich enough. Let $\mathrm{Conv}(\mathcal{H}) = \{\sum_j a_j h_j : \ h_j \in \mathcal{H}, a_j \geq 0, \sum_j a_j = 1\}$ be the *convex hull* of $\mathcal{H}$. A way to ensure that $\mathcal{H}$ has a high enough complexity is to guarantee the existence of some number $\theta > 0$, called a *margin*, such that we can find $F \in \mathrm{Conv}(\mathcal{H})$ with $Y_i F(X_i) \geq \theta$ for all $i = 1, \ldots, n$. Intuitively, the margin θ measures the confidence of our classification: our rule is to classify X_i as $+1$ if $F(X_i) > 0$, but a number $F(X_i)$ very close to 0 suggests that a small perturbation of X_i can flip our classification from $+1$ to -1, whereas such a phenomenon is forbidden by the margin hypothesis. If such an assumption is satisfied, we say that $\mathcal{H}$ is able to *linearly separate the observations with margin* θ. See Figure 1 for an example.

If this assumption is satisfied, then all the edges γ_t are larger than $\theta/2$, ensuring that $\widehat{\mathrm{err}}(H) \leq e^{-\frac{\theta^2 T}{2}}$. Let us prove this fact. Let $D = D_t$ be the weights over the observations $X_1, \ldots, X_n$ at the round t. Let $F = \sum_j a_j h_j$ be the classifier associated with the observations $(X_1, Y_1), \ldots, (X_n, Y_n)$, so that

$$\sum_j a_j \mathbb{E}_D[Y_i h_j(X_i)] = \mathbb{E}_D\left[\sum_j a_j h_j(X_i)\right] \geq \theta. \tag{11}$$

As this average (over the weights a_js) is larger than θ, there exists in particular some h_j with $\mathbb{E}_D[Y_i h_j(X_i)] \geq \theta$. But, a simple computation gives

$$\mathbb{E}_D[Y_i h_j(X_i)] = 1 - 2\widehat{\mathrm{err}}_D(h_j),$$

yielding that $\widehat{\mathrm{err}}_D(h_j) \leq \frac{1}{2} - \frac{\theta}{2}$. In particular, the best weak classifier in $\mathcal{H}$ has an error smaller than this quantity, and **all the edges γ_t are at least $\theta/2$.**

3 AdaBoost's generalization error

The goal of a machine learning algorithm is not to minimize the training error, but to minimize the *generalization error*, that is the probability of misclassifying a new unlabelled observation. Recall that we assumed that our training dataset is made of i.i.d. observations from some law P, and that the unlabelled data follows the same distribution. Therefore, the expectation of the training error (with respect to the training dataset) is equal to the generalization error. More precisely, if h is some fixed classifier and $U_i = \mathbf{1}\{Y_i \neq h(X_i)\}$, then $\mathbb{E}[U_i] = P(Y \neq h(X)) = \mathrm{err}(h)$. In particular, by Hoeffding's inequality, the generalization error *of a fixed classifier* h cannot be far from the generalization error:

$$\mathrm{err}(h) \leq \widehat{\mathrm{err}}(h) + \sqrt{\frac{\log(1/\delta)}{2n}} \tag{12}$$

with probability at least $1-\delta$ (this is obtained by playing around with the inequality (1)).

However, this line of reasoning only holds when h is fixed and was picked independently from the set of observations! Otherwise, the classifier h becomes itself random (as it depends on the random observations (X_i, Y_i)), and we cannot conclude in such a simple fashion. Assume that we pick a classifier H (by looking at the observations) from a family $\mathcal{C}$ of classifiers. Then, we can write

$$\begin{aligned} \mathrm{err}(H) &= \widehat{\mathrm{err}}(H) + (\mathrm{err}(H) - \widehat{\mathrm{err}}(H)) \\ &\leq \widehat{\mathrm{err}}(H) + \sup_{F \in \mathcal{C}} (\mathrm{err}(F) - \widehat{\mathrm{err}}(F)). \end{aligned} \tag{13}$$

The second term $\sup_{F\in\mathcal{C}}(\mathrm{err}(F) - \widehat{\mathrm{err}}(F))$ can be bounded with probability at least $1-\delta$ by $2\mathcal{R}_n(\mathcal{C}) + \sqrt{\frac{2\log(1/\delta)}{n}}$, where $\mathcal{R}_n(\mathcal{C})$ is the *Rademacher complexity of* $\mathcal{C}$.

Definition 3.1. *The Rademacher complexity of a family of classifiers $\mathcal{C}$ is defined as*

$$\mathcal{R}_n(\mathcal{C}) = \mathbb{E}_{P,\varepsilon}\left[\sup_{F\in\mathcal{C}} \left| \frac{1}{n}\sum_{i=1}^n \varepsilon_i F(X_i) \right| \right], \tag{14}$$

where the ε_is are i.i.d. ± 1 signs, equal to $+1$ with probability $1/2$, and -1 with probability $1/2$.

The quantity $\frac{1}{n}\sum_{i=1}^n \varepsilon_i F(X_i)$ is equal to the correlation between the random sign vector $(\varepsilon_1, \ldots, \varepsilon_n)$ and the classification made by F, $(F(X_1), \ldots, F(X_n))$. Therefore, the Rademacher complexity measures how well it is possible, given our observations $X_1, \ldots, X_n$, to label the observations with any given combination of random signs. If a perfect match is always possible, then our model $\mathcal{C}$ can learn "garbage data", where there is no discernible pattern in the labelling. This should definitely be a red flag, indicating that the model $\mathcal{C}$ is too complex. Such a situation exactly corresponds to the Rademacher complexity attaining its maximal value of $\mathcal{R}_n(\mathcal{C}) = 1$. When $\mathcal{C}$ has a moderate complexity, we however expect $\mathcal{R}_n(\mathcal{C})$ to decay at rate $1/\sqrt{n}$. This is for instance the case when $\mathcal{C}$ is finite and made of k classifiers, where it holds that

$$\mathcal{R}_n(\mathcal{C}) \leq \sqrt{\frac{2\log k}{n}}. \tag{15}$$

All in all, we obtain that with probability at least $1-\delta$,

$$\mathrm{err}(H) \leq \widehat{\mathrm{err}}(H) + 2\mathcal{R}_n(\mathcal{C}) + \sqrt{\frac{2\log(1/\delta)}{n}}. \tag{16}$$

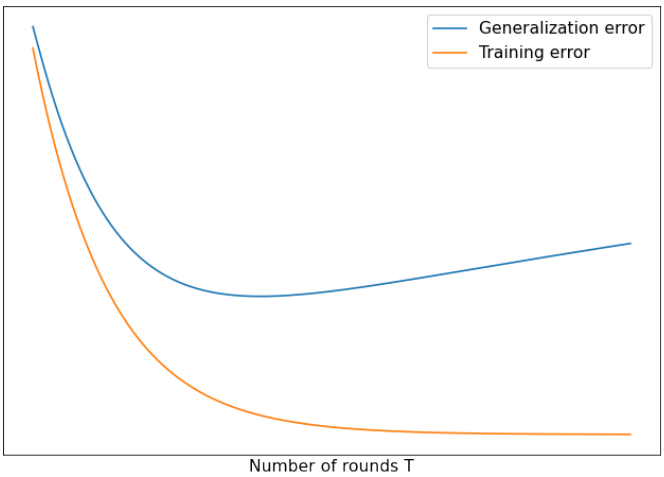

Figure 2: Theoretical behavior of AdaBoost's overfit without margin assumptions. As the number of rounds T increases, the training error decreases exponentially, but the generalization error increases as $\sqrt{T/n}$ for large T.

This inequality explains the *bias-fluctuations trade-off* in machine learning. If one tries to fit a simple model $\mathcal{C}$ to our dataset, then even the best classifier H in $\mathcal{C}$ will make many mistakes when classifying the training set, so that $\widehat{\mathrm{err}}(H)$ is large. We then say that the model $\mathcal{C}$ has a large bias and is *underfitting*. On the other hand, if the model $\mathcal{C}$ is too complex (that is very large), we expect the first term $\widehat{\mathrm{err}}(H)$ to be small. However, the Rademacher complexity of $\mathcal{C}$ will blow up, so that the generalization error $\mathrm{err}(H)$ will still be large: we say that there are large fluctuations (or variance) and that the model $\mathcal{C}$ is *overfitting*.

Let us return to the analysis of the AdaBoost algorithm. For the sake of simplicity, we will assume for the remainder of this section that $\mathcal{H}$ is finite and contains k classifier (although other methods exist to bound the Rademacher complexity when $\mathcal{H}$ is infinite). After T rounds of the algorithm, the final classifier will belong to the class $\mathcal{C}_T$ of classifiers of the form

$$x \mapsto \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right), \quad \alpha_1, \dots, \alpha_T \in \mathbb{R}, \quad h_1, \dots, h_T \in \mathcal{H}. \tag{17}$$

It can be proven that the Rademacher complexity of the class $\mathcal{C}_T$ is small. Namely, we have

$$\mathcal{R}_n(\mathcal{C}_T) \leq \sqrt{\frac{32T\log(enk/T)}{n}}. \tag{18}$$

Roughly speaking, the Rademacher complexity behaves as if $\mathcal{C}_T$ were a finite model of size of order $(nk/T)^T$ (see (15)). Discarding the logarithmic factors and the constants, and assuming that $\mathcal{H}$ is able to linearly separate the observations with margin θ, we obtain in total that, with probability at least $1-\delta$,

$$\begin{aligned} \mathrm{err}(H) &\leq \widehat{\mathrm{err}}(H) + \sqrt{\frac{32T\log(enk/T)}{n}} + + \sqrt{\frac{2\log(1/\delta)}{n}} \\ &\lesssim e^{-\frac{\theta^2 T}{2}} + \sqrt{\frac{T}{n}}. \end{aligned} \tag{19}$$

This inequality suggests the following qualitative behavior. As the number T of rounds of AdaBoost increases, the training error decreases exponentially to 0. When T stays moderately

Figure 3: The fetus' health dataset, taken from [Ayres-de Campos et al., 2000], contains different features describing a fetus, with the goal of predicting if the fetus is healthy or not. The AdaBoost algorithm was trained on this dataset, using *stumps* as base classifiers, which are classifiers with linear boundaries of the form $\{x : \ x_j \leq c\}$ for some feature j and threshold c. Training and generalization errors are displayed as a function of the number of rounds (log-scale). As expected, the training error decreases exponentially, reaching zero after a couple hundred rounds, whereas the generalization error starts increasing after roughly 50 rounds, going from 13% to 17.5%.

small, this leads to a good generalization error. However, when T gets too large, the factor $\sqrt{\frac{T}{n}}$ blows up, and AdaBoost starts overfitting. Illustrations in theory and in practice of this behavior are shown in Figure 2 and Figure 3.

4 Avoiding overfitting under a margin hypothesis

The example shown above shows that the analysis made in the previous section does capture AdaBoost's behavior on some examples. See however in Figure 4 how AdaBoost behaves on a task of classification of raisin species (taken from the UCI ML Repository [Dua and Graff, 2017]).

As expected, the training error decreases exponentially, until reaching zero after approximately 10^3 rounds. More surprisingly, the generalization error keeps decreasing as the number of rounds T gets larger! This is in contradiction with the intuition built in the previous section, based on an analysis of the Rademacher complexityof the model $\mathcal{C}_T$: even when we fit a very complex model, consisting of linear combinations of thousands of base classifiers, there is no overfitting. First, remark that for T larger than 10^3, the training error of H is equal to 0, and there are likely many different classifiers in $\mathcal{C}_T$ that reach 0 training error. However, AdaBoost does not simply pick "at random" one such minimizer, but selects a very special one.

Remember that H can be written as $H(x) = \text{sign}(F(x))$ where $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. Define

$$a_t = \frac{\alpha_t}{\sum_{t=1}^{T} \alpha_t} \tag{20}$$

and $f(x) = \sum_{t=1}^{T} a_t h_t(x)$, so that we may also write $H(x) = \text{sign}(f(x))$. As $\sum_{t=1}^{T} a_t = 1$, the

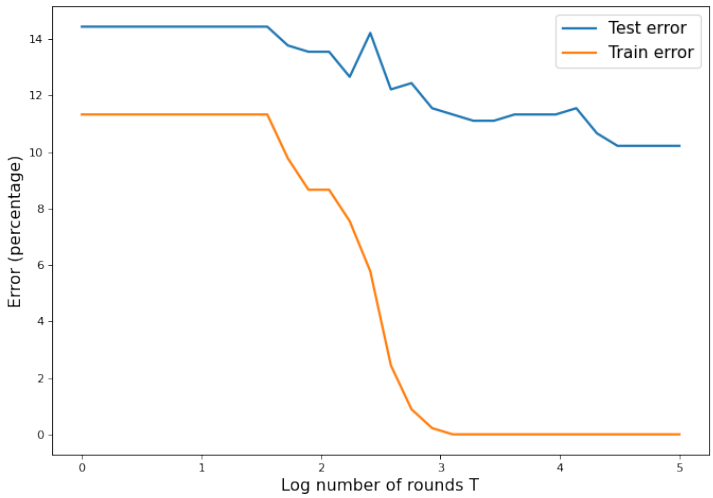

Figure 4: Seven different features describing the geometry of raisin seeds are collected, with the aim of classifying the seeds as belonging either to the "Kecimen" or "Basin" species. The AdaBoost algorithm was trained using stumps as base classifiers. Contrarily to what was predicted in Section 3, the generalization error does not increase with the number of rounds, but keeps decreasing even when the training error has reached zero.

number $f(x)$ is between -1 and $+1$, whereas the classification will be equal to $+1$ as long as $f(x) > 0$, and -1 otherwise. We think of the absolute value $|f(x)|$ as the confidence of the classifier in its prediction: if $|f(x)|$ is close to 0, then a small perturbation of the input x will be enough to switch its sign, changing the classification. In this situation, the classifier is not "sure" of how it should classify x. See in Figure 5 the distribution of the margins $|f(X_i)|$ of the observations X_i in the training set for different numbers of rounds.

As seen on the picture, as the number of rounds increases, and although the training error is always zero, the confidence of H in its classifications still increases. Let us explain this phenomenon theoretically. Fix a margin $\theta \in [0, 1]$. Note that $f(X_i)Y_i > \theta$ means exactly that (i) Y_i and $f(X_i)$ have the same sign, so that X_i is well classified, and (ii) $|f(X_i)| > \theta$, so that the confidence in the classification is at least θ. Let $\hat{N}_\theta(f)$ be the proportion of training examples with $Y_i f(X_i) \leq \theta$. For instance, in the previous example, for $T = 5$, $\hat{N}_{0.5}(f)$ is approximately equal to 0.10. Introduce the function ϕ with

$$\phi(u) = \begin{cases} 1 & \text{if } u \leq 0 \\ 1 - u/\theta & \text{if } 0 \leq u \leq \theta \\ 0 & \text{if } u \geq \theta. \end{cases} \tag{21}$$

Remark that for any classifier of the form $H(x) = \text{sign}(f(x))$,

$$\text{err}(H) \leq \mathbb{E}_P[\phi(Y f(X))] \tag{22}$$

and

$$\frac{1}{n} \sum_{i=1}^{n} \phi(Y_i f(X_i)) \leq \hat{N}_\theta(f). \tag{23}$$

For the AdaBoost algorithm, f belongs to $\text{Conv}(\mathcal{H})$, the convex hull of the classifiers in $\mathcal{H}$.

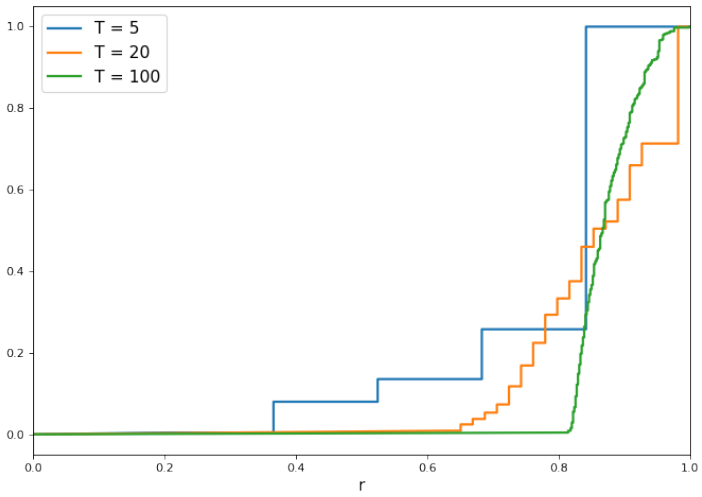

Figure 5: The margin distribution for different number of rounds. For a given r-value, the y-value corresponds to the proportion of observations X_i with margin smaller than r. For instance, for $T = 5$, there are roughly 10% of observation points that are classified with margin smaller than 0.5.

Therefore,

$$\begin{aligned}\mathrm{err}(H) \leq \mathbb{E}_P[\phi(Yf(X))] &= \frac{1}{n}\sum_{i=1}^{n}\phi(Y_i f(X_i)) + \left(\mathbb{E}_P[\phi(Yf(X))] - \frac{1}{n}\sum_{i=1}^{n}\phi(Y_i f(X_i))\right) \\ &\leq \hat{N}_\theta(f) + \sup_{g\in\mathrm{Conv}(\mathcal{H})}\left(\mathbb{E}_P[\phi(Yg(X))] - \frac{1}{n}\sum_{i=1}^{n}\phi(Y_i g(X_i))\right).\end{aligned}$$

By using similar mathematical results than in the previous section, we know that with probability $1-\delta$,

$$\sup_{g\in\mathrm{Conv}(\mathcal{H})}\left(\mathbb{E}_P[\phi(Yg(X))] - \frac{1}{n}\sum_{i=1}^{n}\phi(Y_i g(X_i))\right) \leq 2\mathcal{R}_n(\phi\circ\mathrm{Conv}(\mathcal{H})) + \sqrt{\frac{2\log(2/\delta)}{n}}, \tag{24}$$

where $\phi\circ\mathrm{Conv}(\mathcal{H})$ is the set of functions of the form $\phi\circ g$ for some $g\in\mathrm{Conv}(\mathcal{H})$. There are two known facts on the Rademacher complexity:

1. $\mathcal{R}_n(\phi\circ\mathrm{Conv}(\mathcal{H})) \leq \mathrm{Lip}(\phi)\mathcal{R}_n(\mathrm{Conv}(cH))$, where $\mathrm{Lip}(\phi)$ is the *Lipschitz constant of* ϕ, here equal to $1/\theta$;
2. $\mathcal{R}_n(\mathrm{Conv}(cH)) = \mathcal{R}_n(\mathcal{H})$.

All in all, we obtain that for a fixed $\theta > 0$, and with probability at least $1-\delta$,

$$\mathrm{err}(H) \leq \hat{N}_\theta(f) + \frac{2}{\theta}\mathcal{R}_n(\mathcal{H}) + \sqrt{\frac{2\log(2/\delta)}{n}}. \tag{25}$$

When $\mathcal{H}$ is finite, of size k, we know that the Rademacher complexity is bounded by $\sqrt{\frac{2\log k}{n}}$. Furthermore, a small modification of Theorem 2.1 shows that, if $\mathcal{H}$ linearly separates the observations with maring θ, then $\widehat{\mathrm{err}}(H) \leq 2e^{-\frac{\theta^2 T}{2}}$, so that

$$\mathrm{err}(H) \leq 2e^{-\frac{\theta^2 T}{2}} + \frac{2}{\theta}\sqrt{\frac{2\log k}{n}} + \sqrt{\frac{2\log(2/\delta)}{n}}. \tag{26}$$

Let us compare this bound with the one obtained in (19): the two are extremely similar, with the only distinction that the dependency in $\sqrt{\frac{T}{n}}$ has been replaced by a dependency in $\frac{1}{\theta}\sqrt{\frac{1}{n}}$. Therefore, **under a margin hypothesis, the generalization error of AdaBoost does not increase with the number of rounds T.**

References

[Ayres-de Campos et al., 2000] Ayres-de Campos, D., Bernardes, J., Garrido, A., Marques-de Sa, J., and Pereira-Leite, L. (2000). Sisporto 2.0: a program for automated analysis of cardiotocograms. *Journal of Maternal-Fetal Medicine*, 9(5):311–318.

[Dua and Graff, 2017] Dua, D. and Graff, C. (2017). UCI Machine Learning Repository.

[Schapire and Freund, 2012] Schapire, R. E. and Freund, Y. (2012). *Boosting: foundations and algorithms*. Adaptive computation and machine learning series. MIT Press, Cambridge, MA.