Sample Boosting Algorithm (SamBA) - An Interpretable Greedy Ensemble Classifier Based On Local Expertise For Fat Data (Supplementary Material)

Baptiste Bauvin[1] **Cécile Capponi**[2] **Florence Clerc**[3] **Pascal Germain**[1] **Sokol Koço**[2] **Jacques Corbeil**[4]

[1]Computer Science and Software Engineering Dept., Laval University, Québec, QC, Canada
[2]Laboratoire d'Informatique et Systèmes, Aix-Marseille University , Marseille, France
[3]Department of Computer Science, McGill University, Montreal, QC, Canada
[4]Molecular Medicine Dept., Laval University, Québec, QC, Canada

A RE-WRITING ADABOOST WITH LOCAL KNOWLEDGE

In this section, we show how Adaboost fits our generalized algorithm (see Algorithm 1). We then rewrite the Adaboost algorithm to explicitly fit that generalized skeleton in Algorithm 2.

In order to derive an ensemble classifier from the generalized framework presented in the main paper, we have to give explicit values to the "abstract" functions:

- The relevance function is defined as $\omega(h_t, (x_i, y_i)) = \mathbf{1}\left\{h_t(x_i) = y_i\right\}$
- The difficulty function is defined as $\pi(h_{t+1}, (x_i, y_i)) = \exp\left(-\frac{1}{2}\ln\left(\frac{1-\sum_{i=1}^{m}\boldsymbol{\pi}_t(i)\boldsymbol{\omega}_t[i]}{\sum_{i=1}^{m}\boldsymbol{\pi}_t(i)\boldsymbol{\omega}_t[i]}\right)h_t(x_i)y_i\right)$

For Adaboost, the decision function $P^{\mathcal{F}_T}(.)$ is computed as follows:

$$P^{\mathcal{F}_T}(x) = \sum_{t=1}^{T}\frac{1}{2}\ln\left(\frac{1-\sum\limits_{i=1}^{m}\boldsymbol{\pi}_t(i)\boldsymbol{\omega}_t[i]}{\sum\limits_{i=1}^{m}\boldsymbol{\pi}_t(i)\boldsymbol{\omega}_t[i]}\right)h_t(x)$$

This means that if we define $\hat{\omega}_t^{\text{Ada}}$ to be a constant coefficient over the whole space

$$\hat{\omega}_t^{\text{Ada}}(x) = \hat{\omega}_t^{\text{Ada}} = \frac{1}{2}\ln\left(\frac{1-\sum\limits_{i=1}^{m}\boldsymbol{\pi}_t(i)\boldsymbol{\omega}_t[i]}{\sum\limits_{i=1}^{m}\boldsymbol{\pi}_t(i)\boldsymbol{\omega}_t[i]}\right)$$

, we get

$$P^{\mathcal{F}_T}(x) = \sum_{t=1}^{T}\hat{\omega}_t^{\text{Ada}}h_t(x)$$

As explained in the main paper and as explicitly shown here, this decision function uses for each voter a scalar that is computed on the whole dataset. The weight of each voter does not encapsulate local knowledge.

B PSEUDO CODE FOR SAMBA

In this section, we present the pseudo-code of SamBA in Algorithm 3, based on the functions provided in the main paper.

Algorithm 1: A general skeleton for boosting with local expertise.

```
1 Iterations : $T$ ; Train data : $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^m$ ; Voter space : $\mathcal{H}$ ; $\mathcal{F}_0 = \emptyset$ ; Prior distribution : $\mathcal{P}$
2 $\boldsymbol{\pi}_1 \leftarrow \mathcal{P}$                                  # Prior distribution
3 for $t = 1..T$ do
4     $h_t \leftarrow \arg\max_{h \in \mathcal{H}} \left[\mu^{(\mathcal{L})}(h, \boldsymbol{\pi}_t\right]$      # Learn the best voter on them
5     $\boldsymbol{\omega}_t[i] \leftarrow \omega\left(h_t, (x_i, y_i)\right)$          # Compute its relevance on each sample
6     $\mathcal{F}_t \leftarrow \mathcal{F}_{t-1} \bigcup \{(h_t, \boldsymbol{\omega}_t)\}$       # Store the voter and relevance
7     $\boldsymbol{\pi}_{t+1}(i) \leftarrow \frac{\boldsymbol{\pi}_t(i)\pi(h_t,(x_i,y_i))}{\sum_{i=1}^m \pi(h_t,(x_i,y_i))}$      # Find the difficult samples
8 end
Result: $P^{\mathcal{F}_T}(\cdot) = \sum_{t=1}^T h_t(x)\hat{\omega}_{\mathcal{L}}^{\mathcal{F}_T}(h_t, \cdot)$
```

Algorithm 2: Adaboost re-written as an instance of the skeleton in Algorithm 1

```
1 Number of iterations : $T$ ; Train data : $\mathcal{L} = \{(x_i, y_i) | i = 1..m\}$ ; Voter space : $\mathcal{H}$
2 $\boldsymbol{\pi}_1(i) \leftarrow \frac{1}{m}$                               # Uniform distribution
3 for $t = 1..T$ do
4     $h_t \leftarrow \arg\max_{h \in \mathcal{H}} \left[\mu^{(\mathcal{L})}(h, \boldsymbol{\pi}_t(i))\right]$      # Select the best voter on them
5     $\boldsymbol{\omega}_t[i] \leftarrow \mathbf{1}\{h_t(x_i) = y_i\}$            # Compute its relevance on each sample
6     $\mathcal{F}_t \leftarrow \mathcal{F}_{t-1} \bigcup \{(h_t, \boldsymbol{\omega}_{t,:})\}$      # Store the voter and relevance
7     $\boldsymbol{\pi}_{t+1}(i) \leftarrow \boldsymbol{\pi}_t(i) \frac{\exp\left(-\frac{1}{2}\ln\left(\frac{1-\sum_{i=1}^m \boldsymbol{\pi}_t(i)\boldsymbol{\omega}_t[i]}{\sum_{i=1}^m \boldsymbol{\pi}_t(i)\boldsymbol{\omega}_t[i]}\right) h_t(x_i) y_i\right)}{Z_t}$      # Find the difficult samples
8 end
Result: $P^{\mathcal{F}_T}(.)$
```

C ANALYSIS OF THE INFLUENCE OF THE HYPERPARAMETERS ON THE WEIGHT ESTIMATION FUNCTION

In this section, we aim at providing intuitive understanding on the influence of a and b on the weight estimation function of SamBA. In Equation 1 we recall the way SamBA approximates the weight for classifier h_t when evaluating on a test sample (x, y).

$$\hat{\omega}_{h_t}^{\mathcal{L}}(x) := \begin{cases} \boldsymbol{\omega}_t[i] \text{ if } x = x_i \text{ and } a = 0, \\ \sum_{i=1}^m \frac{\boldsymbol{\omega}_t[i] m}{a^b + d(x_i, x)^b} \text{ else.} \end{cases} \tag{1}$$

As one may understand when looking closely at this expression, the value $\hat{\omega}_{h_t}^{\mathcal{L}}(x)$ is highly influenced by the values of a and b. In the following, we illustrate this on a toy example where $\mathcal{X}$ is the real line, and we explore the impact of those hyperparameters on the function $\hat{\omega}_{h_t}^{\mathcal{L}}$. The considered toy example is a one dimensional random dataset of 200 samples, for which we provide the values of $\boldsymbol{\omega}_t[i]$. In this experiment, we consider that all the samples where $x \leq 0.5$ have a relevance of 0.44 and the others of 0.006. We will call this the *pure* dataset. To understand the usefulness of a, we also generate a random dataset where three samples of each group are provided with the wrong relevance and we will call it the *noisy* dataset. Since this is a toy example, the concept of noise can be debated, however, this is useful to mimic what happens in real life for outlier or mislabelled samples.

Figure 1 highlights (on our example) the behavior of the weight estimation function of SamBA, compared to the standard method, of discarding local knowledge. In the first row, we analyze the differences on a *pure* dataset, where the relevance on the samples is well distributed, with all samples $x \leq 0.5$ voting for a high relevance for our imaginary classifier, and the others, voting for a low relevance.

Algorithm 3: SamBA, the empirically valid training algorithm based on the skeleton of Algorithm 1.

```
1 Iterations : $T$ ; Train data : $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^m$ ; Voter space : $\mathcal{H}$ (decision stumps). ; Hyper-parameters: $a, b$
2 $\boldsymbol{\pi}_1(i) \leftarrow \frac{1}{m}$
3 for $t = 1..T$ do
4     $h_t \leftarrow \underset{h \in \mathcal{H}}{\arg\max} \left[\mu^{(\mathcal{L})}(h, \boldsymbol{\pi}_t(i))\right]$
5     $\boldsymbol{\omega}_t[i] \leftarrow \exp\left(h_t(x_i) y_i\right)$
6     $\mathcal{F}_t \leftarrow \mathcal{F}_{t-1} \bigcup \{(h_t, \boldsymbol{\omega}_t)\}$
7     $\boldsymbol{\pi}_{t+1}(i) \leftarrow \boldsymbol{\pi}_t(i) * \frac{\exp(-\boldsymbol{\omega}_t[i] h_t(x_i) y_i)}{Z_t}$
8 end
9 $\boldsymbol{\omega} \leftarrow \frac{\boldsymbol{\omega}}{\sum_{i,t} \boldsymbol{\omega}_t[i]}$
Result: $\sum_{t=1}^{T} h_t(.) \left( \sum_{i=1}^{m} \frac{\boldsymbol{\omega}_t[i] m}{a^b + d(x_i, .)^b} \right)$
```

C.1 CASE WHERE $a = 0$

In this subsection, we only analyze the behavior of the blue curves, for $a = 0$. Even if this case is purely synthetic, it is interesting to note the impact that b and a have on the weight estimation. Indeed, note how the value of the weight estimation function decreases between all the high values of $\boldsymbol{\omega}_t[i]$ in the case where $b = 1.5$. This is because $\omega_t^{a,b}(x)$ takes into account even the vote of samples that are far: all these parabolas on the $a = 0$ curve (in blue) are due to the fact that all the samples $x_i \geq 0.5$ vote for a low relevance.

In the case where $b = 4$, the distance is much more important, therefore the curve is much smoother, as the opinion of distant samples becomes negligible.

In the case of a more noisy dataset, such as the lower graphs, we see that the fact that $a = 0$ is highly problematic, as it forces the curve $\omega_t^{a,b}(x)$ to reach each value of $\boldsymbol{\omega}_t[i]$. This case is typical of an over-fitting algorithm.

C.2 CASE WHERE $a = 0.02$

We now decipher the information provided by the red curves, for which $a = 0.02$. Setting a to anything else than 0 on *pure* datasets does not improve the classification, it might even lower the confidence of the classifier, as it smoothes the curves when it is not needed.

However, in the case where there is some labelling noise, it is mandatory to avoid overfitting. Mathematically, a can be interpreted as a distance lower threshold for which SamBA considers that training samples too close have too much influence on $\omega_t^{a,b}(x)$. It is clear in the graph for $b = 1.5$ that even with $a = 0.02$, SamBA takes into account the fact that some samples disagree on the classifier relevance, but it does not greatly impact the estimated relevance.

By analyzing these graphs, we gave some insight on the impact of the hyper-parameters of SamBA. The tool used to generate those figures is available with the code of SamBA. We recommend exploring with it to better understand the inner workings of the algorithm and its hyperparameters.

D THEORETICAL ANALYSIS : PROOFS.

In this section, we provide the proofs for the convergence and generalization guarantees provided in the main paper.

D.1 CONVERGENCE GUARANTEE OF ALGORITHM 3

We start by recalling the convergence theorem, provided in the main paper.

Theorem 1 (Error exponential decrease)**.** *We assume $a = 0$. We denote ϵ_t the error of voter h_t on the training set weighted by $\boldsymbol{\pi}_t$. We write $\gamma_t = \frac{1}{2} - \epsilon_t$. We also consider $\boldsymbol{\pi}_1$ an arbitrary distribution over the training set. Then, the weighted training*

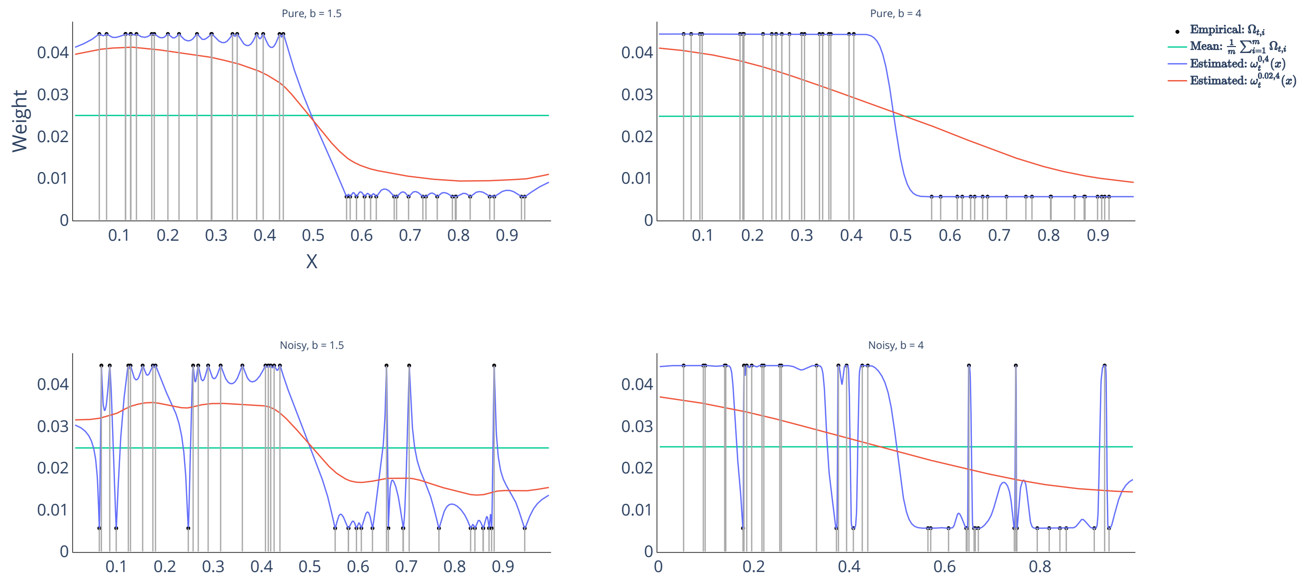


Figure 1: Weight estimation functions on a one dimensional dataset. We plotted the relevance weight (ordinates) of an hypothetical classifier h_t as a function of the samples values on the single feature (abscissa). The black dots represent the empirical weights learned by SamBA. The straight green line represents the estimated weight if no local knowledge is used. The two red and blue curves represent SamBA's weight estimation with two values of a.

error of the combined classifier outputted by Algorithm 3, with respect to $\boldsymbol{\pi}_1$ is bounded by:

$$\Pr_{i\sim\boldsymbol{\pi}_1}\left[\left(P_{0,b}^{\mathcal{F}_T}(x_i)\right) \neq y_i\right] \leq \prod_{t=1}^{T} 1-\gamma_t \leq exp(-\sum_{t=1}^{T}\gamma_t)$$

Proof. To prove such a result, we heavily rely on the methodology used in the error decrease proof of Adaboost, presented in Freund and Schapire [1997].

First, recall that in SamBA, for $a = 0$, we have $P_{0,b}^{\mathcal{F}_T}(x_i) = \sum\limits_{t=1}^{T} \boldsymbol{\omega}_t[i]h_t(x_i)$.

We write $Z_t = \sum\limits_{i=1}^{m} \exp\left(-y_i h_t(x_i)\boldsymbol{\omega}_t[i]\right)\boldsymbol{\pi}_t(i)$, similarly to the boosting framework. Thus, if we rewrite the value of $\boldsymbol{\pi}_{T+1}(i)$ at iteration $T+1$, for the sample x_i, we obtain that

$$\begin{aligned}
\boldsymbol{\pi}_{T+1}(i) &= \boldsymbol{\pi}_T(i)\frac{\exp\left(-\boldsymbol{\omega}_t[i]h_T(x_i)y_i\right)}{Z_T} \\
&= \boldsymbol{\pi}_1(i)\frac{\exp\left(-\sum_{t=1}^{T}\boldsymbol{\omega}_t[i]h_t(x_i)y_i\right)}{\prod\limits_{t=1}^{T} Z_t} \\
&= \boldsymbol{\pi}_1(i)\frac{\exp(-y_i P_{0,b}^{\mathcal{F}_T}(x_i))}{\prod\limits_{t=1}^{T} Z_t}
\end{aligned}$$

which can be rewritten as

$$\boldsymbol{\pi}_1(i)\exp(-y_i P_{0,b}^{\mathcal{F}_T}(x_i)) = \boldsymbol{\pi}_{T+1}(i)\prod_{t=1}^{T} Z_t. \tag{2}$$

Moreover, $\mathbf{1}\left\{\mathrm{sg}\left(P_{0,b}^{\mathcal{F}_T}(x_i)\right) \neq y_i\right\} \leq \exp\left(-y_i P_{0,b}^{\mathcal{F}_T}(x_i)\right)$, therefore the (weighted) training error is given by:

$$\begin{aligned}
\underset{i\sim\boldsymbol{\pi}_1}{\mathbf{Pr}}\left[P_{0,b}^{\mathcal{F}_T}(x_i)y_i < 0\right] &= \sum_{i=1}^{m} \boldsymbol{\pi}_1(i)\mathbf{1}\left\{\mathrm{sg}\left(P_{0,b}^{\mathcal{F}_T}(x_i)\right) \neq y_i\right\} \\
&\leq \sum_{i=1}^{m} \boldsymbol{\pi}_1(i)\exp\left(-y_i P_{0,b}^{\mathcal{F}_T}(x_i)\right) \\
&= \sum_{i=1}^{m} \boldsymbol{\pi}_{T+1}(i)\left(\prod_{t=1}^{T} Z_t\right) \quad \text{using Equation 2} \\
&= \left(\prod_{t=1}^{T} Z_t\right)\sum_{i=1}^{m} \boldsymbol{\pi}_{T+1}(i) \quad \text{by factoring the term in parenthesis} \\
&= \prod_{t=1}^{T} Z_t \quad \text{since } \boldsymbol{\pi}_{T+1} \text{ sums to 1.}
\end{aligned} \tag{3}$$

Analyzing Z_t, we have

$$\begin{aligned}
Z_t &= \sum_{i=1}^{m} \exp\left(-y_i h_t(x_i)\boldsymbol{\omega}_t[i]\right)\boldsymbol{\pi}_t(i) \\
&= \sum_{i:y_i\neq h_t(x_i)} \exp\left(\boldsymbol{\omega}_t[i]\right)\boldsymbol{\pi}_t(i) + \sum_{i:y_i=h_t(x_i)} \exp\left(-\boldsymbol{\omega}_t[i]\right)\boldsymbol{\pi}_t(i)
\end{aligned} \tag{4}$$

Thanks to our definition of $\boldsymbol{\omega}$, we have $\boldsymbol{\omega}_t[i] = \exp\left(h_t(x_i)y_i\right) \in \left[\frac{1}{e}, e\right]$[1]. So, if we note ϵ_t the error of classifier h_t on the training set, weighted by $\boldsymbol{\pi}_{t,\cdot}$, we obtain the following.

$$\begin{aligned}
Z_t &= \sum_{i:y_i\neq h_t(x_i)} \exp\left(\boldsymbol{\omega}_t[i]\right)\boldsymbol{\pi}_t(i) + \sum_{i:y_i=h_t(x_i)} \exp\left(-\boldsymbol{\omega}_t[i]\right)\boldsymbol{\pi}_t(i) \\
&\leq \sum_{i:y_i\neq h_t(x_i)} \exp\left(\frac{1}{e}\right)\boldsymbol{\pi}_t(i) + \sum_{i:y_i=h_t(x_i)} \exp\left(-e\right)\boldsymbol{\pi}_t(i) \\
&= \exp\left(\frac{1}{e}\right)\epsilon_t + \exp\left(-e\right)\left(1-\epsilon_t\right) \\
&= \exp\left(\frac{1}{e}\right)\left(\frac{1}{2}-\gamma_t\right) + \exp\left(-e\right)\left(\frac{1}{2}+\gamma_t\right) \\
&= \left[\exp\left(\frac{1}{e}\right) + \exp\left(-e\right)\right]\frac{1}{2} + \left[\exp\left(\frac{1}{e}\right) + \exp\left(-e\right)\right]\gamma_t
\end{aligned} \tag{5}$$

We then use that

- $\exp\left(\frac{1}{e}\right) + \exp\left(-e\right) \simeq 1.51 < 2$
- $\exp\left(\frac{1}{e}\right) - \exp\left(-e\right) \simeq -1.37 < -1$

and we obtain that $Z_t \leq 1-\gamma_t \leq \exp(-\gamma_t)$.

Therefore, we obtain $\underset{i\sim\boldsymbol{\pi}_1}{\mathbf{Pr}}\left[\mathrm{sg}\left(P_{0,b}^{\mathcal{F}_T}(x_i)\right) \neq y_i\right] \leq \prod_{t=1}^{T} 1-\gamma_t \leq \exp(-\sum_{t=1}^{T}\gamma_t)$ □

[1] Note that in most cases, the voters h_t has value in $\{-1, 1\}$. In that case, the following computation also holds, but the inequality sign is instead an equality.

D.2 GENERALIZATION GUARANTEES

The decision function of SamBA has been defined as follows:

$$P_{a,b}^{\mathcal{F}_T}(x) = \sum_{t=1}^{T} h_t(x) \left(\sum_{i=1}^{m} \frac{\boldsymbol{\omega}_t[i] m}{a^b + d(x_i, x)^b} \right) \tag{6}$$

This decision function relies on the training set and a distance function to compute the weights of each classifier on a unknown test sample. We therefore decided to use the sample compress framework in order to provide generalization guarantees on such a decision function.

We introduce a new index s that varies from 1 to $T \times m$. We use this index s to fuse both sums in Equation 6 into a single sum by considering $\lceil \frac{s}{T} \rceil = t$ and

$$i = \begin{cases} s \mod m \text{ if } s \mod m \neq 0, \\ m \text{ else.} \end{cases}$$

We can then rewrite Equation 6 as follows.

$$P_{a,b}^{\mathcal{F}_T}(x) = \sum_{s=1}^{T\times m} h_s(x) \frac{\boldsymbol{\omega}_s m}{a^b + d(x_s, x)^b} \tag{7}$$

Therefore, if we rearrange the terms to fit the framework of sample compress (SC) classifiers, we obtain the following.

$$P_{a,b}^{\mathcal{F}_T}(x) = \sum_{s=1}^{T\times m} \boldsymbol{\omega}_s \frac{h_s(x) m}{a^b + d(x_s, x)^b} \tag{8}$$

While the simple rearrangement of the terms $\boldsymbol{\omega}_s$ and $h_s(x)$ may seem innocuous, it reveals a further change in perspective. Indeed, now $\boldsymbol{\omega}_s$ is the distribution over the SC-classifiers, and $\frac{h_s(\cdot)m}{a^b + d(x_s, \cdot)^b}$ are the SC-classifiers. These classifiers only rely on a single training sample for the similarity measure. In addition, they rely on at most two samples to set the threshold for the stumps h. Therefore, the compression size of these classifiers is at most 3.

In the sample compress point of view, we consider that the classifiers are drawn from the set

$$\mathcal{H}_{\mathcal{L},\lambda}^{\mathcal{R}} = \{\mathcal{R}(S_{\boldsymbol{i}}, \boldsymbol{\sigma}) : \boldsymbol{i} \in \mathcal{I}_\lambda, \boldsymbol{\sigma} \in \Sigma_\lambda\},$$

with $\mathcal{R}$ a reconstruction function, outputting sample-compressed classifiers from a compression sequence $S_{\boldsymbol{i}}$ and a message $\boldsymbol{\sigma}$ both of size λ. Therefore, the Gibbs and Bayes risks are defined as follows.

Definition 1 (Gibbs Risk)**.** *For any probability distribution Q on a set of SC-voters, the Gibbs risk $R_{\mathcal{D}}(G_Q)$ of the Gibbs classifier G_Q on distribution $\mathcal{D}$ is defined as follows:*

$$R_{\mathcal{D}}(G_{Q,\mathcal{L}}) = \frac{1}{2}\left(1 - \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{D}} \left[\mathop{\mathbf{E}}_{(\boldsymbol{i},\boldsymbol{\sigma})\sim Q} (y \mathcal{R}(\mathcal{L}_{\boldsymbol{i}}, \boldsymbol{\sigma})(x)) \right]\right).$$

Its empirical version, on the dataset $\mathcal{L}$ is noted:

$$R_{\mathcal{L}}(G_{Q,\mathcal{L}}) = \frac{1}{2}\left(1 - \frac{1}{m}\sum_{i=1}^{m} \left[\mathop{\mathbf{E}}_{(\boldsymbol{i},\boldsymbol{\sigma})\sim Q} (y_i \mathcal{R}(\mathcal{L}_{\boldsymbol{i}}, \boldsymbol{\sigma})(x_i)) \right]\right).$$

Definition 2 (Theoretical Bayes Risk)**.** *For any probability distribution Q on a set of SC-voters, the Bayes risk $R_{\mathcal{D}}(B_Q)$ of the majority vote classifier B_Q, relatively to $\mathcal{D}$ is defined as follows.*

$$R_{\mathcal{D}}(B_{Q,\mathcal{L}}) = \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{D}} \left[I \left(\mathop{\mathbf{E}}_{(\boldsymbol{i},\boldsymbol{\sigma})\sim Q} [y \,\mathrm{sg}\,(\mathcal{R}(\mathcal{L}_{\boldsymbol{i}}, \boldsymbol{\sigma})(x)) < 0] \right) \right],$$

with $I(p) = 1$ if predicate p is true, and 0 else.

Based on this point of view, we can apply the theorem provided in Germain et al. [2015] to bound the generalization error of SamBA.

Theorem 2 (Sample compress theorem from Germain et al. [2015])**.** *Let $\mathcal{R}$ be a reconstruction function that outputs SC-classifiers of size at most λ (where $\lambda < m$). For any distribution $\mathcal{D}$ on $\mathcal{X} \times \{-1, 1\}$, for any posterior distribution P on $\mathcal{I}_\lambda \times \Sigma_\lambda$, and any $\delta \in (0, 1]$, we have:*

$$\underset{\mathcal{L}\sim\mathcal{D}^m}{\mathbf{P}}\left(\text{For all posteriors } Q,\ \ R_{\mathcal{D}}(G_{Q,\mathcal{L}}) \le R_{\mathcal{L}}(G_{Q,\mathcal{L}}) + \sqrt{\frac{1}{2(m-\lambda)}\left[KL(Q||P) + 4\lambda + \ln\left(\frac{\xi(m-\lambda)}{\delta}\right)\right]}\right) \ge 1-\delta$$

with

$$\sqrt{m} \le \xi(m) := \sum_{k=0}^{m} \binom{m}{k}\left(\frac{k}{m}\right)^k\left(1-\frac{k}{m}\right)^{m-k} \le 2\sqrt{m} \tag{9}$$

Value for λ As our sc-classifiers are based on decision stumps, they need at least 2 samples to find a threshold, and potentially one more to compute $d(x_i, x)$. Therefore, their compression size is $l \le 3$. Thus, we set the Q-average to be $\lambda \le 3$.

We then derive our bound directly from Theorem 2, by simply applying the usual bound on the Gibbs risk: $R_{\mathcal{D}}(B_{Q,\mathcal{L}}) \le 2R_{\mathcal{D}}(G_{Q,\mathcal{L}})$.

Theorem 3 (SamBA's sample compress bound)**.** *For any distribution $\mathcal{D}$, any set of sc-classifiers of form $\frac{h_s(\cdot)m}{a^b+d(x_s,\cdot)^b}$, any prior $\mathcal{P}$, and any $\delta \in (0, 1]$, we have, for Q the distribution found by the sc-version of SamBA,*

$$\underset{\mathcal{L}\sim\mathcal{D}^m}{\mathbf{P}}\left(R_{\mathcal{D}}(B_{Q,\mathcal{L}}) \le 2R_{\mathcal{D}}(G_{Q,\mathcal{L}}) \le 2\left(R_{\mathcal{L}}(G_{Q,\mathcal{L}}) + \sqrt{\frac{1}{2(m-3)}\left[KL(Q||P) + 12 + \ln\left(\frac{2\sqrt{m-3}}{\delta}\right)\right]}\right)\right) \ge 1-\delta.$$

D.3 A VERSION WITH SMALLER KL-DIVERGENCE

In the previous bound, we have no control on the value of the KL-divergence. Indeed, for each iteration of SamBA, the distribution Q is modified for m sc-classifiers. This might be an issue for the KL-divergence, as it measures the difference between $\mathcal{P}$ and Q. Let us recall the definition of KL-divergence.

Definition 3 (KL-Divergence)**.** *The KL-divergence of two finite distributions $\mathcal{P} = \{p_s\}_{s=1}^{T\times m}$ and $Q = \{q_s\}_{s=1}^{T\times m}$ is defined as follows.*

$$KL(Q||\mathcal{P}) = \sum_{s=1}^{T\times m} q_s \ln\left(\frac{q_s}{p_s}\right) \tag{10}$$

From that definition, it follows that modifying m values of q_s at each iteration might be an issue as the KL-divergence may increase significantly in the end. To overcome this issue, we provide a new bound relying on a slight variation of SamBA. That variation of SamBA considers K random samples during its learning process instead of all the m available. Therefore, our sc-classifiers become the sum of the K random samples taken into account by this new version of SamBA.

$$\sum_{k=1}^{K} \frac{h_t(\cdot)\boldsymbol{\pi}_t[k]K}{a^b + d(x_k, \cdot)^b} \tag{11}$$

Their compression sequence is now of size $K + 3$ which leads to the following modified bound.

Theorem 4 (Modified sample compress bound)**.** *For any distribution $\mathcal{D}$, any set of sc-classifiers of the form $\sum_{k=1}^{K} \frac{h_t(\cdot)\boldsymbol{\pi}_t[k]K}{a^b+d(x_k,\cdot)^b}$, any prior $\mathcal{P}$, and any $\delta \in (0, 1]$, we have, for Q the distribution found by the sc-version of SamBA,*

$$\underset{\mathcal{L}\sim\mathcal{D}^m}{\mathbf{P}}\left(R_{\mathcal{D}}(B_{Q,\mathcal{L}}) \le 2R_{\mathcal{L}}(G_{Q,\mathcal{L}}) \le 2\left(R_{\mathcal{D}}(G_{Q,\mathcal{L}}) + \sqrt{\frac{1}{2(m-K-3)}\left[KL(Q||P) + 4(K+3) + \ln\left(\frac{2\sqrt{m-K-3}}{\delta}\right)\right]}\right)\right) \ge 1-\delta$$

The bound in this theorem is a slight modification on Theorem 3, but it allows to control the KL-divergence, as it is now defined on distributions that weight the sc-voters $\sum_{k=1}^{K} \frac{h_t(\cdot)\boldsymbol{\pi}_t[k]K}{a^b+d(x_k,\cdot)^b}$, which are m times less numerous than the previous bound of Theorem 3.

E EXPERIMENTAL PROTOCOLS

In this section, we provide all the experimental details needed to reproduce the experiments.

All the baselines implementations have been imported from scikit-learn, except for XG-Boost that has been imported from the python library xgboost.

In the following section, we use the arguments name from the libraries to name the hyper-parameters, to avoid any confusion.

E.1 TIME CONSUMPTION EXPERIMENT

The time consumption experiment has been realized on synthetic datasets with every approach. As we work with ensemble methods, we had to set the number of base classifiers (or iteration) for each method. To do so, we used two strategies, one limiting all ensemble methods to 10 base classifiers, to compute their durations on equivalent number of base classifiers. And one setting the number of base classifiers outputted by the real-life experiments of Section 5.2.2, to take into account the fact that the methods have different levels of sparsity.

E.1.1 Equal number of base classifiers

In this experiment,presented in Supplementary Material E.1, we use 10 iterations or base classifiers for each ensemble method. Each approach has been parameterized as follows:

- Random Forest:
  - n_estimators: 10
  - All the other parameters: default
- KNN:
  - All the parameters: default
- SVM-RBF:
  - All the parameters: default
- Decision Tree:
  - max_depth: 10
- Adaboost:
  - n_estimators: 10
  - base_estimator: decision tree of depth 1
  - all the other parameters: default
- SamBA:
  - n_estimators: 10
  - base_estimator: decision tree of depth 1
  - all the other parameters: default.

E.1.2 Different number of base classifiers

In this experiment, presented in Section 5.1, we use a variable number of iterations or base classifiers for each ensemble method. Each approach has been parameterized as follows:

- Random Forest:

  - n_estimators: 127
  - All the other parameters: default
- KNN:
  - All the parameters: default
- SVM-RBF:
  - All the parameters: default
- Decision Tree:
  - int(log2(17))
- Adaboost:
  - n_estimators: 146
  - base_estimator: decision tree of depth 1
  - all the other parameters: default
- SamBA:
  - n_estimators: 7
  - base_estimator: decision tree of depth 1
  - all the other parameters: default.

E.2 EXPERIMENTS ON GENERATED DATASETS

This experiment was used to showcase the different models learned by the studied methods, highlighting the fact that SamBA is an hybrid method between ensemble methods and similarity-based ones.

E.2.1 Dataset generation

To generate the datasets, we used the make_moons function of scikit-learn and a custom function to generate the spirals, available on GitHub. We used random seed for reproducibility and used 1000 samples, uniformly distributed in two classes. The datasets were then splitted in a train an test set with 80

E.2.2 Algorithm parametrization

On those datasets, we mainly used preset hyper-parameters, as the goal of this study is simply to show the advantages and drawbacks of each approach on the synthetic dataset, and not to compare their classification relevance. Therefore, we parameterized the algorithms as follows:

- Random Forest:
  - All parameters: default
- KNN:
  - All the parameters: default
- SVM-RBF:
  - probability: True (to ba able to generate the contour plots)
  - C: 0.1
  - gamma: 1.1
  - All the other parameters: default
- Decision Tree:
  - All the parameters: default
- Adaboost:
  - All the parameters: default

- SamBA:
  - n_estimators: 4
  - b: 20
  - a: 1e-15
  - All the other parameters: default.

E.3 PERFORMANCE STUDY ON REAL-LIFE DATASETS

For this performance study, we respected the most robust experimental standards in Machine Learning, to ensure that our results were trustworthy.

E.3.1 General protocol

For each of the presented dataset, we :

- Splitted the dataset in 80% train and 20% test samples 10 times
- For each train/test split:
  - For each train/test split:
    - For each algorithm:
      - Optimized the hyper-parameters on the training set with random search (50 iterations), and 5-folds cross-validation,
      - Learned on the full training set
      - Predict on both the training and testing sets,
      - Computed balanced accuracy on the test set.

This protocol ensures that the results are not based on a lucky train-test split and that the hyper-parameters were fairly optimized for each algorithm. As we use random search for our hyper-parameters, we provided distributions for each parameter to draw from.

E.3.2 Hyper-parameter distributions

- Random Forest:
  - n_estimators: [1,300],
  - max_depth: [1,10],
  - criterion: gini, entropy
  - All the other parameters: default
- KNN:
  - n_neighbors: [1,10],
  - weights: [uniform, distance]
  - algorithm: "auto", "ball_tree", "kd_tree", "brute"
  - p: 1,2
  - All the other parameters: default
- SVM-RBF:
  - C: $[10^{-3}, 10^{3}]$
  - All the other parameters: default
- Decision Tree:
  - max_depth: [1,300]
  - criterion: gini, entropy
  - splitter: best, random

  - class_weight: balanced, None
  - All the other parameters: default
- Adaboost:
  - n_estimators: [1,100]
  - base_estimator: Decision Tree of depth [1,3]
  - class_weight: balanced, None
  - All the other parameters: default
- SamBA:
  - n_estimators: [1,70]
  - b: [0.1,6]
  - a: [1, 1e-10]
  - class_weight: balanced, None
  - All the other parameters: default.
- Gradient Boosting:
  - n_estimators: [1,300],
  - base_estimator: Decision Tree of depth [1,10]
  - loss: log_loss, deviance, exponential
  - learning_rate: [0,1]
  - All the other parameters: default.
- XG-Boost:
  - n_estimators: [1,300],
  - base_estimator: Decision Tree of depth [1,10]
  - objective: binary:logistic, binary:hinge
  - learning_rate: [0,1]
  - All the other parameters: default.

All the files used to build the graphs are available on GitHub , alongside a sklearn-compatible version of SamBA that is installable.

F ADDITIONAL EXPERIMENTAL INFORMATION

F.1 COMPUTATIONAL TIME COMPARISON

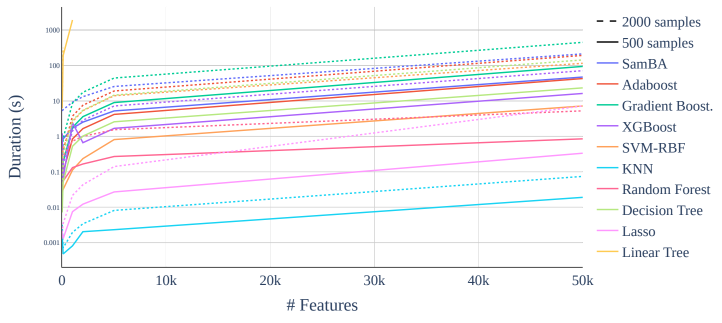
In this experiment, we repeat the one presented in Section 5.1, with a slight modification. Indeed, we set the number of base estimators for each ensemble method to 10. We therefore assess their efficiency in the hypothetical case that they are equivalently sparse.

In Figure 2, we present the results of this experiment. We see that they do not differ for the similarity-based methods. However, it is clear in this setup that SamBA has been designed for fat datasets. Indeed, it is much longer to predict than standard ensemble methods when the number of features is small. This result is to be nuanced by the fact that we plot with log scales,and therefore, the apparent difference is less than 0.74 seconds for a 2000 samples by 100 features dataset.

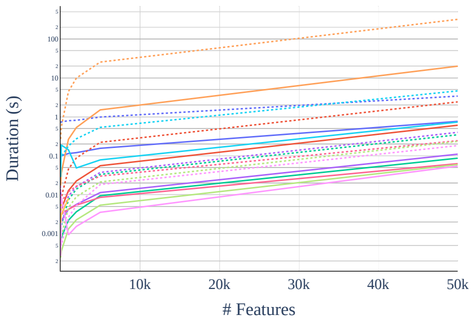
F.2 METAGENOME DATASET DESCRIPTION

The meatgenome dataset[2] presented in the main paper has been first introduced in Deraspe et al. [2020]. It consists of 640 patients, divided in 221 males, 270 females and 149 unknown gender. It labels the patients as *obese* or *not-obese*, based on their BMI. As in normal population, these is an imbalance in the dataset, with approximately 88% of the dataset being obese

[2]Note that this section does not provide an in-depth description of the biological study that lead to the dataset, but provides insights for the reader to get a better understanding of the dataset.



(a) Log train duration for each algorithm



(b) Log test duration for each algorithm

Figure 2: Learning and predicting log duration comparison on two datasets : one with 500 samples, the other with 2000. Each sample being described by an increasing number of features, ranging from 10 to 50k. The ensemble methods are limited to 10 base estimators, and the KNN to 5 neighbors.

Dataset	SamBA	Adaboost	XGBoost	Grad. Boost.	SVM-RBF	KNN	Rand. Forest	Dec. Tree	Lasso
cog (24)	.83 ± 0.06	.77 ± 0.05	.78 ± 0.06	.76 ± 0.04	.75 ± 0.05	.77 ± 0.07	.83 ± 0.04	.72 ± 0.11	.58 ± 0.05
ec (2736)	.84 ± 0.04	.70 ± 0.05	.70 ± 0.06	.65 ± 0.06	.74 ± 0.06	.72 ± 0.05	.84 ± 0.04	.70 ± 0.1	.65 ± 0.06
go (11946)	.85 ± 0.04	.73 ± 0.07	.76 ± 0.1	.71 ± 0.08	.62 ± 0.09	.75 ± 0.06	.86 ± 0.03	.73 ± 0.09	.67 ± 0.06
kegg.module (682)	.85 ± 0.05	.70 ± 0.04	.68 ± 0.06	.69 ± 0.03	.71 ± 0.03	.70 ± 0.06	.83 ± 0.04	.72 ± 0.06	.62 ± 0.06
kegg.pathway (414)	.82 ± 0.03	.67 ± 0.05	.69 ± 0.08	.67 ± 0.07	.73 ± 0.06	.73 ± 0.07	.84 ± 0.03	.69 ± 0.06	.61 ± 0.08
taxa.family (101)	.82 ± 0.04	.68 ± 0.05	.66 ± 0.06	.68 ± 0.08	.65 ± 0.08	.65 ± 0.04	.82 ± 0.05	.65 ± 0.06	.61 ± 0.08
taxa.phylum (37)	.84 ± 0.04	.70 ± 0.07	.67 ± 0.1	.66 ± 0.07	.57 ± 0.1	.63 ± 0.04	.84 ± 0.03	.74 ± 0.07	.55 ± 0.06
taxa.genus (72)	.80 ± 0.06	.63 ± 0.08	.70 ± 0.06	.70 ± 0.05	.68 ± 0.05	.68 ± 0.05	.80 ± 0.06	.73 ± 0.06	.64 ± 0.06

Table 1: Test Balanced accuracies with standard deviations for the algorithms presented in Section 5.2.2

and the remaining 12% labelled as not obese. The samples of this dataset have been selected among multiple countries, originating from four different studies, with no drug use nor additional medical condition.

From these patients, gut metagenome was extracted thanks to Whole Genome Sequencing (WGS). Then, the multiple data types were derived as follows.

- *go*, *ec*, *and cog* are extracted form the abundances of protein annotation features, respectively, their Enzyme Commission (EC) numbers, their Gene Ontology (GO) characterization, their Clusters of Orthologous Group (COG) association.
- *kegg.module* and *kegg.pathway* were extracted from the KEGG [Kanehisa and Goto, 2000] database,
- *taxa.family*, *taxa.phylum* and *taxa.genus* are abundances of the standard phylogenetic groups in the gut metagenomes.

This diversity in the data acquisition leads to a large range of dimensions, allowing to assess the relevance of our approaches on datasets with a wide range of dimensionnalities. As this dataset in itself is a contribution too complex to describe here, for more information on its construction, we highly encourage the reader to read Deraspe et al. [2020].

F.3 FULL SIZE CONTOUR PLOTS

THe full size contour plots are available in Figure 3

F.4 STATISTICAL RESULTS

In Table 1, we present the balanced accuracies fo Section 5.2.2, with their standard deviations.

Dataset	SamBA	Adaboost	XGBoost	Grad. Boost.	SVM-RBF	KNN	Rand. Forest	Dec. Tree	Lasso
cog (24)	10.7 ± 2.0	9.4 ± 7.27	9.3 ± 8.49	17.4 ± 8.0	all	all	21.3 ± 4.45	17.4 ± 6.64	12.75 ± 2.33
ec (2736)	22.2 ± 4.79	58.6 ± 49.62	145.1 ± 75.24	963.8 ± 651.62	all	all	137.3 ± 94.01	19.0 ± 13.44	265.6 ± 124.65
go (11946)	21.5 ± 10.9	168.1 ± 290.98	62.7 ± 59.54	1394.4 ± 1707.67	all	all	191.3 ± 127.98	11.3 ± 8.12	574.1 ± 644.79
kegg.module (682)	20.1 ± 3.3	44.9 ± 25.83	94.4 ± 88.9	333.0 ± 200.31	all	all	113.4 ± 67.39	16.2 ± 13.54	185.1 ± 140.51
kegg.pathway (414)	22.9 ± 3.24	87.6 ± 63.71	82.1 ± 49.52	208.0 ± 123.65	all	all	186.3 ± 41.01	28.3 ± 17.27	73.2 ± 61.59
taxa.family (101)	11.9 ± 1.7	48.3 ± 25.21	48.1 ± 20.21	79.4 ± 17.56	all	all	85.9 ± 12.0	27.5 ± 10.47	57.4 ± 12.82
taxa.phylum (37)	7.7 ± 3.72	22.4 ± 13.55	27.9 ± 6.07	35.7 ± 2.15	all	all	32.4 ± 9.72	21.3 ± 9.43	18.9 ± 11.79
taxa.genus (72)	14.8 ± 2.93	49.5 ± 25.03	42.4 ± 18.19	63.1 ± 10.87	all	all	69.1 ± 6.52	19.1 ± 15.14	47.6 ± 12.15

Table 2: Mean and standard deviation of the support size for each approach over 10 train/test splits.

F.5 FEATURE EFFICIENCY RESULTS

In Figure 5, we present the full study of the feature efficiency of SamBA, compared to the other ensemble methods studied in this paper.

In Table 2, we present the support sizes for each approach alongside its standard deviation.

G INTEPRETABILITY CONTEXT

Interpretability and explainability are notions that are currently at the center of a large number of debates. Therefore, any discussion about those concepts is very interesting. In this work, we mainly rely on the work of Rudin et al. [2021], and Molnar [2022] and hence we consider that interpretability can be seen as a multi-dimensional space, containing approaches with varying

- **Sparsity**: The number of features on which the model relies is a mandatory criterion for the interpretation of its decision.
- **Decision simplicity**: The complexity of the decision function. Even if a decision function relies on, for example, four decision stumps, combining those with logical functions, or weighted majority votes spans a wide spectrum of different function complexities.
- **Learning transparency**: The learning process is similarly important in the interpretation of the decision function. Algorithms such as decision tree are easy to understand in essence, but understanding the Gini score requires a sound mathematical background.

Those characteristics are examples of features on which interpretability relies, and justify why we consider Random Forest (RF) and Boosting partially interpretable. Indeed, in our case, RF and Boosting have the great advantage of natively proposing quantifications of the importance of each feature that have been included in their algorithm. If one considers explainability as the range of post-hoc methods used to understand classifiers, Random Forests and Boosting do not require such methods.

Random forests are problematic in the sense that hey output very dense decision functions. However, they have the great advantage of relying on a uniform majority vote, which is much simpler than the linear combination on which boosting relies.

In contrast, Boosting approaches are usually sparser than Random Forests. Therefore, in this paper, we consider that Boosting and Random Forests are both partially interpretable, as they provide better-than-post-hoc methods to understand their decision, but they still output either dense or mathematically complex decision functions.

References

Maxime Deraspe, Charles Burdet, Juan Manuel Dominguez, François Laviolette, Paul H Roy, and Jacques Corbeil. Cross-study analyses of gut microbiomes from healthy and obese individuals. 2020.

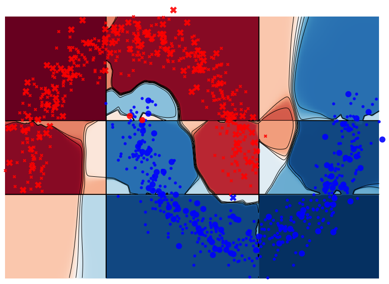
Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Jour. of Comp. and Sys. Sci.*, 55(1):119–139, 1997. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.1997.1504. URL `http://www.sciencedirect.com/science/article/pii/S002200009791504X`.

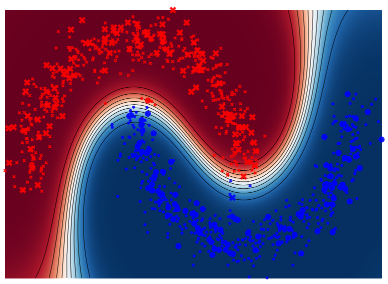
Pascal Germain, Alexandre Lacasse, Francois Laviolette, Mario March, and Jean-Francis Roy. Risk bounds for the majority vote: From a pac-bayesian analysis to a learning algorithm. *Journal of Machine Learning Research*, 16(26):787–860, 2015. URL `http://jmlr.org/papers/v16/germain15a.html`.

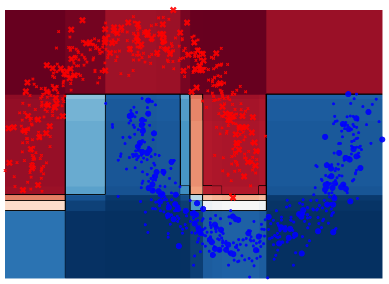
M Kanehisa and S Goto. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 28(1):27–30, January 2000.

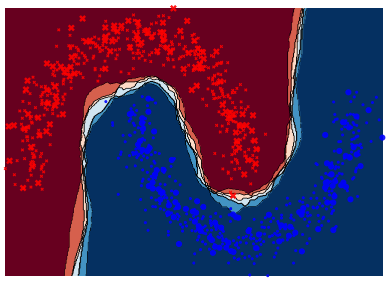
Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022. URL `https://christophm.github.io/interpretable-ml-book`.

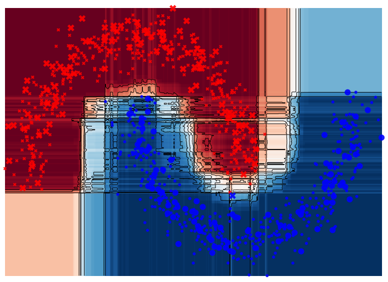
Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *CoRR*, abs/2103.11251, 2021. URL `https://arxiv.org/abs/2103.11251`.

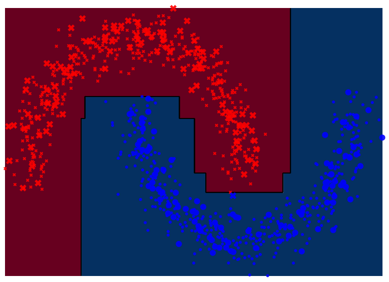

(a) SamBA- Moons

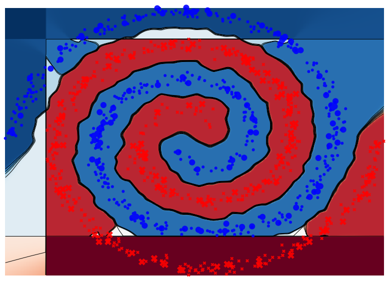

(b) SVM-RBF - Moons

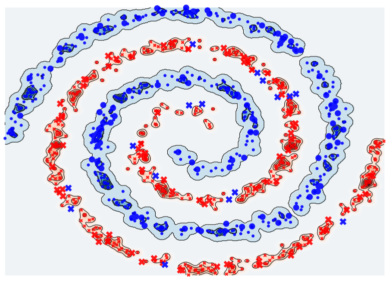

(c) Adaboost - Moons

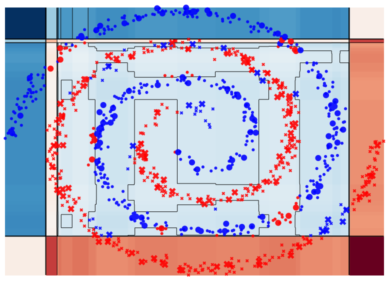

(d) KNN - Moons

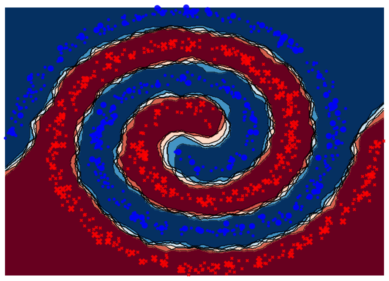

(e) RF - Moons

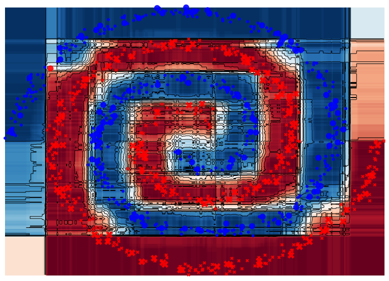

(f) DT - Moons


(g) SamBA- Spirals


(h) SVM-RBF - Spirals


(i) Adaboost - Spirals


(j) KNN - Spirals


(k) RF - Spirals

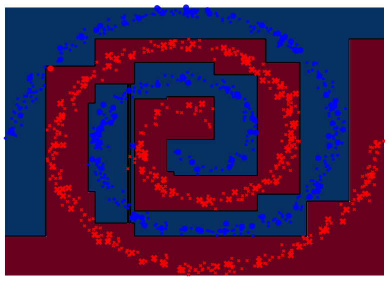
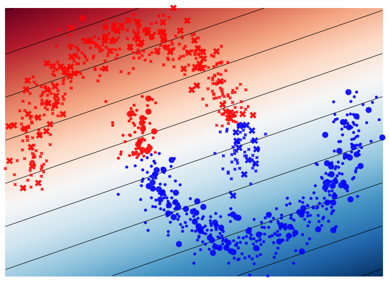
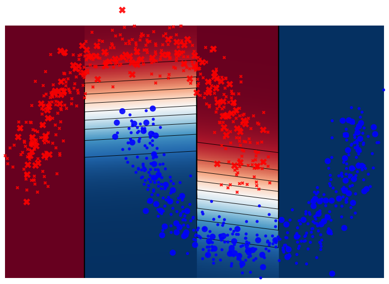

(l) DT - Spirals

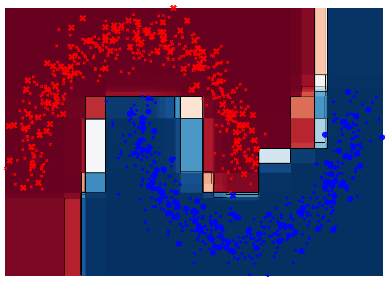
Figure 3: Decision functions contour plots for the six considered algorithms, on the two *pure* generated datasets. The small dots are train samples, the big ones test samples. The color represents the predicted class ans its intensity, the certainty of the decision function on the 2D space.



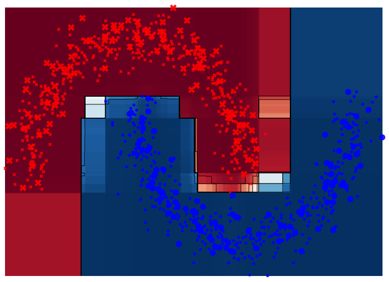
(a) Lasso - Moons



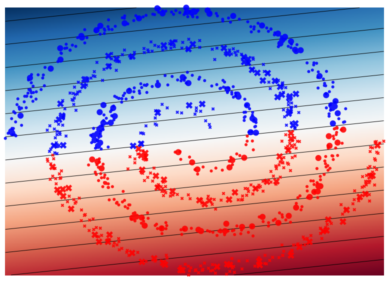
(b) Linear Tree - Moons



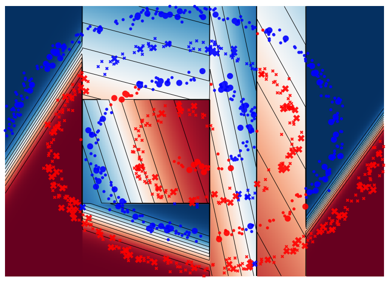
(c) Gradient Boosting - Moons



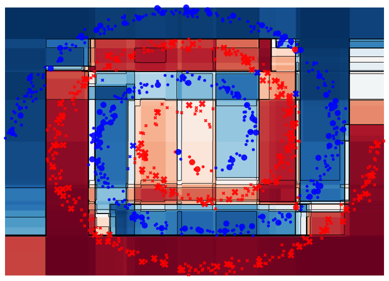
(d) XGBoost - Moons



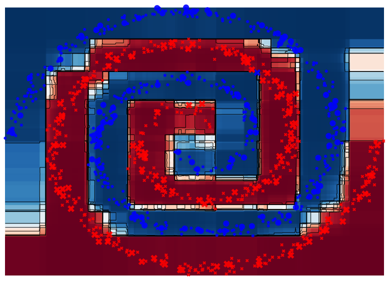
(e) Lasso - Spirals



(f) Linear Tree - Spirals

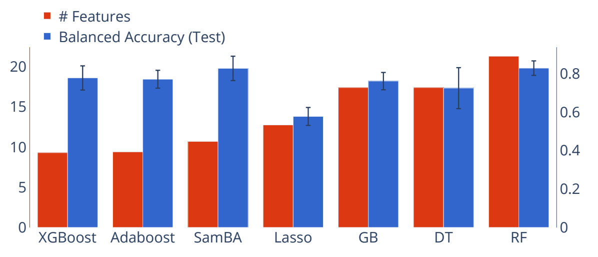


(g) Gradient Boosting - Spirals

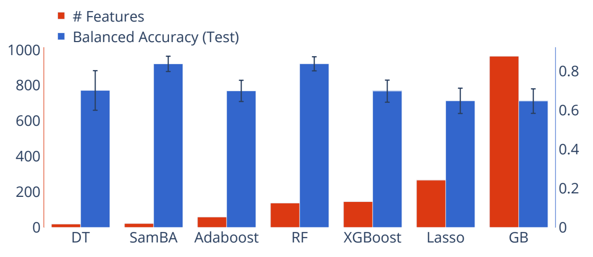


(h) XGBoost - Spirals

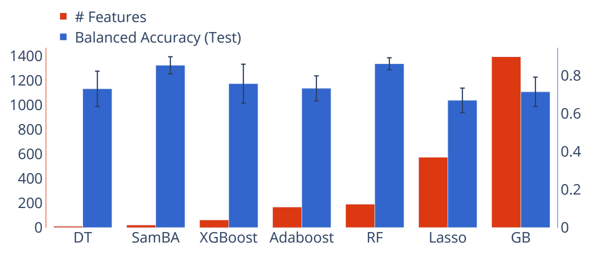
Figure 4: Decision functions contour plots for the two additional algorithms, on the two *pure* generated datasets. The small dots are train samples, the big ones test samples. The color represents the predicted class ans its intensity, the certainty of the decision function on the 2D space.



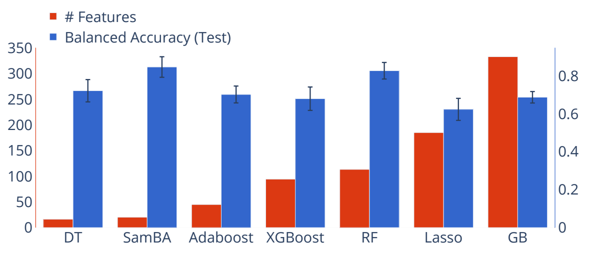
(a) cog



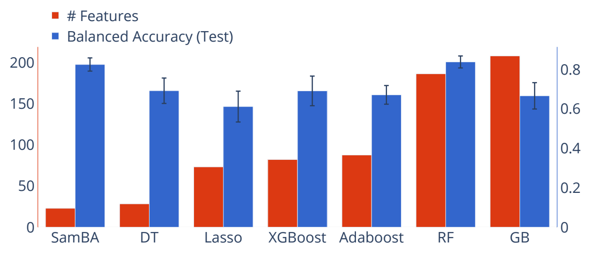
(b) ec



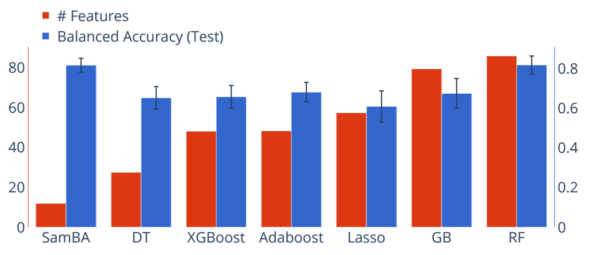
(c) go



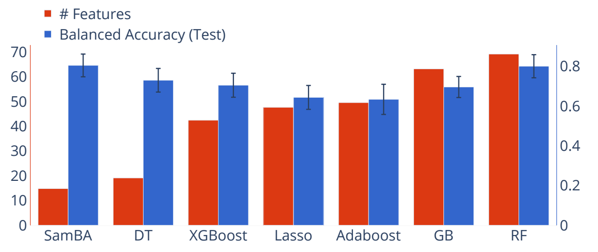
(d) kegg.module



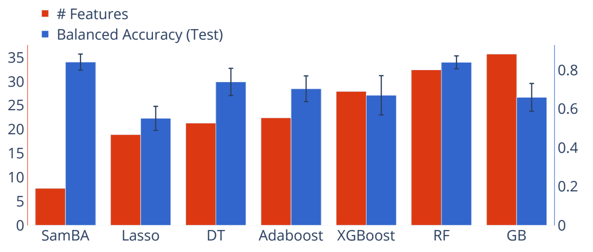
(e) kegg.pathway



(f) taxa.family



(g) taxa.genus



(h) taxa.phylum

Figure 5: Feature efficiency results for all the studied ensemble methods, on all the available data types of the *metagenomes* dataset.