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

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

Abstract

Ensemble methods are a very diverse family of algorithms with a wide range of applications. One of the most commonly used is boosting, with the prominent Adaboost. Adaboost relies on greedily learning base classifiers that rectify the error from previous iterations. Then, it combines them through a weighted majority vote, based on their quality on the entire learning set. In this paper, we propose a supervised binary classification framework that propagates the local knowledge acquired during the boosting iterations to the prediction function. Based on this general framework, we introduce SamBA, an interpretable greedy ensemble method designed for *fat* datasets, with a large number of dimensions and a small number of samples. SamBA learns local classifiers and combines them, using a similarity function, to optimize its efficiency in data extraction. We provide a theoretical analysis of SamBA, yielding convergence and generalization guarantees. In addition, we highlight SamBA's empirical behavior in an extensive experimental analysis on both real biological and generated datasets, comparing it to state-of-the-art ensemble methods and similarity-based approaches.

1 INTRODUCTION

In machine learning, ensemble methods combine base estimators into a more robust model relying on several combination methods such as logical or linear combinations, stacking [Wolpert, 1992] or cascading [Gama and Brazdil, 2000] estimators. Notably, the best performing *deep* models are obtained by combining the outputs of several neural networks [He et al., 2016]. In this work, we consider as an ensemble method, any approach aggregating multiple base estimators [Dietterich, 2000]. Those methods lead to numerous learning algorithms, from the decision tree [Breiman et al., 1984], that is an ensemble method aggregating stumps with logical combinations, to more complex setups such as multi-view learning [Koço and Capponi, 2011], not to mention the celebrated Random Forest [Breiman, 2001] and Adaboost [Schapire and Freund, 2012] majority vote learners.

This work focuses on the application of supervised classification on *fat* datasets, that have the particularity to present a large number of dimensions for a small number of samples [Romero et al., 2016], in opposition with *big* data. Numerous fat datasets are derived from biological tasks, in which algorithm interpretability—the ability for a non-expert to understand the decision function of a model [Rudin et al., 2021]—is central for the results to be endorsed by the users.

This type of dataset raises multiple challenges. Indeed, the curse of dimensionality associated with the small number of samples implies that standard methods, such as deep neural networks [Nielsen, 2018], are frequently unstable and prone to overfitting. To overcome such challenges, a commonly used state-of-the-art approach is to first apply a dimension reduction method, such as principal component analysis [Pearson, 1901] or t-distributed Stochastic Neighbor Embedding [van der Maaten and Hinton, 2008] to map the fat data into a lower-dimensional space. Then, a similarity-based method such as Support Vector Machines [Cortes and Vapnik, 1995] with a Radial Basis Function kernel (SVM-RBF) or k-Nearest Neighbors (KNN) [Fix and Hodges, 1989] is used on the new feature space. However, a drawback of these approaches is the lack of interpretability of their decision function.

To overcome this issue, ensemble methods such as Decision Trees [Breiman et al., 1984], Boosting [Freund and Schapire, 1997, Bauvin et al., 2020] or Random Forests [Breiman, 2001] are currently the gold standard. Indeed, they innately allow reducing the dimension of the datasets, while providing at least partially interpretable[1] models [Drouin et al.,

[1]We discuss the concept of interpretability in Supplementary

2016]. This advantage is central in biomedical applications, such as biomarker discovery [Kothari et al., 2020], in which interpretable models are used as a means to extract new causes of the studied problem [Osseni et al., 2021].

However, those ensemble methods present the drawback of discarding a large majority of the similarity-based information and some are frequently unstable in very large dimension [Li and Belford, 2002]. Indeed, they combine their base classifiers with either logical combinations or majority votes on the entire dataset. For example, boosting relies on the hypothesis that the relevance of the opinion of a base classifier is uniform on the whole decision space, and does not consider local relevance for its voters. This behavior leads to diverse base classifiers sets, but implies a loss in model sparsity. In addition algorithms such as decision trees with linear models in the leaves [Quinlan et al., 1992] or locally weighted linear regression [Atkeson et al., 1997] do consider local knowledge while maintaining sparsity. However, they do not scale on high-dimensional data.

In this work, we propose a general framework for supervised binary classification that enables greedily learning and combining voters by taking into account the local properties of the input space. Elaborating on this framework, we introduce SamBA, a greedy learning algorithm derived from Adaboost that outputs a classifier leveraging local knowledge to express sparse decision functions. SamBA's behavior is analyzed, providing insights on its inner mechanisms, and proving both convergence and generalization guarantees. We present extensive experiments that highlight several assets of the algorithm, including its resource efficiency. We also compare it to state-of-the-art implementations of several ensemble methods and similarity-based classifiers. We study their accuracy and sparsity on synthetic and real life datasets. Finally, the interpretability of SamBA is discussed along with its weighting scheme.

2 GENERALIZING ADABOOST WITH LOCAL EXPERTISE

This paper proposes a generalization of Adaboost's *architecture*, in which the local expertise of the base classifiers is stored during the learning process, and transferred to the prediction function through a weight estimation function. While going through the basic framework and notations, the current section reinterprets established ensemble methods as *local experts* aggregations.

2.1 LOCAL EXPERTISE IN ENSEMBLES

Let us first illustrate that well-known ensemble methods rely on local expertise. For example, the standard Decision Tree [Breiman et al., 1984] builds stumps on increasingly smaller subsets of the original sample set. Indeed, once the root has divided the dataset in two subspaces based on the first decision stump, all the following stumps only focus on improving the precision on their respective subspaces. Furthermore state-of-the-art ensemble methods that are learning base classifiers on subsets of the dataset, such as Random Forest [Breiman, 2001], may be interpreted as local experts' combinations. Indeed, even if they are built on random localities, Random Forest still learns local experts, and combines them with a uniform majority vote.

Algorithm 1: A reminder of Adaboost learning process.

```
1 **Iterations** : $T$ ; **Data** : $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{m}$ ; **Voters** : $\mathcal{H}$.
2 $D_1(i) \leftarrow \frac{1}{m}$.
3 **for** $t = 1..T$ **do**
4     $h_t \leftarrow \underset{h \in \mathcal{H}}{\arg\min} \left[ \underset{i \sim D_t}{\mathbf{P}} \left[ h(x_i) \neq y_i \right] \right]$,
5     $\epsilon_t \leftarrow \underset{i \sim D_t}{\mathbf{P}} \left[ h_t(x_i) \neq y_i \right]$,
6     $\alpha_t \leftarrow \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$,
7     $D_{t+1}(i) \leftarrow D_t(i) \times \frac{\exp(-\alpha_t h_t(x_i) y_i)}{Z_t}$.
8 **end**
9 $Z_t$ a normalization factor such that $\sum_{i=1}^{m} D_{t+1}(i) = 1$.
      **Result:** $\sum_{t=1}^{T} \alpha_t h_t(.)$.
```

Similarly, as shown in Algorithm 1, Adaboost maintains at each iteration a distribution D_t. This distribution encapsulates, for each sample of the learning set, the difficulty of classifying it. Then the algorithm learns a weak classifier that specializes in the difficult samples based on the distribution D_t. Therefore, Adaboost learns specialized weak classifiers at each iteration. However, the relevance of those specialized classifiers is stored as a simple scalar value, α_t. In doing so, Adaboost loses precious information about the local expertise of its weak classifiers. Indeed, the diagram presented in Figure 1a highlights the fact that Adaboost compresses the relevance of its base classifiers in a unique scalar value at each iteration.

2.2 GENERALIZED ADABOOST SCHEME

Let us consider a supervised binary classification task where $\mathcal{X}$ the input space is, of dimension l, and $\mathcal{Y} = \{-1, 1\}$ the target space. We denote by $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{m}$ the empirical dataset drawn according to a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The learning task aims to predict tests samples $(x, y) \sim \mathcal{D}$ accurately. As this work focuses on ensemble methods, we consider that all the base classifiers are chosen in a voter space $\mathcal{H}$, a subset of the space of functions $\mathcal{X} \to [-1, 1]$.

We study iterative learning algorithms that maintains a

Material G

Algorithm 2: 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: $P^{\mathcal{F}_T}(\cdot) = \sum_{t=1}^{T} h_t(\cdot)\hat{\omega}_{\mathcal{L}}^{\mathcal{F}_T}(h_t, \cdot)$
```

weight distribution $\boldsymbol{\pi}$ over the samples belonging to $\mathcal{L}$. In this context, we define the **empirical margin** of $h \in \mathcal{H}$ as $\mu^{(\mathcal{L})}(h, \boldsymbol{\pi}) = \sum_{i=1}^m \boldsymbol{\pi}(i)\left[y_i h(x_i)\right]$. It represents the weighted correctness of the classifier given $\boldsymbol{\pi}$. Let us define two abstract functions generalizing Adaboost ideas of current sample distribution and classifier confidence.

- The **difficulty function** π quantifies the difficulty of a sample (x_i, y_i) for a classifier h as $\pi(h, (x_i, y_i)) \in\ ]0, +\infty[$. As an example, in Adaboost, the difficulty of a sample (x_i, y_i) for the voter selected at iteration t is computed as $\exp\left(-\alpha_t h_t(x_i) y_i\right)$. Note that this function depends on the opposite of the margin $-h_t(x_i)y_i$ and on a proxy of the error of the classifier α_t.
- The **relevance function** ω of a classifier h on a sample (x_i, y_i) contrasts with the difficulty function, and is denoted as $\omega(h, (x_i, y_i)) \in\ ]0, +\infty[$. The relevance and difficulty variations are opposite: the more relevant a classifier h is w.r.t. a sample (x_i, y_i), the less difficulty h has to process that sample. In Adaboost, it is computed for h_t as α_t for the whole dataset.

The Adaboost generalized framework is then defined through Algorithm 2, where lines 5 and 7 respectively replace Adaboost's confidence and sample distribution update.

The main difference with the learning process of Adaboost is that instead of computing the relevance of each classifier h_t as a scalar value α_t, the generalization framework considers it as a vector $\boldsymbol{\omega}_t = (\boldsymbol{\omega}_t[i])_{i=1}^m$, of dimension the size of the learning sample, in order to keep the local relevance information, as shown on Figure 1b.

Relying on the fitted base classifiers and their associated weight vectors, the challenge of our algorithm framework is to design a prediction function able to estimate the relevance of each base classifier for an unseen test sample (x, y). Indeed, Adaboost relies on the hypothesis that the relevance of each classifier is uniform over the samples. Discarding this hypothesis is the base of our work. Therefore, the prediction

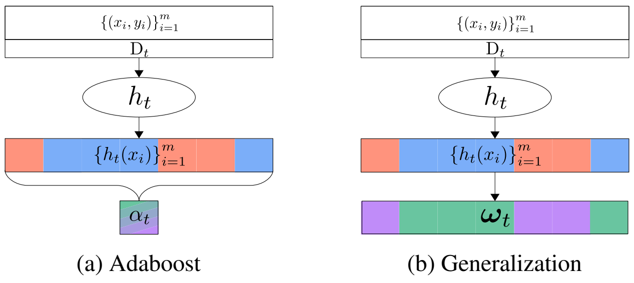


Figure 1: One iteration of greedy learning for Adaboost and Algorithm 2. The red and blue squares respectively represent failure and success on the training samples. The green and purple squares represent the relevance of the base classifier h_t. Note that Algorithm 2 stores the relevance as a vector.

function of the generalization framework becomes

$$\hat{y} = P^{\mathcal{F}_T}(x) = \sum_{t=1}^{T} h_t(x)\hat{\omega}_{\mathcal{L}}^{\mathcal{F}_T}(h_t, x), \qquad (1)$$

where $\hat{\omega}_{\mathcal{L}}^{\mathcal{F}_T} : \mathcal{H} \times \mathcal{X} \to [0, 1]$ is a function that approximates the relevance of h_t on an unseen sample (x, y). $\hat{\omega}_{\mathcal{L}}^{\mathcal{F}_T}$ depends on the information available in the training set $\mathcal{L}$ and the learned weights vector $\boldsymbol{\omega}_t$ for each classifier h_t of the ensemble. As a consequence, the weight of each classifier in the majority vote actually depends on the test sample classified by the ensemble.

This framework raises a number of questions, in particular about relevant definitions of the three central functions ω, π and $\hat{\omega}_{\mathcal{L}}^{\mathcal{F}_T}$ according to the task at hand. Adaboost is equivalent to one instantiation of this framework, as explained in Supplementary Material A. The following section studies another instantiation of that framework, which leads to SamBA, an algorithm intended to exploit proximity among samples in order to solve some difficulties raised by fat data specificities.

3 INTRODUCING SAMBA

SamBA has been designed as an instantiation of the generalization framework of Adaboost presented in the previous section, in order to deal with a specific family of datasets where $m \ll l$. These datasets are frequent in the biological applications of machine learning, and are called *fat* datasets [Romero et al., 2016], in contrast with *big* data. In such datasets, the description space where the best classifier is looked upon is huge w.r.t. the number of available samples, which could lead to overfitting and/or irrelevant dimension reductions. In this section, abstract concepts ω, π, h and $\hat{\omega}^{h_t,\mathcal{L}}$ are embodied and defined in order to overcome the challenges of fat data.

3.1 AN INSTANCE USING LOCAL KNOWLEDGE

Drawing from the efficiency of Adaboost, the relevance of a base classifier in SamBA is also computed as the exponential of the margin, example-wise. This relevance function replaces the *abstract* function of line 5 in Algorithm 2.

$$\boldsymbol{\omega}_t[i] := \exp\left(h_t(x_i)y_i\right). \quad (2)$$

Similarly, the difficulty π is defined as the inverse of the relevance, replacing the *abstract* function of line 7 in Algorithm 2.

$$\boldsymbol{\pi}_t(i) := \exp\left(-h_t(x_i)y_i\right). \quad (3)$$

In addition, we consider $\mathcal{H}$ to be a set of decision stumps on the features of $\mathcal{X}$. This allows for the final decision function to rely on a small subset of the features of $\mathcal{X}$ and implies some sparsity and interpretability of the decision process. In addition, we introduce the notion of support of a classifier.

Definition 1 (Support of a classifier)**.** *Considering any classifier h relying on a set of features of $\mathcal{X}$, its support $\tilde{\mathcal{X}}_h$ is the space projected on those features.*

Next, we define the relevance of classifier h_t over the entire input space $\mathcal{X}$ to be an estimation function based on the vote of each sample of the learning set, weighted by the Euclidean distance computed on the support of the prediction function. This shift in perspective is explained in Section 3.2. We note $d(x_i, x) := ||x_{|\tilde{\mathcal{X}}_{\mathcal{F}_T}} - (x_i)_{|\tilde{\mathcal{X}}_{\mathcal{F}_T}}||^2$ the Euclidean distance on the support of SamBA, with $x_{|\tilde{\mathcal{X}}}$ the projection of x on the support of SamBA after T iterations, $\tilde{\mathcal{X}}_{\mathcal{F}_T}$. Therefore, the weight estimation function of SamBA is defined as

$$\hat{\omega}_{\mathcal{L}}^{h_t}(x) = \hat{\omega}_t^{a,b}(x) := \begin{cases} \boldsymbol{\omega}_t[i] \text{ if } x = x_i \text{ and } a = 0, \\ \sum\limits_{i=1}^{m} \frac{\boldsymbol{\omega}_t[i]m}{a^b + d(x_i,x)^b} \text{ otherwise,} \end{cases} \quad (4)$$

where a, b are two hyper-parameters that control the importance of the Euclidean distance in the weight approximation process. The predicting function $P_{a,b}^{\mathcal{F}_T}$ for a vote of T voters on a sample x such that $x \neq x_i$, $\forall x_i \in \mathcal{L}$, can be written as

$$\hat{y} = P_{a,b}^{\mathcal{F}_T}(x) = \sum_{t=1}^{T} h_t(x) \left(\sum_{i=1}^{m} \frac{\boldsymbol{\omega}_t[i]m}{a^b + d(x_i, x)^b} \right). \quad (5)$$

As SamBA is an instance of Algorithm 2, for brevity's sake, we provide its pseudo-code in Supplementary Material B.

3.2 BEHAVIORAL INSIGHTS OF SAMBA

In this section, we provide insights on the inner mechanisms of SamBA, highlighting differences with Adaboost.

On the transfer of local knowledge to the prediction SamBA has been presented as a variation of Adaboost that allows to capitalize on the local knowledge acquired during the training phase through a weight estimation function. Hence, the weight estimation function is a central piece of the algorithm and highly impacts its prediction.

In SamBA, the weight of a classifier h_t on a test sample (x, y) is computed as $\sum_{i=1}^{m} \boldsymbol{\omega}_t[i] \frac{m}{a^b + d(x_i,x)^b}$. This function can be considered as a majority vote between the relevances $\boldsymbol{\omega}_t[i]$ of the classifier h_t on the training samples (x_i, y_i), weighted by their similarity with the test sample $\frac{m}{a^b + d(x_i,x)^b}$.

In SamBA the weight of a base classifier is thus mainly derived from the opinion of the nearest training sample, depending on the values of hyper-parameters a and b. In doing so, it can be considered at the crossroads of Adaboost and similarity-based methods such as KNN and SVM-RBF.

On the sparsity One of the claims of SamBA is to learn sparse decision functions. The main process that leads to the sparse decision function is that SamBA extracts much more information from each selected feature than classical boosting algorithms. Indeed, not only does SamBA learn decision stumps during the boosting process, it also builds a similarity-based decision function on the projected space. As a consequence, it requires a smaller number of iterations to build a decision function that fits the data, as seen in Section 5.2.2. The main drawback of such a method is its potential sensitivity to noise. This is why we introduced hyper-parameter a, which controls the smoothness of the decision border and is discussed in the following paragraphs.

On the restriction to the support As seen in the previous section, the vote of the samples is weighted by the similarity function; in our case, we base that similarity on the Euclidean distance between two samples, computed on the support of the decision function. This particular point is mandatory when learning on fat data. Indeed, greedy ensemble methods relying on decision stumps output supports with manageable dimensions. Coupling the dimension reduction and model learning processes in a single algorithm is a major advantage. With SamBA, we aim at outputting a decision function relying on a support of significantly smaller dimension than $\mathcal{X}$. Hence, computing the distance on the sole support is crucial to avoid the noise introduced by all the non-selected features.

On the role of a and b as hyper-parameters The hyper-parameters a and b are central in SamBA as they control the importance assigned to the distance. When setting $b = 0$, the weight is approximated by simply averaging the relevance of the classifiers on the entire training set: SamBA becomes a standard boosting algorithm, with no local expertise in the decision function. However, when $b > 1$, the similarity function plays a central role in the decision process, awarding more credit to the opinion of samples closer to (x, y).

The role of a is to ensure that the similarity function is bounded. Indeed, with $a = 0$, if a test sample is drawn too close to a training sample, the opinion of the training sample might be too strongly weighted. Therefore, it might overpower any other one, leading to overfitting, which is common with fat data. In Supplementary Material C, we show that $a = 0$ can be highly crippling in the presence of mislabeled data or outliers, and we provide more insight on the effect of a and b on the similarity function.

4 THEORETICAL ANALYSIS OF SAMBA

4.1 ALGORITHMIC COMPLEXITY

SamBA is slightly more complex than the usual boosting algorithms as its prediction function requires computing the distance vector between (x, y) and the m training samples. This distance is computed on $\tilde{\mathcal{X}}_{\mathcal{F}_T}$, the support of SamBA, of dimension $l' \leq T$. Therefore, computing the similarity function takes $\mathcal{O}(m \times l')$. Thus, to compute the weight of each voter during the prediction process, SamBA takes $\mathcal{O}(m \times l')$, and computing the whole decision function takes $\mathcal{O}(m \times (l' + T))$. Moreover, we denote $C(l, m)$ the complexity of learning one base classifier. As a consequence, the learning process takes $\mathcal{O}(T \times (m + C(l, m)))$ and the entire learning and predicting processes take $\mathcal{O}(T \times [m + C(l, m)] + m \times l')$ for one sample. For comparison, the entire learning and predicting process of a usual boosting algorithm with the same hypotheses takes $\mathcal{O}(T \times [m + C(l, m)])$.

Thus, in the fat data setting, where $m \ll l$, the learning and predicting process of SamBA is as costly as the one of a usual boosting algorithm if SamBA uses a small support $l' \leq T \ll l$. As a consequence, SamBA is better used on datasets with a manageable number of training samples m and a large number of features l. This fits our initial goal of extracting as much information as possible from a small subset of features of a fat dataset. This theoretical reasoning is validated in Section 5.1, in which we analyze the computational time of learning and predicting with SamBA.

4.2 TRAINING AND GENERALIZATION BOUNDS

In this section, we provide essential guarantees for SamBA, that bounds its capabilities for both training and generalizing. First, we prove that SamBA converges during training.

Theorem 1 (Error exponential decrease)**.** *If $a = 0$, let ϵ_t be the error of voter h_t on the training set, weighted by $\boldsymbol{\pi}_t$, $\gamma_t = \frac{1}{2} - \epsilon_t$ and $\boldsymbol{\pi}_1$ an arbitrary distribution over the training set. Then, the weighted training error of the combined classifier outputted by SamBA, 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. The proof resembles that for Adaboost, and is provided in Supplementary Material D. Note that, there are key modifications in order to fit our algorithm. □

This result is not surprising, as SamBA stores more information than Adaboost during training. The main theoretical contribution about SamBA is the following generalization bound. Of note, as the decision function of the algorithm computes different voter weights for each new test sample, classical boosting results based on VC-dimension or Rademacher complexity are hardly achievable.

As a consequence, we cannot derive a generalization bound from existing results on sample compression bounds for Adaboost[Schapire and Freund, 2012], as they rely on VC-dimension in addition to sample compress. Therefore, our bound relies on a different sample compress theory, the PAC-Bayesian sample compression framework [Laviolette and Marchand, 2007], which relates the majority vote classifier risk (coined as the Bayes risk in the PAC-Bayes literature) to the risk of the stochastic Gibbs classifier, taking into account the voter data dependency thanks to the sample compress (SC) framework [Floyd and Warmuth, 1995]. The term SC-classifier refers to a classifier that is defined from a small subset of the training set.

In the PAC-Bayesian framework, the Gibbs risk amounts to be the expected risk of the individual voters, while the Bayes risk[2] is simply the prevailing voters' output.

Definition 2 (Empirical Gibbs Risk)**.** *Given a distribution Q on a set of voters, the Gibbs risk on dataset $\mathcal{L}$ is*

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

Definition 3 (Theoretical Bayes Risk)**.** *Given a distribution Q on a set of voters, the Bayes risk relatively to $\mathcal{D}$ of the majority vote classifier B_Q is*

$$R_{\mathcal{D}}(B_{Q,\mathcal{L}}) = \mathbf{E}_{(x,y) \sim \mathcal{D}}\left[I\left(\mathbf{E}_{h \sim Q}\left[y \operatorname{sg}(h(x)) < 0\right]\right)\right],$$

with $I(p) = 1$ if predicate p is true, and 0 otherwise and sg *returning the sign.*

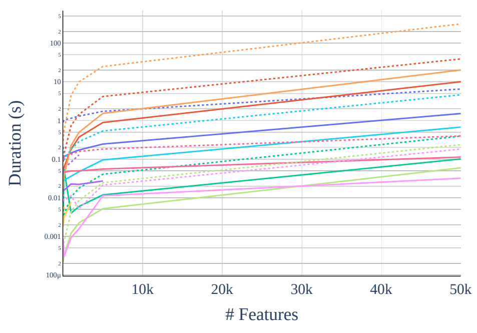
Based on these definitions, we present below a PAC-Bayes sample-compress bound derived from Germain et al. [2015].

Theorem 2 (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, with probability at least $1 - \delta$ over the choice of the training set,*

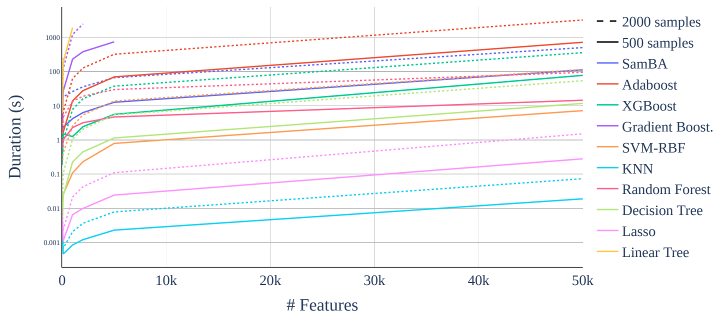
$$R_{\mathcal{D}}(B_{Q,\mathcal{L}}) \leq 2\left(R_{\mathcal{L}}(G_{Q,\mathcal{L}}) + \Psi\right),$$

where $\Psi = \sqrt{\frac{1}{2(m-3)}\left[KL(Q||P) + 12 + \ln\left(\frac{2\sqrt{m-3}}{\delta}\right)\right]}$.

[2]Not to be confused with the usual optimal Bayes predictor.



(a) Log test duration for each algorithm



(b) Log train duration for each algorithm

Figure 2: Learning and predicting 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 the number of base estimators outputted in Section 5.2.2, and the KNN to 5 neighbors.

Proof. The full proof is provided in Supplementary Material D.2. As an outline, we needed to extend the proof technique of Germain et al. [2015] to embrace our specific prediction function of Equation 5. We considered that the vote outputted by SamBA is not the vote of T classifiers $h_t(x)$, weighted by $\hat{\omega}_t^{a,b}(x)$, but the vote of $T \times m$ voters $\frac{h_t(x)m}{d(x_i,x)^b+a^b}$ representing the opinion of a sample, weighted by $\boldsymbol{\omega}_t[i]$. In Supplementary Material D.3, we also provide an additional bound that displays a smaller KL-divergence. □

5 EXPERIMENTS

In this section[3], we empirically compared SamBA to `sklearn` [Pedregosa et al., 2011] versions of Adaboost, Decision Tree (DT), Random Forest (RF), Gradien Bossting (GB), similarity-based methods such as SVM-RBF and KNN, Lasso's linear model [Tibshirani, 1996] and XGBoost's [Chen and Guestrin, 2016] implementation, regarding resource consumption, decision quality and sparsity. We then provide an interpretation of SamBA's decision. In this section, we only use decision stumps as base classifiers for SamBA. Indeed, in this work we aim at producing sparse decision functions. However, if focusing solely on performance, SamBA can be used with any base classifier.

5.1 TIME CONSUMPTION

First, we measure the time consumption of SamBA, compared to our pool of classifiers on increasingly bigger datasets. For each ensemble method, we set the number of iterations to the mean value outputted for the `go` dataset in Table 2. In Figure 2, we plot the duration on one thread to fit and predict on different sizes of synthetic datasets, with 500 and 2000 samples and with dimensions ranging from 10 to 50k.

These experiment results do not highlight any issue with SamBA's time consumption. Indeed, all the algorithms show similar training duration evolution, except for KNN that is faster. However, during the prediction process, all the methods are similar, while SVM-RBF is much longer. We also note that Adaboost uses more time than SamBA. This is due to the fact that in Section 5.2.2, Adaboost outputs a very dense decision function, and therefore need a larger number of base classifiers. In addition, we confirm that Linear Trees are not scalable on fat data, as their training duration is not manageable. This result highlights the fact that SamBA's sparsity is relevant on fat datasets[4].

5.2 CLASSIFICATION QUALITY AND SPARSITY

In this section, we first provide a visual cue on the differences between the algorithms on usual generated datasets. Second, we benchmark those algorithms on a real life biological dataset, providing both fat and non-fat descriptions of its samples.

5.2.1 On Generated Datasets

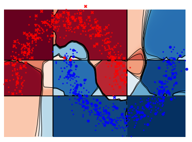
In this experiment, we focus on analyzing the fact that SamBA is able to separate generated datasets that have complex decision borders. To assess SamBA's relevance, we compare it to the same pool of algorithms as in the previous experiment, removing Linear Trees based on their resource consumption. Instead of focusing on outputting the best per-

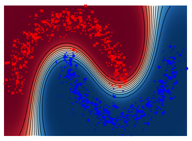
[3]All the code and data used in this section are available on GitHub https://github.com/babau1/samba and the detailed experimental protocols are provided in Supplementary Material E.

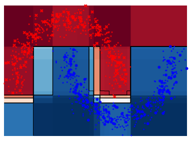
[4]In Supplementary Material F, we provide a similar experiment with the same number of base classifiers for each ensemble method, verifying that SamBA is still relevant in this case.

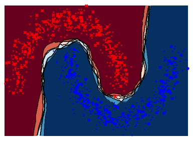
Dataset	SamBA	Adaboost	XGBoost	Grad. Boost.	SVM-RBF	KNN	Rand. For.	Dec. Tree	Lasso
Moons	0.99 $\pm$0.01	0.99 $\pm$0.01	1.0 $\pm$0.0	0.99 $\pm$0.01	0.99 $\pm$0.01	1.0 $\pm$0.0	0.99 $\pm$0.0	0.99 $\pm$0.01	0.86 $\pm$0.02
Spirals	0.99 $\pm$0.01	0.83 $\pm$0.02	0.98 $\pm$0.01	0.94 $\pm$0.01	0.9 $\pm$0.05	1.0 $\pm$0.0	0.99 $\pm$0.01	0.98 $\pm$0.01	0.6 $\pm$0.02
Moons Noisy	0.99 $\pm$0.01	0.99 $\pm$0.0	0.99 $\pm$0.01	1.0 $\pm$0.01	0.58 $\pm$0.04	0.65 $\pm$0.04	0.92 $\pm$0.03	0.98 $\pm$0.01	0.88 $\pm$0.03
Spirals Noisy	0.99 $\pm$0.01	0.71 $\pm$0.03	0.95 $\pm$0.01	0.83 $\pm$0.02	0.5 $\pm$0.0	0.5 $\pm$0.03	0.68 $\pm$0.04	0.91 $\pm$0.03	0.61 $\pm$0.03

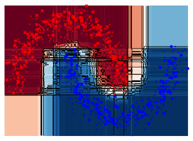
Table 1: Accuracy of greedy ensemble methods and similarity-based methods on generated datasets in two versions, one pure, and one with 50 noisy dimensions added.

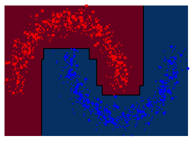

(a) SamBA- Moons

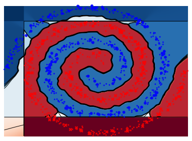

(b) SVM-RBF - Moons

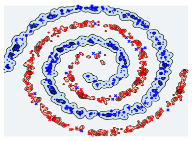

(c) Adaboost - Moons

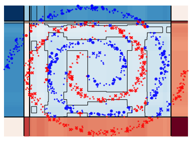

(d) KNN - Moons

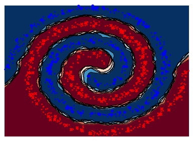

(e) RF - Moons

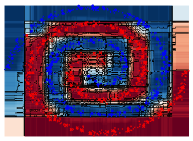

(f) DT - Moons


(g) SamBA- Spirals


(h) SVM-RBF - Spirals


(i) Adaboost - Spirals


(j) KNN - Spirals


(k) RF - Spirals

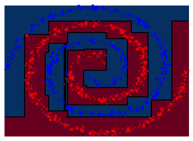

(l) DT - Spirals

Figure 3: Decision functions contour plots for the six considered algorithms, on the two *pure* generated datasets. The small dots are training samples, the big ones test samples. The color represents the predicted class and its intensity represents the certainty of the decision function on the 2D space. We provide full-size versions in Supplementary Material F, alongside the figures for Lasso, XGBoost and Gradient Boosting.

formance, we rather explore the behavior of each of these algorithms and their differences. The goal of this experiment is to show that SamBA has the capability to separate even complex datasets, while keeping the advantages of ensemble methods in the presence of noisy dimensions. The synthetic datasets of this experiment are generated in two versions. The first one considers only the relevant features, in 2D, facilitating visualization. The second one concatenates 50 additional features containing white noise, to assess the capability of each algorithm to deal with the presence of noisy features.

The results are presented in two media. First, Table 1 presents the test accuracies[5] of the algorithms on the four datasets: it shows that SamBA has a behavior comparable to similarity-based algorithms such as SVM-RBF on the *pure* datasets, outputting near-perfect performance. Moreover, we see that the added noisy dimensions highly impact the performance of SVM-RBF and KNN, but are not an issue for the greedy methods. We nuance these results, remarking that SamBA is more stable than its ensemble counterparts on the noisy *Spirals* dataset.

Second, Figure 3[6] shows the contour plots of the decision function of each algorithm on the non-noisy datasets. We plot the samples as dots, colored according to their predicted class, associated to a color map highlighting the certitude of each classifier on the 2D space. Figure 3 illustrates that SamBA's decision contour relies both on the stumps and the similarity between the samples: it seems to be at the crossroads between standard ensemble methods and similarity-based approaches. Figures 3a and 3g illustrate that SamBA is able to output very complex decision functions, which is an advantage on generated datasets with no outliers nor label noise. To complement these results, the following experiment highlights the fact that on real-life datasets, SamBA is competitive and outputs sparse decision functions.

[5]We compute the accuracy here, as the datasets are balanced; contrasting with the real-life experiments.

[6]In Supplementary Material F, we provide full-size versions.

5.2.2 On a Real-Life Dataset

Our initial goal is to apply SamBA on fat datasets aiming at outputting a decision function both sparse and relevant. Biological applications are an endless provider of fat data Drouin et al. [2019]. Indeed, in this domain, acquiring data on a single sample is costly, but each analysis yields a very large amount of information, denoted -omics data. Therefore, in a large majority of -omics applications of machine learning, the approaches have to deal with high-dimensional data. In this experiment, we apply our algorithms to a meta-gen*omics* imbalanced dataset [Deraspe et al., 2020] describing 640 patients, either obese (12%) or not (88%), with 8 types of data, ranging from 24 to 11946 features[7].

[7]We provide a more complete description of the dataset in Supplementary Material F.

Dataset	SamBA	Adaboost	XGBoost	Grad. Boost.	SVM-RBF	KNN	Rand. For.	Dec. Tree	Lasso
cog (24)	**.83** (10.7)	.77 (9.4)	.78 (9.3)	.76 (17.4)	.75 (all)	.77 (all)	.83 (21.3)	.72 (17.4)	.58 (12.75)
ec (2736)	**.84** (22.2)	.70 (58.6)	.70 (145.1)	.65 (963.8)	.74 (all)	.72 (all)	.84 (137.3)	.70 (19.0)	.65 (265.6)
go (11946)	.85 (21.5)	.73 (168.1)	.76 (62.7)	.71 (1394.4)	.62 (all)	.75 (all)	**.86** (191.3)	.73 (11.3)	.67 (574.1)
kegg.module (682)	**.85** (20.1)	.70 (44.9)	.68 (94.4)	.69 (333.0)	.71 (all)	.70 (all)	.83 (113.4)	.72 (16.2)	.62 (185.1)
kegg.pathway (414)	.82 (22.9)	.67 (87.6)	.69 (82.1)	.67 (208.0)	.73 (all)	.73 (all)	**.84** (186.3)	.69 (28.3)	.61 (73.2)
taxa.family (101)	**.82** (11.9)	.68 (48.3)	.66 (48.1)	.68 (79.4)	.65 (all)	.65 (all)	.82 (85.9)	.65 (27.5)	.61 (57.4)
taxa.phylum (37)	**.84** (7.7)	.70 (22.4)	.67 (27.9)	.66 (35.7)	.57 (all)	.63 (all)	.84 (32.4)	.74 (21.3)	.55 (18.9)
taxa.genus (72)	**.80** (14.8)	.63 (49.5)	.70 (42.4)	.70 (63.1)	.68 (all)	.68 (all)	.80 (69.1)	.73 (19.1)	.64 (47.6)

Table 2: Numerical results for all datasets. Each result is the mean over the 10 bootstrapped train/test splits. Best balanced accuracy is highlighted, when equivalent, we highlighted the sparsest approach. The mean size of the support of each algorithm is shown in parentheses. We provide standard deviations in Supplementary Material F.

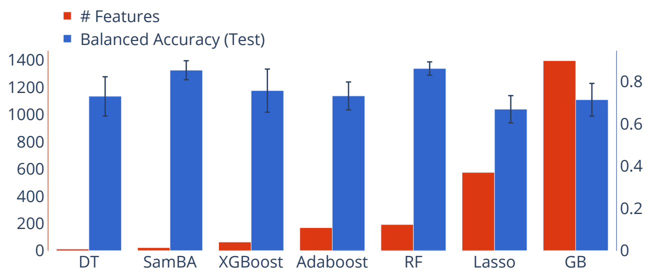


Figure 4: Visualization of the balanced accuracy and size of the support for each ensemble methods learned on the *go* dataset, of dimension 11946, ranked by support size.

Protocol The relevance of SamBA on real-life datasets is compared to the pool of classifiers. The benchmark was realized with SuMMIT [Bauvin et al., 2022], with a 10-iteration bootstrap holdout. The splitting is done by respecting the ratio between the classes, with 80% for learning and 20% for testing. All the classifiers were allowed 50 iterations of randomized search, to avoid the bias of grid search that promotes classifiers with a high number of hyper-parameters, such as SamBA. The hyper-parameters are validated through a 5-folds cross-validation process, and their performance is evaluated by the balanced accuracy to fit the imbalance.

Results Table 2 presents the mean balanced accuracies of all the approaches cited above alongside the number of features they base their model on. In Table 2, the best approaches in pure balanced accuracy are SamBA and the Random Forest. Although SamBA is consistently sparser than Random Forest, that outputs a very dense decision function. Note that SamBA outperforms boosting-based methods on all data types. In addition, SamBA is sparser than most boosted models, except on the `cog` dataset, which dimension is out of our scope of fat datasets. Similarity-based methods such as KNN and SVM-RBF are also outperformed by SamBA on all the datasets, suggesting that it inherits the best from both boosting and similarity based classifiers. However, we mention that SamBA, even if it is sparser than most of the approaches, outputs a generally denser decision than the Decision Tree, but is persistently more accurate.

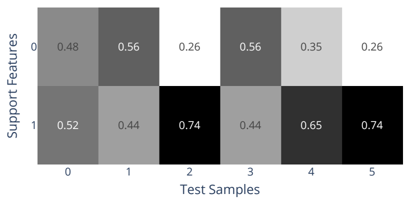


Figure 5: Heatmap of the importance of each support feature for a subset of test samples. A dark rectangle in row i, column j, means that feature i has been important in the classification of sample j.

Interestingly, SamBA is consistently sparser than all the other boosting methods: the 11946 dimensions *go* dataset is drastically reduced to 21 dimensions for SamBA's decision function, compared to Adaboost that uses 168. Figure 4 pictures both the balanced accuracy and the number of features used for *go*: SamBA displays the best ratio between balanced accuracy and sparsity. Indeed, if we overlook the very dense Random Forest, SamBA is the most relevant algorithm of the pool. This experiment illustrates that SamBA can be competitive with state-of-the-art algorithms concerning *feature efficiency*. Similar figures for each type of data are provided in Supplementary Material F.

5.3 INTERPRETING SAMBA'S DECISION

Interpretability is more and more important for machine learning models [Rudin et al., 2021]. Indeed, even if noninterpretable methods can rely on post-hoc explainability approaches [Molnar, 2022] to decipher their decisions, the gold standard is to be able to understand the decision without external tools. Therefore, the design of SamBA also focused on protecting the interpretability of Adaboost.

This section analyzes SamBA's prediction function on the previously introduced *Spirals* dataset. This task is not complex: Section 5.2.1 shows that SamBA solves it easily. In this experiment, we analyzed the decision function of SamBA on the samples of this dataset. To be able to analyze the behavior of SamBA, we trained it on a single train/test split with fixed hyper-parameters, and we outputted the weights it associates with each selected feature for a subset of the test set.

Figure 5 illustrates the individual feature importance for several test samples. We specifically chose the ones with the most variability to highlight the limit cases of SamBA. Firstly, some samples heavily rely on Feature 1. Indeed in the Spirals dataset, a sample on the outer edge of the spiral does not require a combination of the features to be well classified. Therefore, SamBA's weight approximation function allows prioritizing specific features for specific samples during the prediction process. For example, Sample 2 heavily relies on Support Feature 1, while Sample 3 relies on a more uniform distribution over the support features.

This experiment illustrates that SamBA is able to find a custom combination of weak classifiers for each sample, outputting a complex decision function from a sparser feature space than Adaboost. We also showed that interpreting this decision function can lead to better understanding of the mechanisms that lead to the classification of the test samples.

6 CONCLUSION AND FUTURE WORKS

This paper focuses on generalizing Adaboost's framework to enable learning with local knowledge. We first provided a new point of view of several ensemble methods as combinations of local experts. We then proposed a general framework for boosting algorithms to combine local experts. Relying on this abstract framework, we presented SamBA, an instance of the framework specifically designed to tackle the problem of fat data. We analyzed SamBA's behavior and proved theoretical properties leading to a generalization bound. We then presented four experiments highlighting the empirical properties of SamBA. Firstly, we validated the fact that SamBA does not consume critically more resources than state-of-the-art methods, specifically on fat datasets. Through a synthetic and a real-life fat dataset, we compared the performance of SamBA and state-of-the art algorithms. To conclude, we showed that, even if its decision is more complex than that of Adaboost, SamBA is still interpretable.

SamBA offers an original point of view on greedy ensemble methods. We showed in this work that SamBA is relevant on fat datasets, using a similarity function relying on the Euclidean distance. It would be interesting to further investigate the fully general framework, deriving algorithms fitted for different tasks. For example, the multi-environment problem would be a relevant application, with an adapted similarity function. Lastly, tighter generalization bounds could be investigated for other instances of the general framework.

Author Contributions

B. Bauvin conceived the algorithm, ran the experiments, wrote the code and the paper. C. Capponi and J. Corbeil are the supervisors of B. Bauvin. F. Clerc, P. Germain, and C. Capponi all contributed to the work leading to the generalization guarantees. J. Corbeil contributed to the formulation of the real-life experiments. S. Koço contributed in the feasibility tests of the original idea and the empirical study.

Acknowledgements

Importantly, we would like to acknowledge Pr. François Laviolette who passed away recently. He had a huge impact on our work and was a mentor. His guidance and feedback were important to us all. In addition, we warmly thank D. Benielli for her help in the first phase of this long project. We also warmly thank L. Ralaivola for his input in the early attempts at generalization proofs.

This work was supported by the Canada Research Chair in Medical Genomics to J. Corbeil; and scholarships from the Create Program to B. Bauvin; and the Big Data Research Center; the Institute of Intelligence and Data. Computations were performed under the auspices of Calcul Québec and Compute Canada. The operations of Compute Canada are funded by the Canada Foundation for Innovation (CFI), the NSERC, NanoQuébec and the FQRNT. F. Clerc is funded by IVADO through the DEEL project and by a grant from NSERC. P. Germain is supported by the Canada CIFAR AI Chair Program, and the NSERC Discovery grant RGPIN-2020-07223. S. Koço was in Aix-Marseille University when he worked on this project; but has left since.

References

Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning. *Lazy learning*, pages 11–73, 1997.

Baptiste Bauvin, Cécile Capponi, Jean-Francis Roy, and François Laviolette. Fast greedy c-bound minimization with guarantees. *Machine Learning*, 109(9):1945–1986, Sep 2020. ISSN 1573-0565. doi: 10.1007/s10994-020-05902-7. URL `https://doi.org/10.1007/s10994-020-05902-7`.

Baptiste Bauvin, Jacques Corbeil, Dominique Benielli, Sokol Koço, and Cecile Capponi. Integrating and reporting full multi-view supervised learning experiments using summit. In Nuno Moniz, Paula Branco, Luís Torgo, Nathalie Japkowicz, Michal Wozniak, and Shuo Wang, editors, *Proceedings of the Fourth International Workshop on Learning with Imbalanced Domains: Theory and Applications*, volume 183 of *Proceedings of Machine Learning Research*, pages 139–150. PMLR, 23 Sep 2022. URL `https://proceedings.mlr.press/v183/bauvin22a.html`.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

Leo Breiman. Random forests. *Machine Learning*, 45 (1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL `https://doi.org/10.1023/A:1010933404324`.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL `http://arxiv.org/abs/1603.02754`.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

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.

Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45014-6.

Alexandre Drouin, Sébastien Giguère, Maxime Déraspe, Mario Marchand, Michael Tyers, Vivian G. Loo, Anne-Marie Bourgault, François Laviolette, and Jacques Corbeil. Predictive computational phenotyping and biomarker discovery using reference-free genome comparisons. *BMC Genomics*, 17(1):754, Sep 2016. ISSN 1471-2164. doi: 10.1186/s12864-016-2889-6. URL `https://doi.org/10.1186/s12864-016-2889-6`.

Alexandre Drouin, Gaël Letarte, Frédéric Raymond, Mario Marchand, Jacques Corbeil, and François Laviolette. Interpretable genotype-to-phenotype classifiers with performance guarantees. *Scientific Reports*, 9(1): 4071, Mar 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-40561-2. URL `https://doi.org/10.1038/s41598-019-40561-2`.

Evelyn Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989. ISSN 03067734, 17515823. URL `http://www.jstor.org/stable/1403797`.

Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, 21(3):269–304, Dec 1995. ISSN 1573-0565. doi: 10.1023/A:1022660318680. URL `https://doi.org/10.1023/A:1022660318680`.

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`.

João Gama and Pavel Brazdil. Cascade generalization. *Machine Learning*, 41(3):315–343, Dec 2000. ISSN 1573-0565. doi: 10.1023/A:1007652114878. URL `https://doi.org/10.1023/A:1007652114878`.

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`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Sokol Koço and Cécile Capponi. A boosting approach to multiview classification with cooperation. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, pages 209–228, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23783-6. URL `https://link.springer.com/chapter/10.1007/978-3-642-23783-6_14`.

Charu Kothari, Mazid Abiodoun Osseni, Lynda Agbo, Geneviève Ouellette, Maxime Déraspe, François Laviolette, Jacques Corbeil, Jean-Philippe Lambert, Caroline Diorio, and Francine Durocher. Machine learning analysis identifies genes differentiating triple negative breast cancers. *Scientific Reports*, 10(1): 10464, Jun 2020. ISSN 2045-2322. doi: 10.1038/s41598-020-67525-1. URL `https://doi.org/10.1038/s41598-020-67525-1`.

François Laviolette and Mario Marchand. Pac-bayes risk bounds for stochastic averages and majority votes of sample-compressed classifiers. *J. Mach. Learn. Res.*, 8:1461–1487, 2007. doi: 10.5555/1314498.1314548. URL `https://dl.acm.org/doi/10.5555/1314498.1314548`.

Ruey-Hsia Li and Geneva G. Belford. Instability of decision tree classification algorithms. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 570–575, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 158113567X. doi: 10.1145/775047.775131. URL `https://doi.org/10.1145/775047.775131`.

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

Michael A. Nielsen. Neural networks and deep learning, 2018. URL `http://neuralnetworksanddeeplearning.com/`.

Mazid Osseni, Prudencio Tossou, Jacques Corbeil, and Francois Laviolette. Applying pyscmgroup to breast cancer biomarkers discovery. pages 72–82, 01 2021. doi: 10.5220/0010375500720082.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2 (11):559–572, 1901. doi: 10.1080/14786440109462720.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

John R Quinlan et al. Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. World Scientific, 1992.

Adriana Romero, Pierre Luc Carrier, Akram Erraqabi, Tristan Sylvain, Alex Auvolat, Etienne Dejoie, Marc-André Legault, Marie-Pierre Dubé, Julie G Hussin, and Yoshua Bengio. Diet networks: thin parameters for fat genomics. *arXiv preprint arXiv:1611.09340*, 2016.

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`.

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012. ISBN 0262017180.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL `http://jmlr.org/papers/v9/vandermaaten08a.html`.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80023-1. URL `https://www.sciencedirect.com/science/article/pii/S0893608005800231`.