COPYING MACHINE LEARNING CLASSIFIERS

A PREPRINT

Irene Unceta
BBVA Data & Analytics
Universitat de Barcelona
Barcelona, Spain
irene.unceta@bbvadata.com

Jordi Nin
BBVA Data & Analytics
Universitat de Barcelona
Barcelona, Spain
jordi.nin@bbvadata.com

Oriol Pujol
Dept. of Mathematics and Computer Science
Universitat de Barcelona
Barcelona, Spain
oriol_pujol@ub.edu

March 6, 2019

ABSTRACT

We study model-agnostic copies of machine learning classifiers. We develop the theory behind the problem of copying, highlighting its differences with that of learning, and propose a framework to copy the functionality of any classifier using no prior knowledge of its parameters or training data distribution. We identify the different sources of loss and provide guidelines on how best to generate synthetic sets for the copying process. We further introduce a set of metrics to evaluate copies in practice. We validate our framework through extensive experiments using data from a series of well-known problems. We demonstrate the value of copies in use cases where desiderata such as interpretability, fairness or productivization constrains need to be addressed. Results show that copies can be exploited to enhance existing solutions and improve them adding new features and characteristics.

Keywords Classification · Copy Theory · Model-agnostic · Model Enhancement · Copy Fidelity · Interpretability · Fairness · Productivization

1 Introduction

In many every-day examples specific constraints hold that prevent optimal performance of state-of-the-art machine learning. Either data or the models themselves are subject to privacy restrictions [1], [2], [3] or specific regulations apply that require models to be self-explanatory [4], [5], [6] or fair with respect to sensitive data attributes [7], [8], [9]. Other issues include production pipeline limitations, where model complexity is limited by the capacity of company production environments to deliver certain solutions to the market [10] or the need to adapt to an ever changing context. To the best of our knowledge, these constraints have been traditionally addressed individually and by means of tailored solutions. As a result, it is often the case where current out-of-the-shelf machine learning techniques either can not be applied or yield only sub-optimal solutions.

In this work, we study how model-agnostic copies of machine learning classifiers can be used to frame and address the former challenges in a holistic manner. Given a classifier, a copy is a new machine learning model built to reproduce its decision function. We explore the scenario in which previously trained classifiers can be queried, while keeping their internals secret. In this context, we exploit query outputs to obtain a synthetic set of labelled data points and use them to build copies.

One of the main benefits of copying is that features and properties of models can be modified to our advantage. Copies retain the original accuracy while incorporating new features, so that we can enhance existing solutions and adapt them to new needs, when necessary. Specially when it comes to tackling the aforementioned limitations. We summarize the main contributions of this paper as:

- We identify the problem of copying separately from that of learning and develop its theory from scratch.
- We propose a flexible methodology to copy any given classifier in a model-agnostic manner and extensively validate it on well known problems.
- We identify the theoretical sources of loss in the copying process and define a series of practical performance metrics to evaluate copies.
- We illustrate the utility of copying in different scenarios and use cases. First, we address the is-

sues of non-decomposability and delayed time-to-market delivery in non-client mortgage risk scoring and use copies to move to high-performance regulatory-compliant models. Second, we show how a batch classifier can be replaced by an online copy and recover a critical operating point in a loan default prediction problem. Third, we demonstrate the usefulness of copies to ensure a fair classification when predicting superhero alignment.

The rest of this article is organized as follows. *Section 2* presents a literature survey of related work, followed by a description of several copy applications in *Section 3*. The theoretical framework for copying is derived in *Section 4*. In *Section 5* we study copying in practice. We characterize the different error contributions and propose the single-pass copy as the simplest form of copying in real situations. Further, *Section 6* introduces a series of metrics to evaluate copies. In *Section 7* we validate copies through their performance in various of problems and illustrate their usefulness in a number of use cases. We examine the implications of using copies in *Section 8*. The paper concludes with a brief summary of our findings and an outline of future research.

2 Related work

The concept of copies or surrogate models is not new in the literature. We find examples of surrogates in the domain of concept extraction, where trained artificial neural networks are compiled into a set of representative rules [11], [12], [13], [14]. Of particular interest to this article is TREPAN [15], a query algorithm that extracts tree-structured representations of trained neural networks. TREPAN amplifies the dataset used to train the neural networks with new samples generated by randomly selecting values for each attribute and labeling them according to the predictions of the net. After a thorough literature review, we consider this to be the first clear predecessor of a copying algorithm.

Model extraction has been traditionally treated as a standard learning process, where original training data are relabelled and extended to learn an alternative model [16]. Notably, [17] uses pseudo training sets based on the distribution of the original data to compress complex ensemble classifiers into surrogate neural nets. Further, [18] proposes three methods for generating such pseudo training data: RANDOM draws independent samples from the marginal distribution for each attribute, while the other methods, NBE and MUNGE, sample from an estimate of the joint attribute density. Thus, new data used to train the compressors are extracted from the estimated density of the original attributes. However, when copying, one is not tied to the original data distribution.

Similarities can be identified between our work and the topic of adversarial learning [19], [20], [21], [22], where a malicious adversary with access to a query interface exploits knowledge of a model to compromise it. In this setting an attack vector is defined that takes advantage of the vulnerabilities in a local surrounding of the decision boundary to foil the system. In this sense, both adversarial methods and copies may require to model the decision boundary [23],[24], [25]. Yet, in contrast with adversarial learning, our goal is to globally replicate the original classification boundary without loss of accuracy and with the added benefit of endowing the copy with new properties and functionalities not available in the original system.

3 Why should we care about copies?

Leaving the discussion of how to copy machine learning classifiers for the next section, we now elaborate on the usefulness and relevancy of copies in a wide variety of scenarios. In particular, we present the reader with four specific use cases that we later develop in greater depth.

Interpretability. Recent advances in the field of machine learning have led to increasingly sophisticated models, capable of learning ever more complex problems to a high degree of accuracy. This comes at the cost of simplicity [26], [27], a situation that stands in contrast to the growing demand for transparency in automated processing [4],[5], [6]. Recent works have been aimed at interpreting how black-box models work by providing local explanations of individual predictions [28], [29]. We here propose a novel approach to interpretability, whereby models of any arbitrary type can be approximated by copies specifically designed to be globally self-explanatory. Complex models can therefore be substituted with simpler, interpretable ones.

Production. Machine learning deployment is often costly in company environments [30], [10], [31], [32]. Common issues include the inability to maintain the technological infrastructure up to date with latest software releases, conflicting versions or incompatible research and deployment environments. Consider the case of neural network library Tensorflow. Even while the library itself provides detailed instructions on how to serve models in production [33], it typically requires several third-party components for docker orchestration such as Kubernetes or Elastic Container Service [34], which are seldom compatible with on-premise company software infrastructure. In contrast, moving to a copy developed in a less demanding environment helps bridge the gap between the data science and engineering departments.

Fairness. Machine learning models can reproduce existing patterns of discrimination [7], [9]. Some algorithms have been reported to be biased against people with protected characteristics like race [35, 36, 37, 38], gender [39, 40] or sexual orientation [41]. Solutions to the problem of fair learning often come in the form of sophisticated metrics that constrain models not to be discriminative with respect to sensitive attributes [42], [43]. While successful

when it comes to reducing unfairness in classification and regression systems, these solutions are not always easy to implement. Here, we argue that desiderata such as equity of learning can be directly imposed upon copies.

Enhancement. More generally, copies can be used to enhance existing models. Notably, they can be used to evolve from batch to online learning schemes [44]. This extends a model's lifespan as it enables adaptation to data drifts or performance deviations. Equivalently, when new class labels appear during a model's deployment in the wild, copies can be built to account for the new data points and evolve from binary to multiclass classification settings [45].

4 A model agnostic framework for copying

In what follows, we fully characterize the problem of copying. We develop the theoretical background for copying and suggest a pipeline to copy machine learning classifiers in practice. Note that we only consider supervised classification, partly due to the wider application of classifiers as opposed to, for example, regressors. Yet, the framework here presented is meant to be easily extensible to regression, clustering or any other type of process.

4.1 A glimpse into the copying process

For the sake of clarity, we first describe the most intuitive and simplest instantiation of the more general process of copying: the *single-pass copy*.

Given a decision function $f_{\mathcal{O}} : \boldsymbol{s} \rightarrow t$ from input instances to targets, the goal of copying is to obtain an alternative function $f_{\mathcal{C}}$ such that $f_{\mathcal{C}}(\boldsymbol{z}) = f_{\mathcal{O}}(\boldsymbol{z})$ for any $\boldsymbol{z}$ in the data domain $\mathcal{D}$. For this purpose, we assume the attribute domain $\mathcal{D}$ to be known. We also assume that we can interact with the model $f_{\mathcal{O}}$ through membership queries. We define a probability distribution P_Z over $\mathcal{D}$ and draw random samples $\boldsymbol{z}_j \sim P_Z$ for $j = 1, ..., N$. We use these samples to query $f_{\mathcal{O}}$ for predicted class labels, $y_j = f_{\mathcal{O}}(\boldsymbol{z}_j)$. As a result, we obtain a set of labeled pairs $\boldsymbol{Z} = \{(\boldsymbol{z}_j, y_j)\}$. We use these data to build a second classifier $f_{\mathcal{C}}$, that we refer to as the copy and which can be used to substitute $f_{\mathcal{O}}$ to a high degree of fidelity.

The main hypothesis is that it is possible to build a classifier that mimics the behavior of another, provided an optimal set of points can be generated that faithfully represents the target decision function. An example for the single-pass copy is shown in Fig. 1, where a decision tree classifier is used to copy the binary decision function learned by a fully-connected neural network.

4.2 From learning to copying

Let us assume a set of pairs $\boldsymbol{S} = \{(\boldsymbol{s}_i, t_i)\}$ for $i = 1, ..., M$, composed of individual observations of d-dimensional instances $\{\boldsymbol{s}_1, ..., \boldsymbol{s}_M\}$ and their corresponding target values $\{t_1, ..., t_M\}$. In the traditional machine learning setting, we are given the data $\boldsymbol{S}$ along with a new test point $\boldsymbol{x}^*$ and our goal is to predict the value of the corresponding label t^*. This goal is represented by the predictive distribution $P(t^*|\boldsymbol{S}, \boldsymbol{x}^*)$. As a result of the learning process, we obtain a model $f_{\mathcal{O}}$, defined as a function $f_{\mathcal{O}} : \boldsymbol{s} \rightarrow \boldsymbol{t}$ that maps input vectors to targets. In the case where $f_{\mathcal{O}}$ is a classifier, it returns a nominal variable ranging over a set of classes K. In what follows, we refer to model $f_{\mathcal{O}}$ as the *original* and accordingly name $\boldsymbol{S}$ the *original dataset*.

When copying, the original dataset is unknown. In its absence, we query the original, $f_{\mathcal{O}}$. The problem of copying therefore consists of building a model, parameterized by θ, that given a new sample $\boldsymbol{x}^*$ predicts the output $y^* = f_{\mathcal{O}}(\boldsymbol{x}^*)$. This problem is characterized by the predictive distribution $P(y^*|f_{\mathcal{O}}, \boldsymbol{x}^*)$. We marginalize this distribution with respect to the copy parameters θ and obtain

$$P(y^*|f_{\mathcal{O}}, \boldsymbol{x}^*) = \int_{\theta \in \mathcal{H}_C} P(y^*|\theta, f_{\mathcal{O}}, \boldsymbol{x}^*) P(\theta|f_{\mathcal{O}}, \boldsymbol{x}^*) d\theta,$$

for $\mathcal{H}_C$ the complete parameter space for the copy.

Let us make two assumptions. First, once having built the copy, *i.e.* fixed the value of θ, interaction with the original is no longer required, so that $P(y^*|\theta, f_{\mathcal{O}}, \boldsymbol{x}^*) = P(y^*|\theta, \boldsymbol{x}^*)$. Second, knowledge about the unseen data point $\boldsymbol{x}^*$ is not needed to build the copy, so that $P(\theta|f_{\mathcal{O}}, \boldsymbol{x}^*) = P(\theta|f_{\mathcal{O}})$. On this basis, we rewrite the expression above as

$$P(y^*|f_{\mathcal{O}}, \boldsymbol{x}^*) = \int_{\theta \in \mathcal{H}_C} P(y^*|\theta, \boldsymbol{x}^*) P(\theta|f_{\mathcal{O}}) d\theta.$$

We take a *winner takes it all* approach and force the posterior to have the form of a point mass density, $P(\theta|f_{\mathcal{O}}) = \delta(\theta - \theta^*)$, for $\delta(.)$ the Dirac delta function. All the probability mass is then placed onto the optimal θ^*, so that

$$P(y^*|f_{\mathcal{O}}, \boldsymbol{x}^*) = P(y^*|\theta^*, \boldsymbol{x}^*),$$

where the optimal parameter set θ^* can be defined as the maximum *a posteriori*

$$\theta^* = \arg\max_{\theta} P(\theta|f_{\mathcal{O}}). \tag{1}$$

The solution to the equation above defines the copy model that best approximates the original. Because we can only interact with $f_{\mathcal{O}}$ through querying, to compute the optimal θ we access the form of the original decision function through its predictions over a set of synthetic samples. Hence, we introduce the *synthetic dataset*.

Let $\boldsymbol{Z} = \{(\boldsymbol{z}_j, y_j)\}$ for $j = 1, ..., N$, be a set of labelled pairs where $\{\boldsymbol{z}_1, ..., \boldsymbol{z}_N\}$ are artificially generated samples

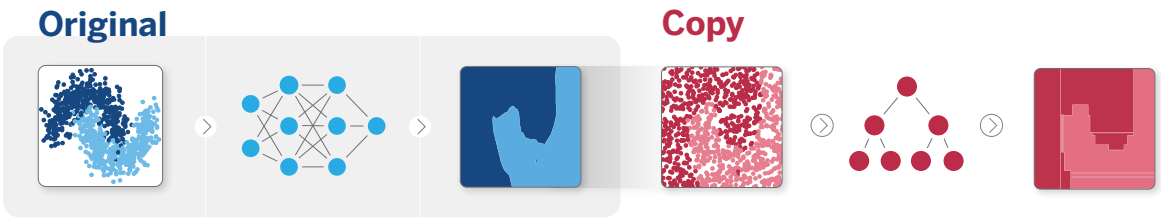


Figure 1: Example for the *single-pass copy*, the simplest form of copying pipeline.

and $\{y_1, ..., y_N\}$ their labels predicted by the original, so that $y_j = f_{\mathcal{O}}(\boldsymbol{z}_j)$. We refer to $\boldsymbol{Z}$ as the synthetic dataset and use it to access the information in $f_{\mathcal{O}}$ through its prediction $f_{\mathcal{O}}(\boldsymbol{z}_j)$ for any given sample. Thus, in practice, we maximize (1) by means of the following integral

$$\theta^* = \arg\max_{\theta} \int_{\boldsymbol{z}\sim P_Z} P(\theta|f_{\mathcal{O}}(\boldsymbol{z}))dP_Z, \tag{2}$$

where $\boldsymbol{z} \sim P_Z$ is drawn independently from a generating distribution P_Z that defines the support of the copying process, *i.e.* the plausible operational space for the copy. P_Z can be related to the distribution of the original training data, P_S, but note that this is not mandatory. Take for example the case of copying in a production setting. In this scenario, we can assume we have access to the original data distribution and therefore we can replace P_Z with P_S. Now consider the case of a completely separable binary problem, as that in Fig. 2, where each class comes from a Gaussian distribution and the decision boundary lies in a low density area of the space. By forcing $P_Z = P_S$ we ensure that the copy has the same behavior in those parts of the space where original data lie. However, the copy may display a completely different behavior around the boundary, where original data are scarce. An interesting modelling question in this scenario would be: *what should the copy do in corner cases?*. Another extreme case is that of counterfactuals, where we might want to include operation regimes of the original classifier even in front of impossible events and data values.

Having chosen an appropriate form for the probability distribution P_Z, we further develop the expression above. In particular, since maximizing the posterior is equal to maximizing the log-posterior, we can rewrite (2) as

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \left[\log\left(\int_{\boldsymbol{z}\sim P_Z} P(\theta|f_{\mathcal{O}}(\boldsymbol{z}))dP_Z\right)\right] \\ &= \arg\max_{\theta} \left[\log\left(\int_{\boldsymbol{z}\sim P_Z} \frac{P(f_{\mathcal{O}}(z)|\theta)P(\theta)}{P(f_{\mathcal{O}}(z))}dP_Z\right)\right],\end{aligned}$$

where we apply Bayes' rule to the terms inside the integral. We can now use *Jensen's inequality*[1] to provide a lower bound for the objective of the form[2]

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \int_{\boldsymbol{z}\sim P_Z} \log\left(\frac{P(f_{\mathcal{O}}(z)|\theta)P(\theta)}{P(f_{\mathcal{O}}(z))}\right)dP_Z \\ &= \arg\max_{\theta} \left[\int_{\boldsymbol{z}\sim P_Z} \log P(f_{\mathcal{O}}(z)|\theta)dP_Z - \right. \\ &\qquad \left. \int_{\boldsymbol{z}\sim P_Z} \log P(f_{\mathcal{O}}(z))dP_Z + \log P(\theta)\right],\end{aligned}$$

and since the term $\int_{\boldsymbol{z}\sim P_Z} \log P(f_{\mathcal{O}}(z))dP_Z$ has no dependence on θ it can be dropped to obtain

$$\theta^* = \arg\max_{\theta} \left[\int_{\boldsymbol{z}\sim P_Z} \log P(f_{\mathcal{O}}(z)|\theta)dP_Z + \log P(\theta)\right]. \tag{3}$$

The specific form of the solution to the equation above depends, among other things, on the characteristics of the considered models. In this seminal article we are interested in studying hard decision copies. Under this framework, we can recover regularized empirical risk minimization models [46] by approximating the distributions in (3) with an exponential family, so that

$$P(f_{\mathcal{O}}(\boldsymbol{z})|\theta) \propto e^{-\gamma_1 \ell_1(f_{\mathcal{C}}(\boldsymbol{z},\theta), f_{\mathcal{O}}(\boldsymbol{z}))}; \quad P(\theta) \propto e^{-\gamma_2 \ell_2(\theta,\theta^+)},$$

for $\ell(a, b)$ a measure of the disagreement between a and b, and θ^+ our prior knowledge about the copy parameters θ. Note that we here explicitly introduce the copy model as a function $f_{\mathcal{C}}(\boldsymbol{z}, \theta) : \boldsymbol{z} \to y$ from synthetic data instances $\boldsymbol{z}$ to predicted class labels $y = f_{\mathcal{O}}(\boldsymbol{z})$, defined by the parameters θ. We use this approximation to rewrite (3) as

[1] *Jensen's inequality* states that for any concave function f it holds that $E[f(X)] \leq f(E[X])$. Note that since the $\log(x)$ function is indeed concave, we can directly apply the inequality.

[2] Note that even though maximization of the lower bound also maximizes the original function, the optimal value of the lower bound may be different from that of the original objective function.

$$\theta^* = \arg\min_{\theta} \Bigg[\int_{\boldsymbol{z}\sim P_Z} \gamma_1 \ell_1(f_{\mathcal{C}}(\boldsymbol{z},\theta), f_{\mathcal{O}}(\boldsymbol{z}))dP_Z + \gamma_2 \ell_2(\theta,\theta^+) \Bigg], \tag{4}$$

where the first term is the expected value of the disagreement between original and copy and the second term refers to the fit of the parameters to the prior. To solve this equation, we approximate the integral with

$$(\theta^*, \boldsymbol{Z}^*) = \arg\min_{\theta, \boldsymbol{z}_j \in \boldsymbol{Z}} \Bigg[\frac{1}{N} \sum_{j=1}^{N} \gamma_1 \ell_1(f_{\mathcal{C}}(\boldsymbol{z}_j,\theta), f_{\mathcal{O}}(\boldsymbol{z}_j)) + \gamma_2 \ell_2(\theta,\theta^+) \Bigg], \tag{5}$$

for $\boldsymbol{Z}$ a synthetic dataset sampled from P_Z. The expression above is a double optimization: we simultaneously optimize the copy parameters θ and the synthetic set $\boldsymbol{Z}$. Importantly, the need to find the optimal $\boldsymbol{Z}$ introduces a new challenge: *which set of synthetic points minimizes the approximation error when substituting the integral in (4) with a finite sum over N elements?* We address this question in the following sections. Finally, we emphasize that this derivation is agnostic to the original data distribution P_S, which we assume to be unknown. Moreover, the only dependence on the original $f_{\mathcal{O}}$ appears through its predictions over a synthetic set, so that the process is also agnostic to the original model type.

4.3 Generalization in copies

Copying differs from learning, as traditionally understood by the machine learning community. When copying we do not exploit the original training data $\boldsymbol{S}$. Instead, we refer to the original decision function $f_{\mathcal{O}}$. As a result, we end up with the dual optimization in (5). Here, the class membership prediction output by the original defines a partition of the space, so that we are confronted with a hard classification boundary with two important characteristics: (i) the synthetic dataset is always separable and (ii) a potentially infinite stream of synthetic data are accessible. In what follows, we illustrate how these two properties shape the process of copying and distance it further from that of learning. We do so drawing connections with the empirical risk minimization framework.

First of all, observe that the form of the first term in (5) is that of empirical risk minimization, where the risk or expected loss particularized to our copying problem is defined over a distribution P_Z as

$$R^{P_Z}(f_{\mathcal{C}}(\boldsymbol{z},\theta), f_{\mathcal{O}}(\boldsymbol{z})) = \mathbb{E}_{\boldsymbol{z}\sim P_Z}[\ell_1(f_{\mathcal{C}}(\boldsymbol{z},\theta), f_{\mathcal{O}}(\boldsymbol{z}))], \tag{6}$$

and approximated by the empirical risk as

$$R^{P_Z}_{emp}(f_{\mathcal{C}}(\boldsymbol{z},\theta), f_{\mathcal{O}}(\boldsymbol{z})) = \frac{1}{N} \sum_{j=1}^{N} \ell_1(f_{\mathcal{C}}(\boldsymbol{z_j},\theta), f_{\mathcal{O}}(\boldsymbol{z_j})).$$

In this context, the second term of (5) corresponds to the regularization term, so that

$$\Omega(\theta) = \ell_2(\theta,\theta^+).$$

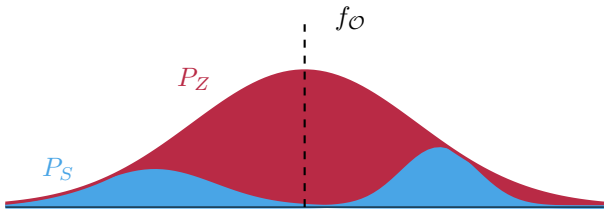


Figure 2: Original gaussian data distribution P_S and learned decision boundary $f_{\mathcal{O}}$. Alternative gaussian distribution for P_Z.

Recall that this formulation can be interpreted as a way of applying the concentration of measure inequalities to bound the generalization error in terms of the empirical risk,

$$R \leq R_{emp} + \mathcal{O}\left(\sqrt{\frac{C}{N}}\right),$$

for C a parameter that governs the classifier capacity and N the size of the synthetic dataset. Although not all generalization bounds have this same form, we find this trade-off between a capacity measure and the number of samples in all of them, including VC-dimension [47] [48] or covering numbers approaches [49], Rademacher complexity frameworks [50], PAC-Bayes bounds on distributions of hypothesis[51] and compression bounds[52].

Assuming this formulation, the dual optimization in (5) can be understood as the scalarization of a multi-objective optimization function $(R_{emp}(), \Omega)$, for γ_2 the parameter that controls the trade off between the empirical risk and the capacity term $C \approx \Omega(\theta)$. The solution to this optimization problem defines a Pareto's optimal surface, from which the optimal point is usually chosen using a validation dataset. Many algorithms, including SVMs, neural networks, boosting or Bayesian models are examples of problems of this form. When copying, given the separability of the synthetic dataset and assuming we select a model with enough capacity, it is always possible to achieve zero empirical error, $R^{P_Z}_{emp}(f_{\mathcal{C}}(\boldsymbol{z},\theta), f_{\mathcal{O}}(\boldsymbol{z})) = 0$. The error then only depends on the generalization gap for the synthetic dataset and, since we can generate infinite synthetic data, this value can be asymptotically reduced to zero. Hence, in theory, copying can be performed wi-

thout loss in accuracy and redefined as an unconstrained optimization problem of the form

$$\underset{\theta, \boldsymbol{Z}}{\text{minimize}} \quad R^{P_Z}_{emp}(f_{\mathcal{C}}(\boldsymbol{z}, \theta), f_{\mathcal{O}}(\boldsymbol{z})). \tag{7}$$

Yet, in practice, the synthetic set is finite. So that it stands to reason to impose that the copy have small capacity and rewrite the copying problem as

$$\begin{aligned} &\underset{\theta, \boldsymbol{Z}}{\text{minimize}} \quad \Omega(\theta) \\ &\text{subject to} \quad \|R^{P_Z}_{emp}(f_{\mathcal{C}}, f_{\mathcal{O}}) - R^{P_Z}_{emp}(f^{\dagger}_{\mathcal{C}}, f_{\mathcal{O}})\| < \epsilon, \end{aligned} \tag{8}$$

for $f^{\dagger}_{\mathcal{C}}$ the copy model obtained as a solution to the unconstrained problem in (7) and ϵ a defined tolerance[3]. The term $\|R^{P_Z}_{emp}(f_{\mathcal{C}}, f_{\mathcal{O}}) - R^{P_Z}_{emp}(f^{\dagger}_{\mathcal{C}}, f_{\mathcal{O}})\| < \epsilon$ defines a feasible set of parameters. The solution to (8) achieves the smallest capacity while keeping $R^{P_Z}_{emp}(f_{\mathcal{C}}, f_{\mathcal{O}})$ within a tolerance of the unconstrained optimal value of the empirical risk, $R^{P_Z}_{emp}(f^{\dagger}_{\mathcal{C}}, f_{\mathcal{O}})$. We argue that there exists a set of parameters θ that fulfill this constraint.

In certain classification problems the optimal loss value is known in advance (consider the hinge-loss used in SVMs, where $R^{P_Z}_{emp}(f^{\dagger}_{\mathcal{C}}, f_{\mathcal{O}}) = 0$). However, this is not always the case (consider least-square errors in classification)[4]. Indeed, the problem above is different from the standard multi-objective optimization in a pure learning setting, where the optimal values of both the loss and the regularization term are unknown. Instead of having a Pareto's surface of plausible optimal solutions, as long as $\Omega(\theta)$ is convex, the solution to our problem is unique.

The optimization problem in (8) can be straightforwardly applied in cases where the capacity is directly modelled, such as those of SVMs and neural networks, using a regularization function, or Bayesian models, selecting the priors. For other models, such as trees, the complexity control must be done by either early stopping or by an external process, such as post-pruning or pre-pruning techniques. Nevertheless, note that techniques such as boosting or deep learning may exhibit a delayed overfitting effect [53, 54, 55]. This property can be exploited to our advantage during the copying process to directly solve (7) instead of (8).

In Fig. 3 we show an example of how the formalization above applies to the particular case of a radial basis function kernel SVM. We use this model to copy the original decision function shown in Fig. 1 for the artificial neural network. Specifically, Fig. 3(a) shows the decision function learned by the copy SVM for a maximal value

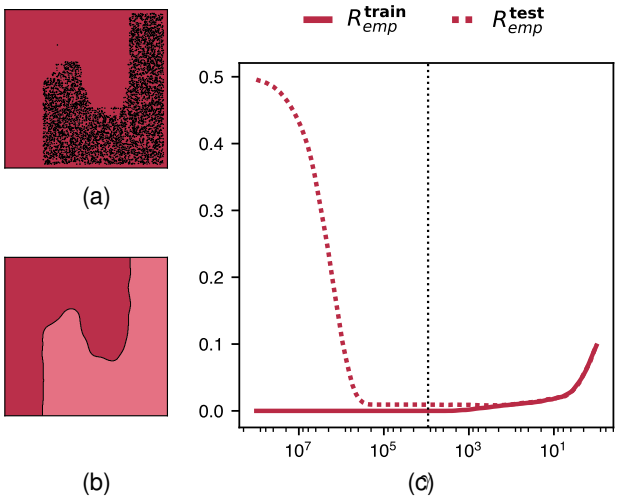


Figure 3: Decision boundaries learned by copies with (a) a maximal and (b) an optimal γ. (c) Empirical risk and generalization error for decreasing values of γ. The dotted black line refers to the optimal γ.

of parameter γ, which controls the width of the kernel $\mathcal{K}(x, x') = e^{-\gamma||x-x'||^2}$ and therefore the capacity of the model. Fig. 3(b) shows the boundary learned for an optimal capacity parameter γ, computed according to (8) for a tolerance parameter $\epsilon = 1e-4$. When comparing both plots we observe the improvement in generalization performance. This improvement is also seen in Fig. 3(c), where train and generalization errors are shown for decreasing values of γ. For a bounded value of the empirical error, the generalization error is reduced as we decrease the capacity of the copy.

5 Copying: from theory to practice

In what follows we bridge the gap between theory and practice. We characterize the elements that interact during the process of copying and introduce the *single-pass copy* as the simplest approach to copying. Under this approach, we provide guidelines on how to perform the different steps.

5.1 Distilling the process of copying

We formalize the idea of copying by introducing the notion of a *fidelity error* $R_{\mathcal{F}}$, which quantifies the disagreement between the original decision function $f_{\mathcal{O}}$ and the copy $f_{\mathcal{C}}$. This error corresponds to the particularization of the expression in (6) for the $0/1$ loss, and can be written as

$$\begin{aligned} R_{\mathcal{F}} &= P_Z(f_{\mathcal{O}}(\boldsymbol{z}) \neq f_{\mathcal{C}}(\boldsymbol{z}, \theta)) \\ &= \mathbb{E}_{\boldsymbol{z} \sim P_Z}[\mathbb{I}_{\{\boldsymbol{z}: f_{\mathcal{O}}(\boldsymbol{z}) \neq f_{\mathcal{C}}(\boldsymbol{z}, \theta)\}}(\boldsymbol{z})], \end{aligned}$$

[3] We here abuse notation and drop the explicit dependence on the synthetic data points $\boldsymbol{z}$ and copy parameters θ. In what follows, we sometimes favour this more concise notation.

[4] Instead of tracking the empirical risk we can track the empirical error, which can be set to zero due to the separability property.

for $\mathbb{I}$ the indicator function. Moreover, if we assume $f_{\mathcal{O}}(\boldsymbol{z}) \in \{-1, +1\}$ and $f_{\mathcal{C}}(\boldsymbol{z}, \theta) \in \{-1, +1\}$, then

$$R_{\mathcal{F}} = \frac{1}{2} \int_{\boldsymbol{z} \sim P_Z} |f_{\mathcal{O}}(\boldsymbol{z}) - f_{\mathcal{C}}(\boldsymbol{z}, \theta)| dP_Z. \tag{9}$$

At this point, we introduce the optimal copy model, $f_{\mathcal{C}}^{\infty}$, as shown in Fig. 4. The shaded region corresponds to the copy hypothesis space, which encompasses all the possible copy models of the same family. The optimal copy model $f_{\mathcal{C}}^{\infty}$ is that which is closest to the original $f_{\mathcal{O}}$. As opposed to the copy $f_{\mathcal{C}}$, which is built on a finite set of N synthetic data points, the optimal copy model assumes infinite synthetic samples. Notably, when the original belongs to the copy hypothesis space, then $f_{\mathcal{O}} = f_{\mathcal{C}}^{\infty}$. We use the optimal copy model, $f_{\mathcal{C}}^{\infty}$, to expand (9) as

$$\begin{aligned}
R_{\mathcal{F}} &= \frac{1}{2} \int_{\boldsymbol{z} \sim P_Z} \Big| f_{\mathcal{O}}(\boldsymbol{z}) + f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) - f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) - \\
&\qquad f_{\mathcal{C}}(\boldsymbol{z}, \theta) \Big| dP_Z \\
&= \frac{1}{4} \int_{\boldsymbol{z} \sim P_Z} \Big(f_{\mathcal{O}}(\boldsymbol{z}) - f_{\mathcal{C}}^{\infty}(z) + f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) - \\
&\qquad f_{\mathcal{C}}(\boldsymbol{z}, \theta) \Big)^2 dP_Z \\
&= \frac{1}{4} \int_{\boldsymbol{z} \sim P_Z} \Big(f_{\mathcal{O}}(\boldsymbol{z}) - f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) \Big)^2 dP_Z + \qquad (10) \\
&\quad \frac{1}{4} \int_{\boldsymbol{z} \sim P_Z} \Big(f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) - f_{\mathcal{C}}(\boldsymbol{z}, \theta) \Big)^2 dP_Z + \qquad (11) \\
&\quad \frac{1}{2} \int_{\boldsymbol{z} \sim P_Z} \Big(f_{\mathcal{O}}(\boldsymbol{z}) - f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) \Big) \Big(f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) - \\
&\qquad f_{\mathcal{C}}(\boldsymbol{z}, \theta) \Big) dP_Z \qquad (12)
\end{aligned}$$

where we drop the square root in the first expression, thanks to the codomain of the functions involved.

We identify three different contributions to the fidelity error. The first term appears in (10) and corresponds to the error we incur when replacing the original model with the optimal copy model, *i.e.* the model in the copy hypothesis space which is closest to the original model. This error quantifies the capacity mismatch between the copy hypothesis class and the original. In what follows, we refer to this mismatch as the *capacity error*, $R_{\mathcal{C}}$, defined as

$$R_{\mathcal{C}} = \frac{1}{2} \int_{\boldsymbol{z} \in P_Z} \Big| f_{\mathcal{O}}(\boldsymbol{z}) - f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) \Big| dP_Z.$$

The second error contribution corresponds to (11) and arises from the the fact that we are limited to a finite number of synthetic samples. For a given copy hypothesis space, this error measures the mismatch between the decision boundary output by the optimal copy model and that obtained when copying on N samples. We call this source of error *coverage error*, $R_{\mathcal{CV}}$, and write it as

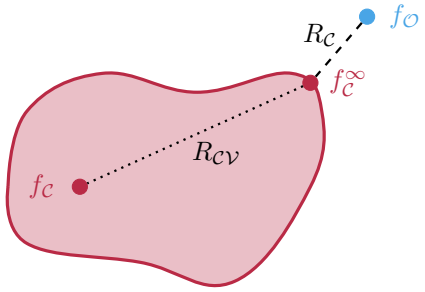


Figure 4: Original, copy and optimal copy models in relation to the copy hypothesis space, together with capacity and coverage errors.

$$R_{\mathcal{CV}} = \frac{1}{2} \int_{\boldsymbol{z} \in P_Z} \Big| f_{\mathcal{C}}^{\infty}(\boldsymbol{z}) - f_{\mathcal{C}}(\boldsymbol{z}, \theta) \Big| dP_Z.$$

Finally, a third error term appears in (12) that accounts for a certain coupling between the capacity and coverage error. We name it the *interaction error*, $R_{\mathcal{I}}$, and it has the form

$$R_{\mathcal{I}} = \frac{1}{2} \int_{\boldsymbol{z} \in P_Z} \Big(f_{\mathcal{O}}(\boldsymbol{z}) - f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) \Big) \Big(f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta) - f_{\mathcal{C}}(\boldsymbol{z}, \theta) \Big) dP_Z.$$

The interaction error as above defined has three main properties. First, the only cases in which it is non-zero is when $f_{\mathcal{O}}(\boldsymbol{z})$ agrees with $f_{\mathcal{C}}(\boldsymbol{z}, \theta)$, while both differ from $f_{\mathcal{C}}^{\infty}(\boldsymbol{z}, \theta)$. Whenever this happens, the net effect of the interaction error is negative, meaning that it acts in the direction of decreasing the total error. This means that the interaction error is only relevant when we select a copy hypothesis very distant from the family of models the original belongs to and at the same time we conduct a poor synthetic sample generation. Under these circumstances, the interaction term corrects errors due to a mismatch in capacity when the coverage error is also high. Second, as the number of synthetic samples increases, $N \to \infty$, the copy approaches the optimal model, $f_{\mathcal{C}} \to f_{\mathcal{C}}^{\infty}$. This reduces the coverage error contribution and, consequently, also the value of the interaction term. Finally, note that we would generally choose a large capacity copy, so that either $f_{\mathcal{O}}$ is in the copy hypothesis space or $f_{\mathcal{C}}^{\infty}$ and $f_{\mathcal{O}}$ are close. As a result, the capacity and interaction terms are reduced and the largest contribution to the error is the coverage term.

In summary, we can express the fidelity error in terms of the capacity, coverage and interaction errors as

$$R_{\mathcal{F}} = R_{\mathcal{C}} + R_{\mathcal{CV}} + R_{\mathcal{I}}.$$

5.2 The Single-Pass Copy

Copying involves the joint optimization of the synthetic data $\boldsymbol{Z}$ and the copy parameters θ through (8), or its simplified form (7). Conducting a simultaneous optimization would require the copy hypothesis space to have certain properties, such as online updating among others. Thus, for the sake of simplicity, in the rest of this article we consider the simplest approach to solving the copying problem: *single-pass copy*.

We cast the simultaneous optimization problem into one where only a single iteration of an alternating projection optimization scheme is used. This effectively splits the problem in two independent sub-problems, finding $\boldsymbol{Z}^*$ and then optimizing for θ^*. The single-pass works as follows:

- **Step 1: Synthetic sample generation**. Observe that the error induced by the sampling $\boldsymbol{Z}$ only appears in the coverage error. Thus, the first step for a simple copy is to find $\boldsymbol{Z}^*$ such that the coverage error, $R_{\mathcal{CV}}$, is minimal. That is,

$$\boldsymbol{Z}^* = \arg\min_{\boldsymbol{Z}} R_{\mathcal{CV}}$$

- **Step 2: Finding the optimized model**. Given a set of synthetic data pairs $\boldsymbol{Z}^*$, find θ^* such that

$$\begin{aligned} &\underset{\theta}{\text{minimize}} \quad \Omega(\theta) \\ &\text{subject to} \quad \|R^{P_Z}_{emp}(f_{\mathcal{C}}, f_{\mathcal{O}}) - R^{P_Z}_{emp}(f^{\dagger}_{\mathcal{C}}, f_{\mathcal{O}})\| < \epsilon, \end{aligned}$$

  or its simplified version (7), provided the adequate conditions hold.

5.3 Synthetic sample generation process

Here, we assume that allowing a sufficiently large synthetic set ensures a good coverage of the whole data space and provide a set of guidelines on how to generate such a set. We define the optimal set of synthetic data points as that for which the coverage error is minimal with respect to the chosen copy hypothesis,

$$\boldsymbol{Z}^* = \arg\min_{\boldsymbol{Z}} \int_{\boldsymbol{z}\sim P_Z} \Big| f^{\infty}_{\mathcal{C}}(\boldsymbol{z},\theta) - f_{\mathcal{C}}(\boldsymbol{z},\theta) \Big| dP_Z.$$

Again, we need to define an empirical approximation to the former equation, so that in practice we minimize the empirical coverage error

$$\boldsymbol{Z}^* = \arg\min_{\boldsymbol{Z}} \sum_{j=1}^{N} \Big| f^{\infty}_{\mathcal{C}}(\boldsymbol{z}_j,\theta) - f_{\mathcal{C}}(\boldsymbol{z}_j,\theta) \Big|, \qquad (13)$$

for $\boldsymbol{z}_j$ a sample from the finite synthetic set $\boldsymbol{Z}$ and $N = |\boldsymbol{Z}|$ the total number of samples. Independently of how effectively and efficiently the copy model capitalizes the samples in $\boldsymbol{Z}$, in order to minimize the coverage error we need a good evaluation through (13). To this end, we deduce a set of considerations that must hold for any synthetic dataset generator under this framework.

For the sake of our next discussion, let us once again consider a binary classification problem and let $f^{\infty}_{\mathcal{C}}(x) \in \{-1,+1\}$ and $f_{\mathcal{C}}(x) \in \{-1,+1\}$ stand for the optimal copy model and copy decision functions, respectively. For this case, the coverage error is given by

$$\begin{aligned} R_{\mathcal{CV}} &= \frac{1}{2}\int_{\boldsymbol{z}\sim P_Z} \Big| f^{\infty}_{\mathcal{C}}(\boldsymbol{z},\theta) - f_{\mathcal{C}}(\boldsymbol{z},\theta) \Big| dP_Z \\ &= \frac{1}{2}\int_{\boldsymbol{z}\sim P_Z} \Big| f^{\infty}_{\mathcal{C}}(\boldsymbol{z},\theta) \Big| \Big| 1 - \frac{f_{\mathcal{C}}(\boldsymbol{z},\theta)}{f^{\infty}_{\mathcal{C}}(\boldsymbol{z})} \Big| dP_Z \\ &= \frac{1}{2}\int_{\boldsymbol{z}\sim P_Z} \Big(1 - \frac{f_{\mathcal{C}}(\boldsymbol{z},\theta)}{f^{\infty}_{\mathcal{C}}(\boldsymbol{z},\theta)} \Big) dP_Z \\ &= \frac{1}{2}\int_{\boldsymbol{z}\sim P_Z} dP_Z - \frac{1}{2}\int_{\boldsymbol{z}\sim P_Z} \frac{f_{\mathcal{C}}(\boldsymbol{z},\theta)}{f^{\infty}_{\mathcal{C}}(\boldsymbol{z},\theta)} dP_Z \\ &= \frac{1}{2} - \frac{1}{2}\int_{\boldsymbol{z}\sim P_Z} f_{\mathcal{C}}(\boldsymbol{z},\theta) f^{\infty}_{\mathcal{C}}(\boldsymbol{z},\theta) dP_Z. \end{aligned}$$

Let us now define a partition of the space such that $\mathcal{D} = \mathcal{D}_+ \cup \mathcal{D}_-$ and $\mathcal{D}_+ \cap \mathcal{D}_- = \emptyset$ for $\mathcal{D}_+ = \{\boldsymbol{z} | \boldsymbol{z} \in \mathcal{D}, f^{\infty}_{\mathcal{C}}(\boldsymbol{z}) = 1\}$ and $\mathcal{D}_- = \{\boldsymbol{z} | \boldsymbol{z} \in \mathcal{D}, f^{\infty}_{\mathcal{C}}(\boldsymbol{z}) = -1\}$ are the two sub-spaces defined by the original decision function. We rewrite the equation above in terms of this partition as

$$R_{\mathcal{CV}} = \frac{1}{2} - \frac{1}{2}\int_{\boldsymbol{z}\in\mathcal{D}_+} f_{\mathcal{C}}(\boldsymbol{z},\theta) dP_Z + \frac{1}{2}\int_{\boldsymbol{z}\in\mathcal{D}_-} f_{\mathcal{C}}(\boldsymbol{z},\theta) dP_Z.$$

Because the partial domains $\mathcal{D}_+$ and $\mathcal{D}_-$ are unknown, we cannot compute this expression directly. Instead, we approximate it using Monte Carlo integration. We define a synthetic data generator that draws random samples $\boldsymbol{z}_j$ from the distribution P_Z and labels them according to the predictions $y_j = f^{\infty}_{\mathcal{C}}(\boldsymbol{z}_j,\theta)$ of the optimal model to obtain a set of data pairs $\boldsymbol{Z} = \{(\boldsymbol{z}_j, y_j)\}$ for $j = 1, ..., N$.

In the general case, the optimal model $f^{\infty}_{\mathcal{C}}$ is unknown to us. To overcome this limitation, we assume that the copy hypothesis space contains the original model decision boundary, so that $f^{\infty}_{\mathcal{C}}(\boldsymbol{z},\theta) = f_{\mathcal{O}}(\boldsymbol{z})$. On this basis, we can replace $f^{\infty}_{\mathcal{C}}(\boldsymbol{z},\theta)$ by $f_{\mathcal{O}}(\boldsymbol{z})$ and query the original model instead to obtain the set of labeled synthetic pairs. Note that this assumption does not always hold. As an extreme case, consider approximating a deep artificial neural network with a linear model. This assumption has important practical consequences that will be discussed at the end of this subsection. For now, let us suppose that it does hold. We can then approximate the coverage error above in terms of the synthetic dataset obtained by sampling the original model according to some distribution P_Z, as

$$R_{\mathcal{CV}} \approx \frac{1}{2} - \frac{1}{2N_+}\sum_{j=1}^{N_+} f_{\mathcal{C}}(\boldsymbol{z}_j, \theta) + \frac{1}{2N_-}\sum_{j=1}^{N_-} f_{\mathcal{C}}(\boldsymbol{z}_j, \theta),$$

for N_+ and N_- the number of samples lying in $\mathcal{D}_+$ and $\mathcal{D}_-$, respectively. Note that we use the same distribution for approximating both integrals.

We define the probability of a sample lying in $\mathcal{D}_+$ as $p_+ = \mathbb{P}(\boldsymbol{z} \in \mathcal{D}_+)$ and the probability of a sample lying in $\mathcal{D}_-$ as $p_- = \mathbb{P}(\boldsymbol{z} \in \mathcal{D}_-)$. These two probabilities depend on the *size* of the positive and negative domains. In particular,

$$p_+ = \int_{\boldsymbol{z}\in\mathcal{D}_+} P_Z(\boldsymbol{z})dz, \quad p_- = \int_{\boldsymbol{z}\in\mathcal{D}_-} P_Z(\boldsymbol{z})dz.$$

With these quantities, we can see that $N_+ = Np_+$ and $N_- = Np_-$. Thus,

$$R_{\mathcal{CV}} \approx \frac{1}{2} - \frac{1}{2Np_+}\sum_{j=1}^{Np_+} f_{\mathcal{C}}(\boldsymbol{z}_j, \theta) + \frac{1}{2Np_-}\sum_{j=1}^{Np_-} f_{\mathcal{C}}(\boldsymbol{z}_j, \theta).$$

The optimization process explicitly depends on the probability distribution P_Z. In the simplest case, we can assume this distribution to be flat on the domain $\mathcal{D}$, so that $z \sim \mathcal{U}(\mathcal{D})$. Under this assumption, p_+ and p_- explicitly correspond to the fraction of volume for each of the classes. Recalling the form of the standard error for the Monte Carlo estimator under this distribution, we can express the standard error for $R_{\mathcal{CV}}$ as

$$\sigma(R_{\mathcal{CV}}) \propto \mathcal{O}\Big(\frac{1}{\sqrt{Np_+}} + \frac{1}{\sqrt{Np_-}}\Big).$$

We can now discuss four key notions to be considered during the synthetic sample generation process. First, we confirm the need to define a domain $\mathcal{D}$ for the synthetic generator. This is a reasonable assumption, since we need to have an approximate idea of the dynamic range of all attributes to build meaningful queries. In what follows, we assume this information to be known to us.

Second, the coverage error depends, among other things, on the fraction of volume occupied by each of the original decision regions in $\mathcal{D}$. If the spatial support of one of the regions is very small with respect to the whole volume, it may be difficult to have a meaningful number of samples on that region, which may lead to large approximation errors. This compels us to introduce the notion of *volume unbalance*, which describes the spatial support for each class label. We take this notion into account to ensure a balanced representation of all class labels in the synthetic dataset. Note that volume unbalance is unrelated to class distribution: a class with large data support may occupy a very small volume.

Third, the volume unbalance effect can be alleviated by a good choice of the sampling distribution P_Z. For example, we can try to infer a sampling distribution that allocates a large amount of the probability mass close to the unknown boundaries. Or, alternatively, we can use heuristics that balance exploration of the sampling domain with some exploitation of the already known areas or boundaries.

Finally, we revisit our former assumption that the original model belongs to the copy hypothesis space. To minimize the effect of volume unbalance, we need to consider the different decision regions learned by the original. If the assumption does not hold, there is a mismatch between the decision boundaries achievable by the optimal copy model and the original. As a consequence, when sampling the original instead of the optimal copy model, we cannot guarantee that the provided information can be efficiently exploited by the copy. In other words, a synthetic dataset obtained using the original model may be good for certain copy hypotheses but less useful for others. Consider for example a non-linear original decision boundary and a linear copy model. Fitting the linear model may not benefit from exploring the twists of the original decision boundary during the synthetic sample generation process. Thus, during this process we should consider the properties and assumptions of the copy hypothesis space to effectively exploit each generated sample.

6 Performance metrics in practice

Ideally, when evaluating copies, we would want to answer questions of the form: *"what does the performance on the synthetic dataset tell us about the goodness of the copy?"*, *"is a perfect accuracy on a synthetic validation set an optimal indicator of a good copy?"*, *"does the copy have enough capacity to replicate the original decision function?"* or *"assuming we have access to the original training dataset and model, what metrics should we use to evaluate the copy?"*. This section is devoted to answering such questions. We divide our discussion into two different parts. First, we propose a measurable approximation to the fidelity error. Second, we propose a set of performance metrics that serve as sanity checks when the original accuracy or the original dataset or both are accessible.

6.1 Empirical fidelity error

All the error of the copy is captured by the fidelity error, as defined in *Section 5*. Nevertheless, in practice, we do not know the exact form of the decision functions $f_{\mathcal{O}}$ and $f_{\mathcal{C}}$, nor do we have access to the domain $\mathcal{D}$. Hence, we cannot directly compute the fidelity error. Alternatively, we use an approximation of the fidelity error over a certain evaluation set. Given a set $\boldsymbol{X} = \{\boldsymbol{x}_l\}$ for $l = 1, ..., |\boldsymbol{X}|$, we define the *empirical fidelity error* with respect to that set as

$$R_F^{\boldsymbol{X}} = \frac{1}{|\boldsymbol{X}|}\sum_{l=1}^{|\boldsymbol{X}|}\mathbb{I}[f_{\mathcal{O}}(\boldsymbol{x}_l) \neq f_{\mathcal{C}}(\boldsymbol{x}_l)],$$

There are different choices for the evaluation set $\boldsymbol{X}$, that may lead to different values of the empirical fidelity error. In particular, we could use the synthetic dataset $\boldsymbol{Z}$ and define the empirical fidelity error $\mathcal{R}_F^{\boldsymbol{Z}}$ with respect to this set

$$R_F^{\boldsymbol{Z}} = \frac{1}{N}\sum_{j=1}^{N}\mathbb{I}[f_{\mathcal{O}}(\boldsymbol{z}_j) \neq f_{\mathcal{C}}(\boldsymbol{z}_j)]. \tag{14}$$

Here, because of the Monte Carlo integration, we incur in a new approximation error. As a result, a low $R_F^{\boldsymbol{Z}}$ is no guarantee of a good copy. For $R_F^{\boldsymbol{Z}}$ to be a valid assessment of the total error, the synthetic dataset must be large enough and the volume unbalance effect needs to be controlled for. If these conditions are met and the $R_F^{\boldsymbol{Z}}$ is still different from zero, the theoretical sources of error should be considered. The error either arises from a low capacity of the copy (large capacity error) or because there is a mismatch between the optimal achievable model and the one obtained (large coverage error). In the first case, we should change the model to a larger capacity option. In the second, more samples or better quality samples may be needed.

6.2 Copy accuracy and copy accuracy estimation

When the original dataset $\boldsymbol{S}$ is available, we can define additional metrics that help us better characterize the behavior of the copy. We define *copy accuracy*, $\mathcal{A}_{\mathcal{C}}$ as the generalization performance of the copy in the original data environment. We express it in terms of the accuracy when predicting original test set pairs $\{(t_i, s_i)\}$, so that

$$\mathcal{A}_{\mathcal{C}} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{I}[t_i = f_{\mathcal{C}}(s_i)], \tag{15}$$

for $\mathbb{I}$ the indicator function. Additionally, we can also compute the empirical fidelity error over the original dataset as

$$R_F^{S} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{I}[f_{\mathcal{O}}(s_i) \neq f_{\mathcal{C}}(s_i)] \tag{16}$$

We remark that R_F^S and R_F^Z are usually very different values. The difference corresponds to the mismatch between the probability density functions of the original dataset, $P_S(s)$, and the synthetic sample generator, $P_Z(z)$.

When the performance of the original model is known, we refer to this value as the *original accuracy*, $\mathcal{A}_{\mathcal{O}}$. We take this to be an upper bound to the performance the copy can achieve in the original data environment. That is, an upper bound to the copy accuracy. Indeed, since we assume all the loss to be captured by the fidelity error, we can write $\mathcal{A}_{\mathcal{C}}$ in terms of the original accuracy as

$$A_{\mathcal{C}} = A_{\mathcal{O}}(1 - R_{\mathcal{F}}).$$

We approximate the theoretical value $R_{\mathcal{F}}$ with the empirical fidelity error with respect to the synthetic set, $\mathcal{R}_F^{\boldsymbol{Z}}$, and introduce the *estimated copy accuracy*, $\widehat{A_{\mathcal{C}}}$, as

$$\widehat{A_{\mathcal{C}}} \quad \approx \quad A_{\mathcal{O}}(1 - R_{\mathcal{F}}^{Z}). \tag{17}$$

Thus, in cases where the original accuracy is known but the original dataset is not, we can exploit the above expression to provide an estimation of the copy accuracy.

7 Experiments

The experiments are divided as follows. First, we validate copies on 60 datasets from UCI Machine Learning Repository database [56], including both binary and multiclass classification problems. We then present the reader with two financial use cases. On the one hand, we derive regulatory-compliant high-performing copies for non-client mortgage loan default prediction in a private dataset from BBVA. On the other hand, we apply the copying process to a dataset containing information on individuals applying for a loan from Lending Club website [57]. Finally, we study the application of copies to obtain a fair classification of alignment in the superheroes dataset from SuperHeroDb [58].

7.1 Experimental setup

We convert nominal attributes to numerical and re-scale all variables to zero-mean and unit variance. Original data are split into stratified 80/20 training and test sets. Copies are built with synthetic datasets of size $1e6$, generated using a naïve balanced generator (see the Appendix). We build copies using no cross-validation or hyper-parameter tuning. We let decision trees grow until each leaf contains a single sample and train neural networks and boosting methods with no regard for generalization. We run each experiment 100 times and evaluate the performance of copies using test sets comprised of $1e6$ synthetic points. For validation purposes, we assume both the original accuracy and the original dataset to be known in all cases. We report averages over all repetitions for the directly measured copy accuracy $\mathcal{A}_{\mathcal{C}}$ and its estimated value $\widehat{\mathcal{A}_{\mathcal{C}}}$. We also report the mean empirical fidelity error measured over both the original and the synthetic test sets.

7.2 UCI classification

In our first experiment, we use 60 datasets from the UCI database. We do so by following [59], who present a comparison of 122 datasets from this source. We refer the

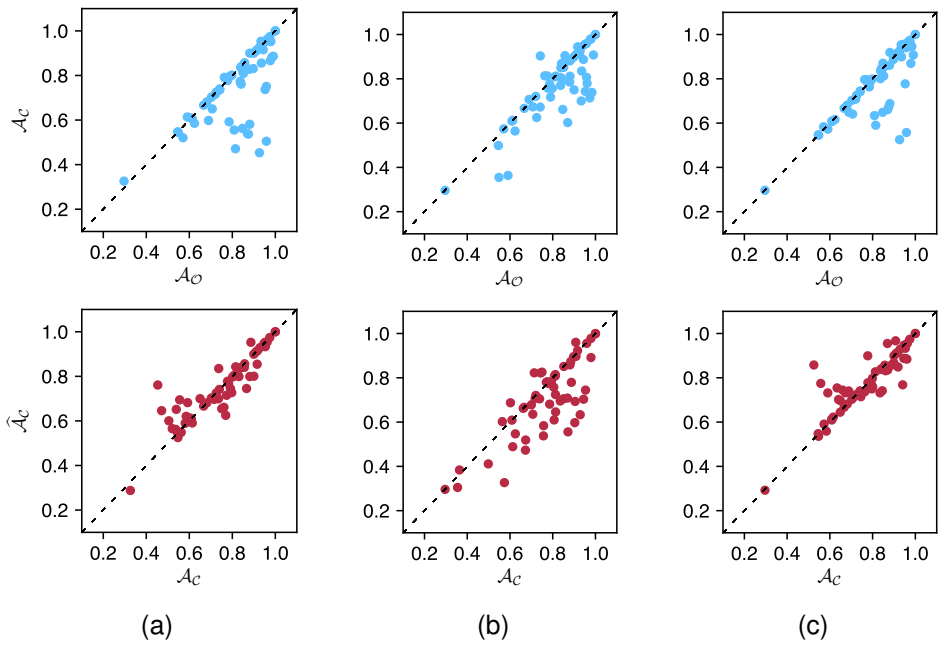


Figure 5: From top to bottom, distribution of average copy accuracy against original accuracy and distribution of average estimated copy accuracy against average true copy accuracy for all datasets and for copies based on (a) decision trees, (b) logistic regression and (c) random forest.

reader to this paper for a specific description of initial data selection and preprocessing. We discard 62 datasets due to several reasons: we do not consider those datasets which contain less than 100 samples and remove those with at least one class label with a frequency smaller than 10% of the total size of the dataset. We additionally impose that the total number of inputs be greater than double the number of attributes. We obtain a total of 60 datasets, among which 42 correspond to binary classification problems and the remaining 18 are multiclass.

7.2.1 Training original models

To learn these 60 datasets, we use 6 state-of-the-art classification algorithms, including adaboost classifier (*adaboost*), an artificial neural network (*ann*), a random forest classifier (*random_forest*), a linear SVM (*linear_svm*), a SVM with a radial basis function kernel (*rbf_svm*) and a gradient-boosted decision tree classifier (*xgboost*). We use standard methods from Python's *scikit-learn* module to train the original model for the first 5 algorithms and *xgboost* library to fit the gradient-boosting trees. We sort all datasets in alphabetical order, group them in sets of 10 and randomly assign a classifier to each group. We do this to avoid any bias regarding the algorithms used for each problem. A full description of the 60 datasets, including general data attributes and their assigned classifier, can be found in Table 1.

To further avoid any interference from our part, we build a generic pipeline for all problems. We perform a standard grid search to find the optimal parameter set for each dataset and classifier pair. We do so using a 3-fold cross-validation. Among the 60 classifiers, three learn a decision function that excludes at least one of the original class labels. This occurs for *pittsburg-bridges-REL-L*, for which only two of the original three classes are learned, and *planning* and *statlog-australian-credit*, for which the resulting decision function learns a single class label for all the data points. Moreover, because we use a fixed pipeline, not all originals yield an optimal performance. This is the case, for example, of *echocardiogram*, where the original accuracy is equal to 0.3. We choose to show this result for two reasons. First, we want to keep the experimental setup as agnostic as possible and hence the random pairing of models and datasets. Second, it introduces an important idea: a copy can only be as good as its target original. Non-optimal originals lead to poorly performing copies. We stress, nonetheless, that in a real setting one would be interested in copying only those models that perform reasonably well.

During synthetic data generation, we identify three cases of volume unbalance: *congressional-voting*, *ilpd-indian-liver* and *statlog-image*. Evenwhile riginal datasets are balanced with respect to class distribution, when randomly sampling the original space we only recover a small fraction of samples for one or more of the class labels. This can lead to sub-optimal results, given that it is often cheaper for the copying algorithm to wrongly classify data points in the subsampled class or classes. Imposing that the synthetic dataset be balanced mitigates this issue to a great extent and ensures that the copy treats all labels equally.

7.2.2 Building copies

We assay different copy model hypotheses to evaluate how the heuristics and form of each copy algorithm impact the process. Fig. 5 shows the distribution of the measured performance metrics for copies based on sklearn's decision tree (*decision_tree*), logistic regression (*logistic_regression*) and random forest (*random_forest*) classifiers. We choose decision trees because they are easily interpretable in most cases, logistic regression because it is a linear model and random forest as an example of a bagging method.

In particular, Fig. 5(a), Fig. 5(b) and Fig. 5(c) show the distribution of the mean copy accuracy $\mathcal{A}_{\mathcal{C}}$ against the original accuracy $\mathcal{A}_{\mathcal{O}}$ and the estimated copy accuracy $\widehat{\mathcal{A}_{\mathcal{C}}}$ for all datasets and copy algorithms. We observe that points for both *decision_tree* and *random_forest* are scattered around the main diagonal, whereas copies based on *logistic_regression* show a greater dispersion. Specially when comparing $\mathcal{A}_{\mathcal{C}}$ to $\widehat{\mathcal{A}_{\mathcal{C}}}$. The value of $\widehat{\mathcal{A}_{\mathcal{C}}}$ tends to underestimate that of $\mathcal{A}_{\mathcal{C}}$. In other words, the empirical fidelity error over the synthetic data overestimates the real error. This is in part due to the difference in the distributions P_S and P_Z. When evaluating $R^{\boldsymbol{Z}}_{\mathcal{F}}$, we measure the performance of the copy in the space defined by P_Z, so that we may penalize the copy for errors in regions where no original data are present.

In Table 1 we summarize the performance metrics for the different problems and copy algorithms. We report $\mathcal{A}_{\mathcal{C}}$, $\widehat{\mathcal{A}_{\mathcal{C}}}$ and empirical fidelity error $R^{\boldsymbol{S}}_{\mathcal{F}}$ computed over the original dataset. Results show the ability of copies to replicate the behavior of originals in most problems. Overall, copy accuracy is competitive for the proposed synthetic dataset size. Moreover, for most cases, the estimated copy accuracy provides a reliable approximation to the accuracy of the copy in real data, while empirical fidelity error yields values close to 0 for the majority of models and datasets, which indicates that the copies are correctly built.

When using a *logistic_regression* to copy higher capacity originals such as *ann* or *xgboost*, we identify several datasets where no degradation is observed when substituting the latter with the former. This is the case, for example, for *breast-cancer-wisc* and *wine*, where $\mathcal{A}_{\mathcal{C}}$ is reasonably close to $\mathcal{A}_{\mathcal{O}}$, even while the copy algorithm can only learn linear relations among attributes. We take this as an indication that the chosen originals are too complex to learn relatively simple decision functions. Copying here allows us to use a simpler model with less parameters and training requirements.

We identify a number of cases where copies built using *decision_tree* or *random_forest* models clearly outperform those obtained using a *logistic_regression*. Notably, this happens for dataset *energy-y1* and *iris*. This difference in performance arises from the fact that the original decision function is not linear[5]. More complex and non-linear classifiers are therefore needed to build well-fitted copies. Note that this is a clear example where the capacity error dominates over the coverage and interaction terms. Moreover, because the chosen copy hypothesis space, the logistic family, is not well-suited to model the original decision function, the approximation that $f^{\infty}_{\mathcal{C}} = f_{\mathcal{O}}$ does not hold, so that we further incur in a large approximation error.

Finally, it is worth mentioning that there are cases where the copy hypothesis space is well chosen, but the measured empirical fidelity error is high. See for the example the cases of *musk_1* and *musk_2*, where originals are based on *linear_svm* and copies are built using *random_forest* and which are both high dimensional problem. We observe that, in both cases, $\mathcal{A}_{\mathcal{C}}$ is notably lower than $\mathcal{A}_{\mathcal{O}}$. This happens when the coverage contribution is substantial. Indeed, in the most complex and intricate datasets we detect that $1e6$ synthetic data points are not enough to ensure a small $R_{\mathcal{CV}}$.

7.3 Risk scoring for non-client mortgage loans

We consider a private dataset of non-client[7] mortgage loan applications recorded during 2015 all over Mexico. The dataset consists of 19 attributes for 1.328 loan applicants as described in [60]. Even though all individuals in the dataset were granted a loan, only 1.025 paid it off. This corresponds to a ratio of defaulted loans of 23% and motivates the needs for more refined credit risk scoring models.

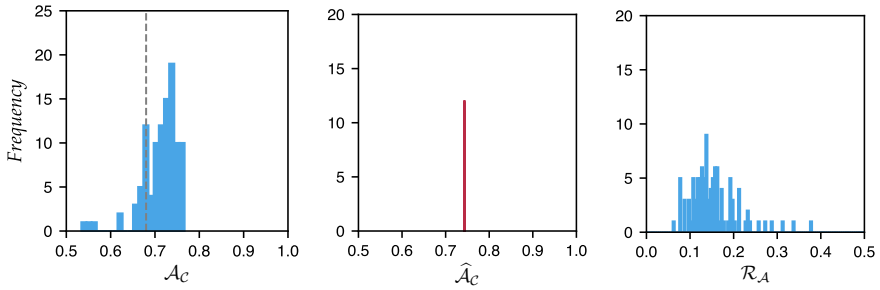
A widely established technique for this type of modelling is logistic regression. Partly because it performs relatively well on credit prediction settings, while at the same time offering the additional advantage of a relative ease of interpretation that complies with regulatory requirements. Even so, models based on logistic regression fail to account for non-linearities in the data, which are usually modelled during an increasingly complex preprocessing step.

During this step, which is critical to maximize business objectives, domain knowledge by experts is exploited to generate a set of highly predictive artificially generated attributes. A qualified risk analyst is required to conduct a tedious process of trial and error until a valid set of predictive variables is found. This incurs in a large economical cost and a delayed time-to-market delivery. Even worse, preprocessing largely reduces the interpretability of the resulting model: new variables often reflect complex relations among original data attributes and therefore remain non-decomposable [27] as far as the regulators are concerned.

[5] Note that even while original training data might be linearly separable, the original decision boundary may be non-linear.

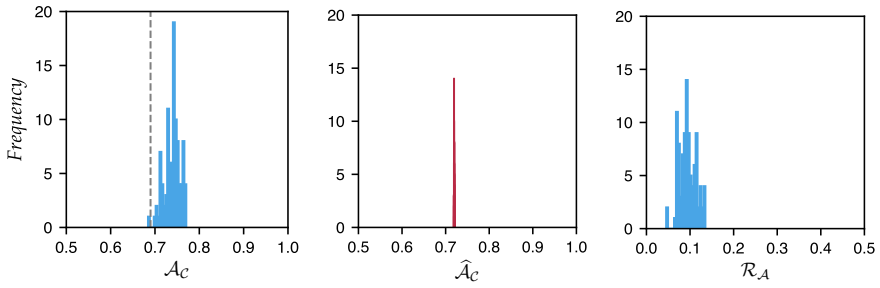
[6] Blank spaces correspond to cases where originals learn a single class label and thus cannot be copied using a logistic regression.

[7] The term non-client refers to those individuals who had no previous contractual relation with the bank at the time of loan application.



(a) (b) (c)

Figure 6: Distribution of values computed for the *scenario 1* for metrics (a) true copy accuracy, (b) estimated copy accuracy and (c) empirical fidelity error over the original dataset. For comparison purposes, the accuracy of a decision tree trained on original data is shown in black in (a).



(a) (b) (c)

Figure 7: Distribution of values computed for the *scenario 2* for metrics (a) true copy accuracy (b) estimated copy accuracy and (c) empirical fidelity error over the original dataset. For comparison purposes, the accuracy of a decision tree trained on original data is shown in black in (a).

In what follows, we present two different scenarios to tackle these issues. In the first, we create a new model that remains interpretable while retaining the prediction performance of the original logistic regression, trained on complex hand-crafted attributes. In the second, we decrease time-to-market delivery by avoiding the preprocessing step and training a high capacity model instead. We then copy this model with a simpler architecture, that is nonetheless compliant with production and regulatory requirements.

7.3.1 Deobfuscated risk scoring models

We emulate a standard production pipeline by preprocessing the data to obtain 6 carefully crafted variables. We use them to train a standard logistic regression that yields an accuracy of 0.77. We copy this whole predictive system, composed of both the preprocessing module and the logistic model, using a decision tree classifier. In Fig. 6 we show the distribution of scores for this experiment. We obtain an averaged copy accuracy of 0.71 ± 0.04 and an estimated copy accuracy of 0.74314 ± 0.00018. The mean empirical fidelity error is 0.03488 ± 0.00018 and the mean empirical fidelity error on the original test set is 0.15 ± 0.05.

Note that while the value of the empirical fidelity error over synthetic data is small, the value computed over the original test set provides evidence that the process can be improved. We argue that if we were to increase the number of synthetic data points, and better explore the boundaries, the approximation error would converge to a more reliable value and the overall error would be further reduced.

The copy takes all 19 features in the original dataset and outputs a prediction that approximates that of the original. It does so without any internal preprocessing of variables, *i.e.* original attributes appear deobfuscated. Thus, the problem of non-decomposability is effectively solved. For validation purposes, we also show in Fig. 6(a) the accuracy achieved by a decision tree classifier when directly trained using the original dataset. We can observe that the predictive performance of the decision tree architecture on the original training samples is smaller than that obtained through the copying process. This shows an additional advantage of using a copy as it might be used to guide a certain model to better solutions in its parameter space.

7.3.2 High-performance regulatory compliant copies

We use all the 19 features in the original dataset to train a gradient-boosted tree. This model achieves an original accuracy of 0.79. We copy it by means of a decision tree classifier and report the results in Fig. 7. The true mean copy accuracy averaged over all runs is 0.74 ± 0.02 and the accuracy estimated using (17) is equal to 0.7194 ± 0.0003. Thus, the average empirical fidelity error is 0.09 ± 0.0003 and the average empirical fidelity error over the original dataset is 0.09 ± 0.02. Note that while final model attributes differ from this application to that of *scenario_1*, original instances are shared in both cases, so as to minimize any bias regarding the specific choice of data.

In this scenario, copies are generated without any preprocessing. Yet the difference in performance between the preprocessed logistic model in *scenario_1* and the copy decision trees is minor when tested against original test data. In Fig. 7(a) we display the accuracy achieved by a decision tree trained directly on the original data. This value is equal to 0.69 ± 0.01. Comparison between this result and the mean true copy accuracy for this problem provides further evidence for the benefits of using copies in this context.

7.4 Online loan default prediction

Here, we assume that the environment where the original operates has changed over time. So much so that the performance of the model has decreased notably, rendering it unfit for future prediction. We also assume the original model type to be unknown. In this circumstance, we show that a copy can be built to display online behavior and recover the original operation point.

For predicting whether a potential borrower will repay a loan, the Lending Club website publishes statistics about individual loan applicants [57]. The complete dataset contains a comprehensive list of features for all loans issued through the 2007-2015 period, including the current loan status, latest payment information, number of finance inquires, borrower's annual income or zip code, among others. We remove null and missing values and drop all fields which provide no useful information for inference. We also identify and drop all features that cause data leakage as those that are typically not available at the time the prediction is output [61]. Finally, we label the dataset by classifying all loans identified as defaulted, charged off or late as *bad*. The resulting database contains information about a total of 50 attributes for 887,379 loans divided into two classes.

We feed these data to an original neural network with denseNet architecture[62] consisting of 5 hidden layers with 256, 128, 64, 32 and 16 neurons, respectively. We use self-normalizing units[63] to avoid internal covariate shift, dropout rate of 10%, and a least squared loss optimized using Adam. Because the original dataset is highly unbalanced, with the *bad* loans accounting only for the 8% of the data, we use balanced batches. We choose our operation point to be that for which the recall value for both classes are closer to each other. We achieve an original accuracy of 0.63 with a recall of 0.59 for *bad* and 0.63 for *good*.

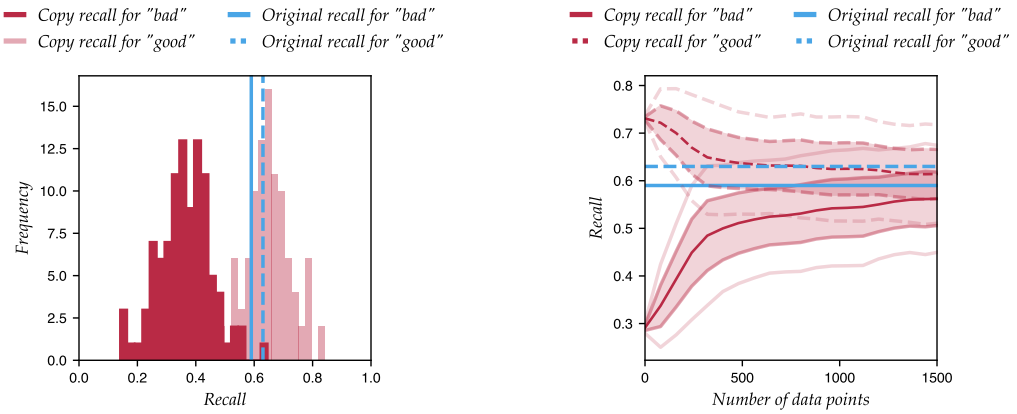
We copy using an artificial neural network, with a much more simple architecture. We use five fully connected layers with 256, 128, 64, 32 and 16 selu neurons, no dropout and a least square loss with a default parameter Adam optimizer. We obtain a mean copy accuracy of 0.63 ± 0.07. The estimated copy accuracy value is equal to 0.603 ± 0.009, corresponding to an empirical fidelity error of 0.042 ± 0.009. Equivalently, the empirical fidelity error over the original dataset is 0.45 ± 0.07. The copy recall distribution over the original dataset is shown in Fig. 8(a) for both classes. Observe that we correctly recover the recall operating points of one of the classes, but suffer a loss of around 20% for the other. Indeed, we build a copy with online capabilities, with the same accuracy as the original and with a reasonably close operating point. Moreover, in the presence of new data points,copies can be fine tuned to achieve a new desirable operating point, as shown in Fig. 8(b). In this particular example, we recover an equal rate of 59% after visiting a few hundred examples of the original training data.

In this experiment, we show how copies can be used to move a trained classifier to an online setting. Moreover, we observe that copies can recover the original operation point. It is worth noting that this particular example also shows the capability of copies to serve as an analysis tool for other models. In particular, we observe that the denseNet architecture and the fully connected neural network both have very similar operation points.

7.5 A fair classification of superhero alignment

For this experiment, we exploit a fictitious example that nonetheless represents a use case common to many real scenarios. We use superheroes dataset [58], which describes characteristics such as demographics, powers, physical attributes and studio of origin of every superhero in SuperHeroDb [64]. In total, the dataset contains information about 177 attributes for 660 superheroes. We use these data to define a binary classification problem choosing superhero alignment as the target attribute. We label as *good* all superheroes marked as so and label as *bad* all other entries, including those labelled as bad, neutral or unknown. That is, we assume every superhero not explicitly labelled as good to be bad. We use the resulting binary classification dataset to train an artificial neural network with two hidden layers of 100 and 300 neurons with *reLu* activation. We use a least squared loss optimized using Adam and no drop-out. We obtain an original accuracy of 0.67.

Among the many attributes of this classification model, there are those such as *gender* and *race*, that may be deemed sensitive. Using these can result in a biased prediction and since be problematic. In this experiment we



(a) (b)

Figure 8: Distribution of (a) recall scores achieved by original and copy and (b) recall scores over number of new training data points for the positive and negative classes.

assume that the original artificial neural network has been trained using *gender* and *race* and that it cannot be modified to correct for any bias. To do so, we build a copy to reproduce the learned decision function, excluding these attributes.

First, we check that no other variable is correlated with *gender* and *race* and can potentially leak this information into the copy. We train different models to predict the *gender* or *race* of each entry using information about all the rest of the variables. In Table 2 we show the accuracies of these models within the different classes. We average 100 runs and obtain a mean balanced accuracy of 0.42 ± 0.08 when predicting *gender* and of 0.28 ± 0.03 when predicting *race*. We conclude that all other attributes are independent from *gender* and *race*, so that we can safely remove them without incurring in any leakage of information.

We generate a synthetic dataset from which we extract the two problematic attributes and use the remaining data to learn a new artificial neural network with the same architecture as that of the original model, *i.e.* two hidden layers consisting of 100 and 300 neurons. The resulting model yields a true copy accuracy of 0.68 ± 0.04, averaged over all runs, while the estimated copy accuracy is equal to 0.661 ± 0.002. The loss in accuracy we incur when substituting the original with the copy in the original data space is notably low. Finally, the mean empirical fidelity error evaluated over the original test data points is 0.18 ± 0.03. Even while this value may seem high, we stress that the removal of two features results in a loss of information from original to copy. Thus, there is a certain shift in the original decision function that leads to a disagreement between both models. Yet, this disagreement does not penalize the performance of the copy. All in all, we reproduce the original decision boundary to a high degree of fidelity and with the added benefit of eliminating sensitive attributes from the copy.

8 Discussion

Here, we focus on general insights learned from the experiments carried on in the previous section. The error contributions that interact throughout the copying process are collectively defined through the fidelity error, or its approximation through empirical fidelity error. When evaluating copies, the condition that empirical fidelity error be small is necessary, but not sufficient. Having significant errors in certain regions and none in others may lead to a low average error, while altogether not ensuring a good performance. The opposite is also true: a large empirical fidelity error does not necessarily lead to a low copy accuracy. Take, for example, errors distributed around the decision function. This may happen, for instance, when trying to copy a smooth decision function using linear decision cuts. If errors are very substantial, this may be seen as a problem. However, if data points in the original dataset are distributed far away from the boundary, errors in the synthetic space would not translate into errors in the original space. No effective loss would therefore be measured when substituting the original with the copy.

To a large extent, copy evaluation depends on the amount of information available. The more knowledge we allow ourselves, the more reliable our estimates will be. Notably, if we assume the original dataset to be accessible, we can obtain a direct estimate of the generalization performance of copies in the original data domain. Furthermore, if the original data were known, we could exploit this information to choose P_Z to be as close to P_S as possible. That is, we could redefine the copy operation space to match the original data distribution. If, additionally, the form of the original model were also known, we could refine the choice of copy hypothesis. In those cases where original and copy both have similar decision boundary shapes, copying is conducted with greater ease. That is, when the original decision function is formed of cuts perpendicular

to the axes, *i.e.* it is a random forest, it is easier to copy with a decision tree than it is with a SVM with a radial basis kernel. Conversely, those originals with smooth decision functions are better copied using classifiers other than trees.

At this stage, we may ask ourselves the question: *if the original data are available why should we copy the original model instead of learning a new classifier?* There exist scenarios in which a new training may not be advisable. For starters, a newly learned model may display very different behavior and decision properties than the original. This is impracticable in production environments where performance has to be preserved and controlled. Another reason to use a copy instead of learning a new model is not having to take care of the overfitting effect. As shown in *Section 4.3*, when copying we can avoid the hyper-parameter optimization step. In general, copies can be understood as a tool to bridge the gap between accuracy and any other desired property. In particular, copies help in breaking the trade-offs that we face in training high-performance models in contexts where other characteristics such as interpretability, simplicity or compliance are required.

9 Conclusions and future work

In this paper we propose and validate a model-agnostic framework to copy machine learning classifiers. Copies provide a reliable solution to many open issues in machine learning deployment. They allow us to enhance classifiers with new features to adapt them to new requirements. We derive the theory behind copying and discuss the implications of building copies in practice. We identify the different sources of loss and provide guidelines on how best to generate synthetic sets representative of the original decision function. We also introduce a set of metrics to evaluate copies in practice, assuming access to different levels of information. Our experiments demonstrate that this approach is feasible and that copies can be used to substitute originals in real data environments. Moreover, the case studies presented show the potential of copies to approach open issues such as those related to interpretability, fairness or productivization of machine learning models.

The problem of representing the decision behavior of a trained machine learning model using a finite number of samples is far from being solved. While the approach here presented has the advantage of simplicity, the form of the synthetic sample generator itself is in need of refinement. In particular, an in-depth study should be conducted to evaluate different methods to sample closed domains where class distribution is governed by an unknown decision function. Much research also remains to be done on how to solve the joint optimization problem. While the single pass-copy provides a reasonable approximation, more global approaches should be studied.

In this work we restrict ourselves to exploring the application of copies to the areas of interpretability, fairness and general enhancement. Nonetheless, There exist other fields were copies are potentially useful. Particularly that of privacy, where copies be specifically built to be privacy-preserving with respect to the original data attributes and instances.

Acknowledgements

This work has been partially funded by the Spanish project TIN2016-74946-P (MINECO/FEDER, UE), and by AGAUR of the Generalitat de Catalunya through the Industrial PhD grant 2017-DI-25. We gratefully acknowledge the support of BBVA Data & Analytics for sponsoring the Industrial PhD.

References

[1] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proc. of Conf. on Computer and Communications Security (SIGSAC)*, 2015.

[2] Reza Shokri, Cornell Tech, Marco Stronati, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 3–18, 2017.

[3] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine Learning Models that Remember Too Much. In *Proc. of Conf. on Computer and Communications Security (SIGSAC)*, pages 587–601, 2017.

[4] European Union Commission. Legislation. *Official Journal of the European Union*, 2016.

[5] Bryce Goodman and Seth Flaxman. European Union Regulations on Algorithmic Decision-Making and a Right to Explanation. *AI Magazine*, 38(3):50–57, 2017.

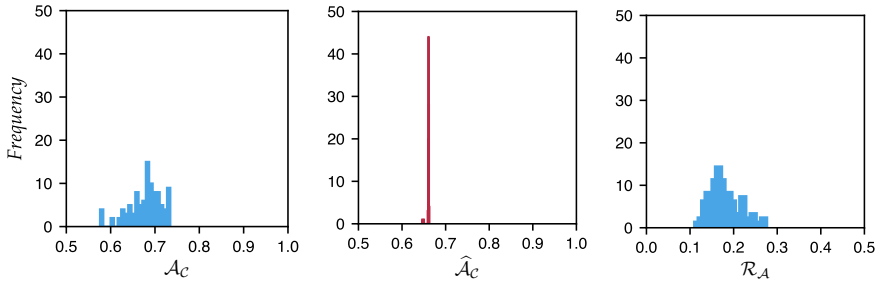
[6] Andrew D Selbst and Julia Powles. Meaningful Information and the Right to Explanation. *Int. Data Privacy Law*, 7(4):233–242, 2017.

[7] Solon Barocas and Andrew D Selbst. Big Data's Disparate Impact. *California Law Review*, 104(671):671–732, 2016.

[8] Batya Friedman and Helen Nissenbaum. Bias in Computer Systems. *ACM Trans. on Information Systems*, 14(3):330–347, 1996.

[9] Moritz Hardt. How Big Data is Unfair. *Medium*, 2014.

[10] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden Technical Debt in Machine Learning Systems. In *Proc. of Conf. on Neural Information Processing Systems (NIPS)*, pages 2503–2511, 2015.



(a) (b) (c)

Figure 9: Distribution of values computed for the *superheroes* experiment for metrics (a) true copy accuracy (b) estimated copy accuracy and (c) empirical fidelity error over the original dataset.

[11] Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and Critique of Techniques for Extracting Rules from Trained ANNs. *Knowledge-Based Systems*, 8(6):373–389, 1995.

[12] Mark W. Craven and Jude W. Shavlik. Learning Symbolic Rules Using Artificial Neural Networks. In *Proc. of Int. Conf. on Machine Learning (ICML)*, pages 73–80, 1993.

[13] L.M. Fu. Rule Learning by Searching on Adapted Nets. In *Proc. of the 9th Nat. Conf. on Artificial Intelligence*, pages 590–595, 1991.

[14] Sebastian Thrun. Extracting Rules from Artificial Neural Networks with Distributed Representations. *Advances in Neural Information Processing Systems*, 7, 1995.

[15] M. W. Craven and J. W. Shavlik. Extracting Tree-structured Representations of Trained Networks. In *Proc. of Conf. on Neural Information Processing Systems (NIPS)*, pages 24–30, 1995.

[16] Guido Bologna and Yoichi Hayashi. A Comparison Study on Rule Extraction from Neural Network Ensembles, Boosted Shallow Trees, and SVMs. *Applied Computational Intelligence and Soft Computing*, 2018:1–20, 2018.

[17] Xinchuan Zeng and Tony R. Martinez. Using a Neural Network to Approximate an Ensemble of Classifiers. *Neural Processing Letters*, 2000.

[18] Christian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model Compression. In *Proc. of ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pages 535–541, 2006.

[19] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2017.

[20] Nilesh Dalvi, Pedro Domingos, Mausam Sumit, and Sanghai Deepak Verma. Adversarial Classification. In *Proc. of ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2004.

[21] Daniel Lowd and Christopher Meek. Adversarial Learning. In *Proc. of ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pages 641–647, 2005.

[22] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *Proc. of the 25th USENIX Security Symposium*, pages 601–618, 2016.

[23] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *Tech. Report. CoRR*, 2016.

[24] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *Proc. of ACM Asia Conf. on Computer and Communications Security*, pages 506–519, 2017.

[25] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2014.

[26] Been Doshi-Velez, Finale; Kim. Towards a rigorous science of interpretable machine learning. In *eprint arXiv:1702.08608*, 2017.

[27] Zachary C. Lipton. The Mythos of Model Interpretability. In *ICML Workshop on Human Interpretability in Machine Learning*, pages 96–100, 2016.

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *Proc. of ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1135–1144, 2016.

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-Precision Model-Agnostic

Explanations. In *Proc. of Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[30] Ilias Flaounas. Beyond the technical challenges for deploying Machine Learning solutions in a software company. In *ICML Human in the Loop Machine Learning Workshop*, 2017.

[31] Alfred Spector, Peter Norvig, and Slav Petrov. Google's Hybrid Approach to Research. *Communications of the ACM*, 55(7), 2012.

[32] Alice Zheng and Sethu Raman. The Challenges of Bringing Machine Learning to the Masses. In *NIPS Workshop on Software Engineering for Machine Learning*, 2014.

[33] Deploy TensorFlow. `https://www.tensorflow.org/deploy/`.

[34] Wai Chee Yau. How Zendesk Serves TensorFlow Models in Production. *Medium*, 2017.

[35] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine Bias: There's Software Used Across the Country to Predict Future Criminals. And It's Biased Against Blacks. *ProPublica*, 2016.

[36] Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification *. *Proc. of Machine Learning Research*, 81:1–15, 2018.

[37] Brendan F. Klare, Mark J. Burge, Joshua C. Klontz, Richard W. Vorder Bruegge, and Anil K. Jain. Face Recognition Performance: Role of Demographic Information. *IEEE Trans. on Information Forensics and Security*, 7(6):1789–1801, 12 2012.

[38] Alice B Popejoy and Stephanie M Fullerton. Genomics is Failing on Diversity. *Nature*, 538:161–164, 2016.

[39] Tolga Bolukbasi, Kai Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Proc. of Conf. on Neural Information Processing Systems (NIPS)*, pages 4356–4364, 2016.

[40] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics Derived Automatically from Language Corpora Contain Human-like Biases. *Science*, 356(6334):183–186, 2017.

[41] Saikat Guha, Bin Cheng, and Paul Francis. Challenges in Measuring Online Advertising Systems. In *Proc. of ACM Int. Conf. on Data Communications (SIGCOMM)*, pages 81–87, 2010.

[42] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness Through Awareness. In *Proc. of the 3rd Innovations in Theoretical Computer Science Conf.*, pages 214–226, 2012.

[43] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of Opportunity in Supervised Learning. In *Proc. of Conf. on Neural Information Processing Systems (NIPS)*, pages 3323–3331, 2016.

[44] Leon Bottou and Yann Le Cun. Large Scale Online Learning. In *Proc. of Conf. in Neural Information Processing Systems (NIPS)*, pages 217–234, 2004.

[45] Sergio Escalera, David Masip, Eloi Puertas, Petia Radeva, and Oriol Pujol. Online error-correcting output codes. *Pattern Recognition Letters*, 32:458–467, 2009.

[46] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[47] Vladimir N. Vapnik. Principles of Risk Minimization for Learning Theory. In *Proc. of Conf. in Neural Information Processing Systems (NIPS)*, pages 831–838, 1992.

[48] Vladimir N. Vapnik and Alexey Chervonenkis. *Theory of Pattern Recognition (in Russian)*. Nauka, 1974.

[49] F. Cucker and S. Smale. On the Mathematical Foundations of Learning. *Bulletin of the Amer. Math. Soc.*, 39(1):1–49, 2002.

[50] S. Mendelson. A Few Notes on Statistical Learning Theory. In *Advanced Lectures on Machine Learning. Lecture Notes in Computer Science*, volume 2600. Springer, 2003.

[51] David A. McAllester. Pac-bayesian model averaging. In *Proc. of Int. Conf. on Computational Learning Theory (COLT)*, pages 164–170, 1999.

[52] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger Generalization Bounds for Deep Nets via a Compression Approach. In *Proc. of Int. Conf. on Machine Learning (ICML)*, 2018.

[53] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the Margin: a New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[54] Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of Optimization and Implicit Regularization in Deep Learning. In *eprint arXiv:1705.03071*, 2017.

[55] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. SGD Learns Over-parameterized Networks that Provably Generalize on Linearly Separable Data. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2017.

[56] Dua Dheeru and Efi Karra Taniskidou. UCI Machine Learning Repository, 2017.

[57] Lending Club Loan Data, Kaggle. `https://www.kaggle.com/wendykan/lending-club-loan-data`.

[58] Super Heroes Dataset, Kaggle. `https://www.kaggle.com/claudiodavi/superhero-set`.

[59] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim, and Dinani Amorim

Fernández-Delgado. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014.

[60] Irene Unceta, Jordi Nin, and Oriol Pujol. Towards Global Explanations for Credit Risk Scoring. In *eprint arXiv:1811.07698*, 2018.

[61] Anahita Namvar, Mohammad Siami, Fethi Rabhi, and Mohsen Naderpour. Credit Risk Prediction in an Imbalanced Social Lending Environment. *Int. Journal of Computational Intelligence Systems*, 11(1), 2018.

[62] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proc.of the IEEE Conf. on Pattern Recognition and Computer Vision (CVPR)*, 2017.

[63] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks. In *Proc. of Conf. on Neural Information Processing Systems (NIPS)*, 2017.

[64] Superhero Database. `https://www.superherodb.com/`.

A Synthetic data generation in practice

We propose a set of routines that provide specific information about the original decision function and derive a protocol that specifies the form of each of such routines. Further, we exploit this protocol to obtain a balanced synthetic dataset following a naïve approach following the conditions listed in *Section 5.3*.

Algorithm 1 NaïveBalancedGenerator(**int** N, **int** t, **int** b, **int** K, **int** r, **int** δ, **int** L, **Classifier** $f_{\mathcal{O}}$)

$S \leftarrow \emptyset, \quad S_b \leftarrow \emptyset$
$QUERIES \leftarrow \mathcal{U}(\mathcal{D}), \quad ORACLE \leftarrow f_{\mathcal{O}}$
repeat ▷ Sample until all classes reach a minimum size
  $\hat{S} \leftarrow \emptyset$
  for $i = 0$ to tN **do** ▷ Sample points uniformly
    $z_a \sim QUERIES, \quad y_a \leftarrow ORACLE(z_a)$
    $\hat{S} \leftarrow \hat{S} \cup \{(z_a, y_a)\}$
  end for
  for each $k \in \{1, \ldots, K\}$ such that $|\hat{S}_k| == 0$ **do**
    for $i = 0$ to L **do**
      $z_b \leftarrow SEEDS(k), \quad y_b \leftarrow ORACLE(z_b)$
      $\hat{S} \leftarrow \hat{S} \cup \{(z_b, y_b)\}$
      for $i = 0$ to $\frac{N/K-b}{L} - 1$ **do** ▷ Exploit vicinity
        repeat
          $v \sim \mathcal{N}_d(0, 1), \quad ||v|| \sim \mathcal{U}(0, r)$
          $z_c \leftarrow z_b + ||v||v, \quad y_c \leftarrow ORACLE(z_c)$
          $\hat{S} \leftarrow \hat{S} \cup \{(z_c, y_c)\}, \quad r \leftarrow r - \delta$
        until $y_c == y_b$
      end for
    end for
  end for each
  $S \leftarrow S \cup \hat{S}$
until $\forall k \backslash |S_k| \geq N/K - b$
for each $k \in \{1, \ldots, K\}$ **do**
  $S_b \leftarrow S_b \cup sample(S_k, N/K)$ ▷ Down sample
end for each
return S_b

A querying protocol

We define three kinds of information supply:

- QUERIES(): On feature domain $\mathcal{D}$ and probability distribution P_Z as inputs, it outputs a vector $\boldsymbol{z}_j \sim P_Z$ such that $\boldsymbol{z}_j \in \mathcal{D}$. We require that the vector $\boldsymbol{z}_j$ be total, *i.e.* that all its variables be *determined*.
- ORACLE(): Given a vector $\boldsymbol{z}_j$ as input, it outputs a class label y_j, such that $y_j = f_{\mathcal{O}}(\boldsymbol{z}_j)$.
- SEEDS(): Given a class label as input, it gives as output a sample vector $\boldsymbol{s}_i \in \boldsymbol{S}$, which is a positive example of the given class.

First, we have access to a query generator. This generator returns sample data points that follow a certain distribution and which belong to the original feature domain. A call to a routine QUERIES produces one of such samples. The second source of information is a routine ORACLE, which accepts query samples to produce synthetic pairs $(\boldsymbol{z}_j, y_j)$, for $y_j = f_{\mathcal{O}}(\boldsymbol{z}_j)$. While many types of queries have been described in the literature [65], [66], [67], we here only allow *membership queries*, which output a class label. Thus, when presented with a sample, routine ORACLE outputs a prediction that, assuming $f_{\mathcal{O}}$ is a classifier, corresponds to a class label.

The generator behind QUERIES could potentially follow many different heuristics. Ultimately, our objective is to obtain a finite set of optimal labelled samples. This is challenging because the form of the original boundary is unknown to us. Moreover, because we have *a priori* no information about the original data distribution, we lack any insight on how the different classes may be distributed throughout the space. In theory, we could generate an infinite number of query points. This approach, however, is not tractable in practice, since we are limited by our computational resources.

A naïve balanced generator

We propose the naïve balanced generator as a preliminary approach to solve the volume unbalanced effect discussed in previous sections. In the simplest case, we assume the distribution P_Z to be flat in the domain, $P_Z \sim \mathcal{U}$, so that QUERIES is a randomized algorithm that draws syn-

thetic samples from a uniform distribution. The complete algorithm for the naïve balanced generator is shown in *Algorithm 1*. We generate an initial set of tN samples, for N the desired size of the synthetic dataset and t a tunable parameter. We label these samples according to routine ORACLE and monitor the frequency of each class. To ensure that all classes in the original dataset are equivalently represented in the synthetic set, we impose that the relative frequency of the different labels lie in the bounded interval $[N/K - b, N/K + b]$, for K the number of classes and b a balance controlling parameter. We repeat the process until we obtain a sufficient number of samples for all labels, *i.e.* a frequency of at least $N/K - b$ for all classes. We then down-sample the more populated classes using a random sampler within each class buffer.

In most cases this setting suffices to obtain a representative set of points. In rare occasions, however, the sole use of routines QUERIES and ORACLE is not sufficient. This occurs when volume unbalance is very pronounced and consecutive calls to QUERIES fail to produce samples of one or more classes. To overcome this issue, we introduce a third routine SEEDS. A call to routine SEEDS recovers data points from the original dataset that represent positive instances for the sub-sampled class or classes. We exploit the neighborhood of the seeded samples to obtain additional data points for the sub-sampled classes. We generate random vectors with a direction drawn from a normal distribution and a module drawn from a uniform distribution bounded in $[0, s]$, for a parameter s. We start with a sufficiently large s and iteratively decrease its value with step δ until all generated samples are assigned a label equal to that of the source point. We proceed like this until we obtain a minimum number of $N/K - b$ for all classes[6].

[6]In the UCI experiment, five datasets resort to the routine SEEDS to obtain samples of at least one of the classes. This is the case for some iterations of *congressional-voting*, *ilpd-indian-liver*, *pittsburg-bridges-T-OR-D*, *spambase* and *statlog-image*.

Table 1: Experimental Results for the 60 Datasets in the UCI Experiment[5]

Dataset	Classes	Samples	Features	Original	$\mathcal{A}_{\mathcal{O}}$	*decision_tree*			*logistic_regression*			*random_forest*		
						$\mathcal{A}_{\mathcal{C}}$	$\widehat{\mathcal{A}}_{\mathcal{C}}$	$R^{\boldsymbol{S}}_{\mathcal{F}}$	$\mathcal{A}_{\mathcal{C}}$	$\widehat{\mathcal{A}}_{\mathcal{C}}$	$R^{\boldsymbol{S}}_{\mathcal{F}}$	$\mathcal{A}_{\mathcal{C}}$	$\widehat{\mathcal{A}}_{\mathcal{C}}$	$R^{\boldsymbol{S}}_{\mathcal{F}}$
abalone	3	3341	8	*adaboost*	0.57	0.52 ± 0.02	0.5653 ± 0.0002	0.27 ± 0.02	0.57 ± 0.00	0.3266 ± 0.0001	0.41 ± 0.00	0.58 ± 0.01	0.5590 ± 0.0109	0.27 ± 0.01
acute-inflammation	2	96	6	*adaboost*	1	1.00 ± 0.00	1.0000 ± 0.0000	0.00 ± 0.00	1.00 ± 0.00	1.0000 ± 0.0000	0.00 ± 0.00	1.00 ± 0.00	1.0000 ± 0.0000	0.00 ± 0.00
acute-nephritis	2	96	6	*adaboost*	1	1.00 ± 0.00	1.0000 ± 0.0000	0.00 ± 0.00	1.00 ± 0.00	0.9987 ± 0.0000	0.00 ± 0.00	1.00 ± 0.00	1.0000 ± 0.0000	0.00 ± 0.00
bank	2	3616	16	*adaboost*	0.85	0.82 ± 0.03	0.8428 ± 0.0004	0.10 ± 0.03	0.87 ± 0.00	0.5558 ± 0.0001	0.12 ± 0.00	0.87 ± 0.00	0.8353 ± 0.0026	0.07 ± 0.00
blood	2	598	4	*adaboost*	0.71	0.65 ± 0.04	0.7000 ± 0.0001	0.14 ± 0.05	0.67 ± 0.00	0.5190 ± 0.0001	0.38 ± 0.00	0.64 ± 0.02	0.7018 ± 0.0226	0.16 ± 0.04
breast-cancer	2	228	9	*adaboost*	0.74	0.74 ± 0.01	0.7411 ± 0.0000	0.00 ± 0.01	0.67 ± 0.00	0.4731 ± 0.0001	0.31 ± 0.00	0.74 ± 0.00	0.7412 ± 0.0000	0.00 ± 0.00
breast-cancer-wisc	2	559	9	*adaboost*	**0.93**	0.93 ± 0.00	0.9286 ± 0.0000	0.00 ± 0.00	**0.93 ± 0.00**	0.6333 ± 0.0001	0.06 ± 0.00	0.93 ± 0.00	0.9286 ± 0.0000	0.00 ± 0.00
breast-cancer-wisc-diag	2	455	30	*adaboost*	0.95	0.95 ± 0.00	0.9473 ± 0.0000	0.00 ± 0.00	0.95 ± 0.00	0.7444 ± 0.0000	0.04 ± 0.00	0.95 ± 0.00	0.9473 ± 0.0000	0.00 ± 0.00
breast-cancer-wisc-prog	2	158	33	*adaboost*	0.73	0.72 ± 0.00	0.7250 ± 0.0000	0.00 ± 0.00	0.62 ± 0.00	0.5468 ± 0.0001	0.35 ± 0.00	0.72 ± 0.00	0.7249 ± 0.0000	0.00 ± 0.00
breast-tissue	6	84	9	*adaboost*	0.59	0.61 ± 0.02	0.5909 ± 0.0000	0.16 ± 0.04	0.36 ± 0.00	0.3838 ± 0.0001	0.59 ± 0.00	0.57 ± 0.02	0.5909 ± 0.0223	0.18 ± 0.01
chess-krvkp	2	2556	36	*ann*	0.99	0.89 ± 0.02	0.9527 ± 0.0002	0.11 ± 0.02	0.91 ± 0.00	0.9603 ± 0.0000	0.10 ± 0.00	0.91 ± 0.01	0.9670 ± 0.0091	0.09 ± 0.01
congressional-voting	2	348	16	*ann*	0.61	0.61 ± 0.00	0.6092 ± 0.0000	0.00 ± 0.00	0.61 ± 0.00	0.6087 ± 0.0000	0.00 ± 0.00	0.61 ± 0.00	0.6092 ± 0.0000	0.00 ± 0.00
conn-bench-sonar-mines-rocks	2	166	60	*ann*	0.88	0.58 ± 0.08	0.5897 ± 0.0004	0.40 ± 0.07	0.81 ± 0.01	0.8079 ± 0.0000	0.12 ± 0.01	0.69 ± 0.03	0.7405 ± 0.0327	0.21 ± 0.04
connect-4	2	54045	42	*ann*	0.87	0.54 ± 0.04	0.6518 ± 0.0003	0.46 ± 0.04	0.60 ± 0.00	0.6871 ± 0.0000	0.39 ± 0.00	0.66 ± 0.00	0.7442 ± 0.0031	0.32 ± 0.00
contrac	3	1178	9	*ann*	0.55	0.55 ± 0.01	0.5251 ± 0.0001	0.07 ± 0.02	0.50 ± 0.00	0.4111 ± 0.0000	0.40 ± 0.00	0.55 ± 0.00	0.5357 ± 0.0043	0.03 ± 0.01
credit-approval	2	552	15	*ann*	0.79	0.80 ± 0.02	0.7268 ± 0.0003	0.10 ± 0.02	0.72 ± 0.00	0.7066 ± 0.0000	0.19 ± 0.00	0.79 ± 0.01	0.7527 ± 0.0071	0.03 ± 0.01
cylinder-bands	2	409	35	*ann*	0.69	0.60 ± 0.05	0.6083 ± 0.0002	0.37 ± 0.05	0.71 ± 0.00	0.6354 ± 0.0000	0.25 ± 0.00	0.65 ± 0.01	0.6445 ± 0.0121	0.35 ± 0.01
echocardiogram	2	104	10	*ann*	**0.3**	0.33 ± 0.04	0.2879 ± 0.0000	0.05 ± 0.03	0.30 ± 0.00	0.2960 ± 0.0000	0.00 ± 0.00	0.30 ± 0.00	0.2922 ± 0.0000	0.00 ± 0.00
energy-y1	3	614	8	*ann*	**0.96**	**0.96 ± 0.01**	0.9537 ± 0.0001	0.00 ± 0.01	**0.78 ± 0.00**	0.7744 ± 0.0001	0.23 ± 0.00	**0.96 ± 0.00**	0.9551 ± 0.0039	0.00 ± 0.00
energy-y2	3	614	8	*ann*	0.84	0.84 ± 0.00	0.8403 ± 0.0001	0.00 ± 0.00	0.79 ± 0.00	0.7835 ± 0.0001	0.09 ± 0.00	0.84 ± 0.01	0.8410 ± 0.0065	0.01 ± 0.00
fertility	2	80	9	*random_forest*	0.9	0.90 ± 0.00	0.8993 ± 0.0000	0.00 ± 0.00	0.75 ± 0.00	0.8255 ± 0.0000	0.15 ± 0.00	0.90 ± 0.00	0.8993 ± 0.0000	0.00 ± 0.00
haberman-survival	2	244	3	*random_forest*	0.61	0.61 ± 0.01	0.6125 ± 0.0000	0.01 ± 0.01	0.61 ± 0.00	0.4888 ± 0.0001	0.26 ± 0.00	0.61 ± 0.01	0.6123 ± 0.0074	0.00 ± 0.01
heart-hungarian	2	235	12	*random_forest*	0.76	0.79 ± 0.04	0.7445 ± 0.0002	0.07 ± 0.03	0.81 ± 0.00	0.6457 ± 0.0001	0.05 ± 0.00	0.80 ± 0.00	0.7498 ± 0.0000	0.03 ± 0.00
hepatitis	2	124	19	*random_forest*	0.74	0.74 ± 0.05	0.6991 ± 0.0002	0.05 ± 0.04	0.90 ± 0.00	0.5973 ± 0.0000	0.16 ± 0.00	0.75 ± 0.01	0.7145 ± 0.0097	0.00 ± 0.01
ilpd-indian-liver	2	466	9	*random_forest*	0.62	0.59 ± 0.02	0.6207 ± 0.0001	0.36 ± 0.02	0.56 ± 0.00	0.6020 ± 0.0000	0.47 ± 0.00	0.62 ± 0.01	0.6224 ± 0.0089	0.37 ± 0.01
ionosphere	2	280	33	*random_forest*	0.94	0.92 ± 0.03	0.8543 ± 0.0004	0.07 ± 0.03	0.89 ± 0.00	0.7793 ± 0.0000	0.11 ± 0.00	0.95 ± 0.01	0.8867 ± 0.0138	0.05 ± 0.01
iris	3	120	4	*random_forest*	0.93	**0.95 ± 0.02**	0.9332 ± 0.0000	0.02 ± 0.02	**0.70 ± 0.00**	0.6778 ± 0.0003	0.30 ± 0.00	**0.95 ± 0.02**	0.9333 ± 0.0163	0.02 ± 0.02
magic	2	15216	10	*random_forest*	0.88	0.83 ± 0.01	0.8024 ± 0.0003	0.10 ± 0.01	0.79 ± 0.00	0.6809 ± 0.0001	0.17 ± 0.00	0.86 ± 0.00	0.8316 ± 0.0015	0.06 ± 0.00
mammographic	2	768	5	*random_forest*	0.8	0.80 ± 0.00	0.7974 ± 0.0001	0.01 ± 0.00	0.76 ± 0.00	0.5377 ± 0.0001	0.19 ± 0.00	0.80 ± 0.00	0.7974 ± 0.0039	0.01 ± 0.00
miniboone	2	104051	50	*random_forest*	0.94	0.86 ± 0.01	0.8402 ± 0.0004	0.12 ± 0.01	0.84 ± 0.00	0.6951 ± 0.0004	0.15 ± 0.00	0.90 ± 0.00	0.8679 ± 0.0011	0.07 ± 0.00
molec-biol-splice	3	2552	60	*linear_svm*	0.84	0.77 ± 0.01	0.7151 ± 0.0004	0.17 ± 0.01	0.77 ± 0.00	0.7801 ± 0.0001	0.14 ± 0.00	0.80 ± 0.00	0.7590 ± 0.0040	0.15 ± 0.00
mushroom	2	6499	21	*linear_svm*	0.98	0.95 ± 0.02	0.9526 ± 0.0002	0.03 ± 0.02	0.98 ± 0.00	0.9777 ± 0.0000	0.00 ± 0.00	0.87 ± 0.05	0.9549 ± 0.0544	0.11 ± 0.06
musk-1	2	380	166	*linear_svm*	**0.88**	0.54 ± 0.04	0.5620 ± 0.0003	0.46 ± 0.05	0.88 ± 0.00	0.8732 ± 0.0000	0.01 ± 0.00	**0.67 ± 0.06**	0.7323 ± 0.0577	0.32 ± 0.04
musk-2	2	5278	166	*linear_svm*	**0.96**	0.50 ± 0.05	0.6005 ± 0.0004	0.50 ± 0.05	0.96 ± 0.00	0.9556 ± 0.0000	0.00 ± 0.00	**0.56 ± 0.04**	0.7745 ± 0.0375	0.44 ± 0.04
oocytes_merluccius_nucleus_4d	2	817	41	*linear_svm*	0.82	0.47 ± 0.06	0.6460 ± 0.0003	0.52 ± 0.06	0.81 ± 0.00	0.8144 ± 0.0000	0.00 ± 0.00	0.59 ± 0.03	0.7317 ± 0.0264	0.38 ± 0.03
oocytes_trisopterus_nucleus_2f	2	729	25	*linear_svm*	0.81	0.56 ± 0.05	0.6946 ± 0.0002	0.43 ± 0.07	0.81 ± 0.00	0.8084 ± 0.0000	0.00 ± 0.00	0.63 ± 0.03	0.7543 ± 0.0296	0.32 ± 0.03
parkinsons	2	156	22	*linear_svm*	0.9	0.83 ± 0.04	0.7990 ± 0.0003	0.11 ± 0.06	0.90 ± 0.00	0.8950 ± 0.0000	0.00 ± 0.00	0.89 ± 0.01	0.8562 ± 0.0126	0.02 ± 0.02
pima	2	614	8	*linear_svm*	0.72	0.72 ± 0.01	0.6967 ± 0.0001	0.04 ± 0.01	0.72 ± 0.00	0.7203 ± 0.0000	0.00 ± 0.00	0.71 ± 0.01	0.7097 ± 0.0050	0.02 ± 0.01
pittsburg-bridges-MATERIAL	3	84	7	*linear_svm*	0.91	0.91 ± 0.00	0.9090 ± 0.0000	0.00 ± 0.00	0.91 ± 0.00	0.8968 ± 0.0000	0.00 ± 0.00	0.91 ± 0.00	0.9091 ± 0.0000	0.00 ± 0.00
pittsburg-bridges-REL-L	3	82	7	*linear_svm*	**0.67**	0.67 ± 0.00	0.6667 ± 0.0000	0.00 ± 0.00	0.67 ± 0.00	0.6667 ± 0.0000	0.00 ± 0.00	0.67 ± 0.00	0.6667 ± 0.0000	0.00 ± 0.00
pittsburg-bridges-T-OR-D	2	81	7	*rbf_svm*	0.86	0.86 ± 0.00	0.8562 ± 0.0000	0.00 ± 0.00	0.90 ± 0.00	0.6932 ± 0.0001	0.24 ± 0.00	0.86 ± 0.00	0.8568 ± 0.0000	0.00 ± 0.00
planning	2	145	12	*rbf_svm*	**0.7**	0.70 ± 0.00	0.7027 ± 0.0000	0.00 ± 0.00	-	-	-	0.70 ± 0.00	0.7027 ± 0.0000	0.00 ± 0.00
ringnorm	2	5920	18	*rbf_svm*	0.98	0.88 ± 0.01	0.7992 ± 0.0005	0.11 ± 0.01	0.74 ± 0.00	0.7049 ± 0.0000	0.26 ± 0.00	0.95 ± 0.00	0.8877 ± 0.0022	0.05 ± 0.00
seeds	3	168	7	*rbf_svm*	0.88	0.90 ± 0.02	0.8006 ± 0.0003	0.06 ± 0.03	0.88 ± 0.00	0.8582 ± 0.0001	0.00 ± 0.00	0.92 ± 0.02	0.8487 ± 0.0158	0.04 ± 0.02
spambase	2	3680	57	*rbf_svm*	0.93	0.45 ± 0.10	0.7610 ± 0.0002	0.55 ± 0.11	0.92 ± 0.00	0.9227 ± 0.0000	0.02 ± 0.00	0.53 ± 0.08	0.8577 ± 0.0824	0.48 ± 0.09
statlog-australian-credit	2	552	14	*rbf_svm*	**0.68**	0.68 ± 0.00	0.6812 ± 0.0000	0.00 ± 0.00	-	-	-	0.68 ± 0.00	0.6812 ± 0.0000	0.00 ± 0.00
statlog-german-credit	2	800	24	*rbf_svm*	0.79	0.59 ± 0.03	0.6820 ± 0.0003	0.36 ± 0.04	0.78 ± 0.00	0.7817 ± 0.0000	0.02 ± 0.00	0.76 ± 0.01	0.7369 ± 0.0121	0.07 ± 0.01
statlog-heart	2	216	13	*rbf_svm*	0.85	0.81 ± 0.02	0.8051 ± 0.0002	0.05 ± 0.01	0.85 ± 0.00	0.8510 ± 0.0000	0.00 ± 0.00	0.81 ± 0.01	0.8271 ± 0.0056	0.04 ± 0.01
statlog-image	7	1848	18	*rbf_svm*	0.95	0.74 ± 0.01	0.8351 ± 0.0003	0.25 ± 0.02	0.74 ± 0.00	0.8218 ± 0.0001	0.25 ± 0.00	0.78 ± 0.01	0.8993 ± 0.0076	0.21 ± 0.01
statlog-vehicle	4	676	18	*rbf_svm*	0.85	0.56 ± 0.05	0.5489 ± 0.0002	0.40 ± 0.04	0.66 ± 0.00	0.6618 ± 0.0000	0.33 ± 0.00	0.65 ± 0.02	0.6971 ± 0.0186	0.29 ± 0.02
synthetic-control	6	480	60	*xgboost*	0.96	0.75 ± 0.02	0.6547 ± 0.0007	0.23 ± 0.03	0.81 ± 0.00	0.6096 ± 0.0001	0.22 ± 0.00	0.94 ± 0.01	0.7683 ± 0.0095	0.02 ± 0.01
teaching	3	120	5	*xgboost*	0.55	0.55 ± 0.02	0.5476 ± 0.0000	0.01 ± 0.01	0.35 ± 0.00	0.3047 ± 0.0002	0.65 ± 0.00	0.55 ± 0.01	0.5476 ± 0.0097	0.00 ± 0.01
tic-tac-toe	2	766	9	*xgboost*	0.97	0.97 ± 0.00	0.9740 ± 0.0000	0.00 ± 0.00	0.71 ± 0.00	0.8224 ± 0.0002	0.26 ± 0.00	0.97 ± 0.00	0.9740 ± 0.0000	0.00 ± 0.00
titanic	2	1760	3	*xgboost*	0.78	0.78 ± 0.00	0.7778 ± 0.0000	0.00 ± 0.00	0.76 ± 0.00	0.5837 ± 0.0000	0.10 ± 0.00	0.78 ± 0.00	0.7778 ± 0.0000	0.00 ± 0.00
twonorm	2	5920	20	*xgboost*	0.98	0.87 ± 0.01	0.7454 ± 0.0003	0.13 ± 0.01	0.98 ± 0.00	0.8913 ± 0.0000	0.02 ± 0.00	0.96 ± 0.00	0.8842 ± 0.0021	0.02 ± 0.00
vertebral-column-2clases	2	248	6	*xgboost*	0.77	0.78 ± 0.02	0.7669 ± 0.0001	0.05 ± 0.02	0.81 ± 0.01	0.7244 ± 0.0001	0.14 ± 0.01	0.80 ± 0.01	0.7687 ± 0.0074	0.02 ± 0.01
vertebral-column-3clases	3	248	6	*xgboost*	0.84	0.84 ± 0.02	0.8379 ± 0.0000	0.02 ± 0.01	0.80 ± 0.00	0.7575 ± 0.0000	0.10 ± 0.00	0.84 ± 0.00	0.8378 ± 0.0000	0.00 ± 0.00
waveform	3	4000	21	*xgboost*	0.84	0.77 ± 0.01	0.6250 ± 0.0005	0.18 ± 0.01	0.85 ± 0.00	0.7053 ± 0.0000	0.09 ± 0.00	0.83 ± 0.00	0.7318 ± 0.0044	0.08 ± 0.00
waveform-noise	3	4000	40	*xgboost*	0.84	0.76 ± 0.01	0.6610 ± 0.0004	0.19 ± 0.01	0.87 ± 0.00	0.7083 ± 0.0000	0.08 ± 0.00	0.85 ± 0.00	0.7415 ± 0.0043	0.07 ± 0.00
wine	3	142	11	*xgboost*	**0.92**	0.92 ± 0.00	0.9147 ± 0.0000	0.00 ± 0.00	**0.94 ± 0.00**	0.7031 ± 0.0001	0.08 ± 0.00	0.92 ± 0.00	0.9147 ± 0.0000	0.00 ± 0.00

Table 2: Validation Results for Models Trained to Predict *gender* and *race*

Target	Class label	Count	Accuracy
Gender	*Male*	93	0.77 ± 0.15
	Female	37	0.34 ± 0.06
	Other	2	0.1 ± 0.2
Race	*Other*	62	0.60 ± 0.06
	Human	37	0.54 ± 0.05
	Mutant	18	0.30 ± 0.04
	God / Eternal	4	0.09 ± 0.15
	Alpha	3	-
	Kryptonian	2	0.93 ± 0.17
	Frost Giant	2	-
	Demon	2	-
	Alien	2	0.03 ± 0.11