Journal of Machine Learning Research 1 (2022) 1-48 Submitted 4/00; Published 10/00

Adapting Machine Learning Classifiers using Sequential Copies

Nahuel Statuto NAHUEL.STATUTO@ESADE.EDU
Department of Operations, Innovation and Data Sciences
Universitat Ramon Llull, ESADE
Sant Cugat del Vallès, 08172, Catalonia, Spain

Irene Unceta IRENE.UNCETA@ESADE.EDU
Department of Operations, Innovation and Data Sciences
Universitat Ramon Llull, ESADE
Sant Cugat del Vallès, 08172, Catalonia, Spain

Jordi Nin JORDI.NIN@ESADE.EDU
Department of Operations, Innovation and Data Sciences
Universitat Ramon Llull, ESADE
Sant Cugat del Vallès, 08172, Catalonia, Spain

Oriol Pujol ORIOL_PUJOL@UB.EDU
Department of Mathematics and Computer Science
Universitat de Barcelona
Barcelona, 08007, Catalonia, Spain

Editor:

Abstract

Copying corresponds to the process of replicating the decision behaviour of a machine learning model using another that is endowed with new features and characteristics. This process plays a relevant role in circumstances where external constraints limit the performance of a predictive system. In such cases, copies can retain the original prediction capabilities, while showing additional properties that better adapt the existing solution to the demands of its environment. So far, copies have been studied under the single-pass framework. In this work, we propose an extension to introduce the sequential approach. The main contribution of this new approach is to limit the number of computer resources needed to train or maintain a copy. This computer resource reduction allows companies to reduce the maintenance costs of machine learning models in production. To empirically validate the sequential approach, this work contains experiments with synthetic and real-world datasets.

Keywords: Sustainable AI, machine learning copies, knowledge transfer

1. Introduction

As machine learning models are becoming bigger and more complex, they require increasing volumes of computational and human resources (Bianchini and Scarselli, 2014; Rieck et al., 2019). This investment, however, is often short-lived as models quickly become obsolete due to changes in their immediate environment (Hong et al., 2020). These changes can be either internal or external to the companies deploying the models, and are related to new

production requirements, updates in the technological infrastructure, new market trends or shifts in regulation. Independently of their source, changes result in a constant need to re-think and re-train models (Sculley et al., 2015; Schelter et al., 2018). This, together with the aforementioned high cost associated to this process, makes machine learning sustainability one of the main challenges faced by industrial machine learning practitioners today(Paleyes et al., 2022).

In this context, copies (Unceta et al., 2020a) have been proposed as a viable, agile solution to re-use a machine learning model's acquired knowledge to train a new one that is better adapted to the demands of the environment (Unceta et al., 2020b). This approach, similar to knowledge distillation (Hinton et al., 2015; Buciluǎ et al., 2006), effectively circumvents the issue of having to re-train models from scratch and therefore has numerous benefits in terms of cost and use of companies' resources. In practice, copying refers to the process of projecting an existing decision function onto a new hypothesis space that satisfies the changing requirements. This process is agnostic to the form of the original model, for which only its decision behavior is indirectly known by means of a hard predictions query interface, as opposed to scenarios where label probabilities can be used as soft targets for training the new model (Liu et al., 2018b; Wang et al., 2020).Besides, copying is also performed without access to the data the original model was train with. Instead, the copy is trained using synthetic samples drawn at random from a given probability distribution and labelled according to the predictions of the original model.

In this zero-knowledge setting, the problem of copying therefore involves two simultaneous optimizations. On the one hand, we need to gather knowledge about the model's decision behavior throughout the space by generating well-distributed synthetic data points. On the other hand, we need to obtain the optimal copy parameters to fit the resulting dataset. Previous papers on copying have focused on solving this problem in a single step (Unceta et al., 2020a), by creating a massive synthetic dataset and training a copy on it. This approach is known as the single-pass approach.

In this article, we study a novel approach to build a copy from a sequence of small synthetic datasets instead. In particular, we propose a methodology that exploits the memorization capacity of the copy to sequentially refine its decision boundary until it replicates the original model behaviour to a desired level of fidelity. This algorithm is inspired by strategies of learning by boredom, where samples are recurrently presented to a model until the desired error tolerance is achieved. We summarize the main contributions of this paper as:

- We formalize the problem of building a copy from a sequence of small synthetic datasets.
- We explore the theoretical framework and implications of this setting, and study how these can be exploited when building a copy in a sequential fashion.
- We validate our proposal in a series of well-established classification problems. These include for binary and multiclass problems for a varying number of dimensions. We evaluate sequential copying framework in terms of its predictive accuracy but also in terms of its computational cost and required memory allocation.

The rest of this article is organized as follows. Section 2 presents a literature survey of related work. Then, we formalize the copying problem in Section 3. Later, section 4 extends the current state-of-the-art introducing the sequential copying approach and providing guidelines for its practical implementation. Then, Section 5 describes some preliminary results on a set of toy problems. Finally, The paper concludes with a summary of our findings and an outline of future research.

2. Related Work

Machine learning models are designed to find an approximation of the unknown function that underlies the predictive relationship between the inputs and outputs. The use of machine learning is motivated by the need to transform this relationship function, which is usually not accessible, to another, which we can control and which is, therefore, more suitable under certain circumstances. Conceptually, we identify the acquired knowledge with the trained model learned parameters.

The idea of taking advantage of the knowledge of one machine learning model to train another has been studied from different perspectives and scenarios in the literature (Breiman, 2001; Buciluǎ et al., 2006; Hinton et al., 2015). Recently, this notion has become more popular under the umbrella of the knowledge distillation research field, where different authors study how the knowledge acquired by a complex model, the teacher, can be exploited to guide the training of a simpler model, the student (Liu et al., 2018b).

The main reason to proceed this way is that most commercial machine learning applications deal with non-trivial problems, such as speech recognition or computer vision. To train these machine learning models, we require extracting the inherent relationships from large and redundant datasets. This task usually involves using high-capacity models, such as deep neural networks, causing high costs for companies in terms of the required computational resources and the training time. However, empirical work has shown that shallower models can approximate arbitrary decision boundaries regardless of their complexity or shape (Cybenko, 1989). However, shallow models tend to harder to train than deeper ones, which have the added benefit of ensuring a generally improved predictive performance on raw data (Dauphin and Bengio, 2013; Eigen et al., 2013).

An essential step in distillation is the creation of a transfer set to train the student model. Traditionally, knowledge transfer has been considered a standard learning process, where the training samples are relabelled and extended to learn an alternative model (Bologna and Hayashi, 2018). Different methods have been proposed for the relabelling step. For instance in (Che et al., 2016; Hinton et al., 2015), the authors directly enrich the raw training data points by adding the class probabilities provided by the teacher. Alternatively, in (Bastani et al., 2018; Liu et al., 2018a) the authors generate additional synthetic training data points by using another model to extend and enrich the training set. This sampling generation is the main idea of the adversarial learning research area (Dalvi et al., 2004; Lowd and Meek, 2005). However, in this case, the goal is not to defeat the teacher, it is to help the student to learn class decision boundaries by providing the most representative training samples.

One of the conclusions of all these research works is that the performance of a machine learning model is highly dependent on the training set. Besides being small and minimizing data redundancies, a desirable training set must represent the decision boundaries smoothly.

The sequential copying framework is a possible but unexplored way to generate a valid training dataset while the copy is fine-tunned sequentially.

3. Mathematical Framework

In what follows we mathematically formalize the problem of copying and discuss the existing approaches to solving it.

3.1 Problem Statement

Let us assume a set of data points $D \subseteq \mathbb{R}^d$ for d the dimensionality of the input space. Let us further assume that these points are labelled according to a set of classes C_i of cardinality n_c. We can define a classifier trained on these data as a function f that maps each data point in D into a class

$$f : \mathbb{R}^d \longrightarrow \{0, 1\}^{n_c}$$

for n_c the total number of classes. According to this definition, the value of f on any given point $z \in \mathbb{R}^d$ can then be expressed as

$$f(z) = [\underbrace{0, \ldots, 1, \ldots, 0}_{n_c}],$$

a n_c-vector with zeros in all the positions except for the position i, which corresponds to the class C_i predicted for this point. In this context, we can define a copy as a new classifier g, parameterized by $\theta \in \Theta$, whose decision function mimics f all over the sample space.

Mathematically, we can define a copy as

$$g : \mathbb{R}^d \times \Theta \longrightarrow \big[0, 1\big]^{n_c}.$$

Note that according to this definition a copy returns a n_c-vector of probabilities instead of a binary result for each class. This is, when applied on a given synthetic data point $z \in \mathbb{R}^d$ generated without any knowledge of the original data points D, the copy returns a vector expressing the predicted probability that the considered sample belongs to each of the n_c classes

$$g(z, \theta) = [p_{C_1}, \ldots, p_{C_i}, \ldots, p_{C_{n_c}}],$$

for p_{C_i} the predicted probability for class C_i.

Copying is characterized by the predictive distribution $P(\theta|f, g)$. This distribution can be expressed as a new function $F(\theta)$ of the copy parameters as follows

$$F(\theta) = \int_{z \in \mathbb{R}^d} \mathcal{P}(\theta|f(z), g)\ \mathrm{d}z \tag{1}$$

where the integral runs through the totality of the space $\mathbb{R}^d$. Hence, the problem of copying can be reduced to finding the optimal parameters θ^* such that $F(\theta)$ is maximal

$$\theta^* = \arg\max_{\theta} F(\theta) \tag{2}$$

The integral in Eq. 1 has no analytical solution. In theory, we could potentially generate an infinite number of samples z at random and evaluate the integral on them. In practice,

however, we can only generate a finite set of data points. Solving this problem in practice therefore requires that we discretize the expressions above. In particular, we consider $S \subseteq \mathbb{R}^d$ to be a discrete set of synthetic samples z and rewrite Eq. 1 as follows

$$F(\theta) = \sum_{z \in S} \mathcal{P}(\theta | f(z), g). \tag{3}$$

We can then obtain the optimal copy parameters θ^* by maximizing this expression. The most straight-forward approach to solving this optimization problem is assuming that $F(\theta)$ takes the discrete form above and using a single iteration of an alternating projection optimization scheme. We refer to this approach as the *single-pass* solution.

3.2 The single-pass approach

In the single-pass approach we approximate the integral in Eq. 1 using a discrete subset $S_s \subseteq S$ of artificial data points $z \in \mathbb{R}^d$ drawn at random from a given probability distribution. We estimate the optimal parameter set for the copy using this subset as

$$\theta^*_{S_s} = \arg \max_{\theta} F(\theta).$$

The resulting value approaches the true value θ^* as $S_s \to S$. Hence, the bigger the size of the generated subset the closer our estimated parameters will be to the true optimal parameters. Note, however, that a higher cardinality of the set S_s alone is not sufficient to guarantee a good approximation of the optimal parameters. A main limitation of the single-pass approach is that samples are generated without any feedback mechanism. Hence, there exists no effective system to ensure that the resulting set includes representative enough samples. Or, in other words, that the generated samples are relevant for the considered task. This issue is worsened as the dimensionality of the problem increases.

Moreover, an additional drawback of the single-pass approach is that we are limited by the available computational resources of the system used to train the copy. This means that the cardinality of S will be bound by the amount of memory available. This limitation is amplified in information systems such as relational databases, where memory resources are scarce. Therefore, in practice, generating a large amount of high-dimensional samples quickly becomes infeasible. As a result, the single-pass approach often yields sub-optimal results.

In this work we propose an alternative approach, that builds upon the single-pass to generate successive sets S in a sequential form. This circumvents the issue of memory storage, since a new set of samples can be generated from iteration to iteration, effectively erasing the memory each time. In addition, it allows for feedback to enter the system, such that we can estimate the goodness of the generated samples and the resulting copy parameters at each time step. In what follows, we introduce the *sequential approach* to the copying problem and discuss possible practical implementations.

4. The sequential approach

Let us begin by introducing a sequence of finite subsets S_i such that

$$S_i \subseteq S_{i+1} \subseteq \cdots \subseteq S$$

for $i \in \mathbb{N}$. This sequence is defined such that for increasing values of i the subset S_i grow closer to S, and in the limit where i approaches infinity $\lim_{i\to\infty} S_i = S$.

Taking these new subsets as reference, we can re-define the function F_i corresponding to each subset S_i in this sequence as

$$F_i(\theta) = \sum_{z\in S_i} \mathcal{P}(\theta|f(z), g) \tag{4}$$

and consequently introduce the instantaneous estimation of the optimal parameter set θ_i^* according to the expression below:

$$\theta_i^* = \arg\max_{\theta} F_i(\theta). \tag{5}$$

We can think of this expression as representing an intermediate *optimal* step for the copy parameters. The idea behind the sequential approach is that if the condition that the S_i converge to S is sufficient to ensure that the θ_i^* equally converge to θ^*, then we can iteratively approximate the optimal parameter set on artificial subsets generated by drawing samples at random. Hence, in what follows we show that solving the optimization problem in Eq. 2 equals solving the same problem in Eq. 5 in the limit where i approaches infinity. To do this, we first prove that the sequence of functions F_i converges to $F(\theta)$ for increasing values of i, *i.e.* that $\lim_{i\to\infty} F_i(\theta) = F(\theta)$. Then, we also prove that from this result it derives that the sequence of θ_i^* converges to θ^* as i approaches infinity. We do so under several assumptions.

4.1 Uniform convergence for F_i

Let us introduce the following proposition on the uniform convergence of functions.

Proposition 1 *A sequence of functions $f_i : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$ is uniformly convergent to a limit function $f : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$, if and only if*

$$||f - f_i||_\infty \xrightarrow{i\to\infty} 0$$

where $||\cdot||_\infty$ denotes the supremum norm of the functions $f_i - f$ on D [1].

On this basis, we propose the following theorem.

Theorem 2 *A sequence of functions $\{F_i\}_i$ defined as $F_i(\theta) = \sum_{z\in S_i} \mathcal{P}(\theta|f(z), g)$, uniformly converges to $F(\theta) = \sum_{z\in S} \mathcal{P}(\theta|f(z), g)$.*

Proof Let us start by taking the supremum norm

$$\begin{aligned}
||F(\theta) - F_i(\theta)||_\infty &= \sup_{\theta\in\Theta} \Big\{ \Big| F(\theta) - F_i(\theta) \Big| \Big\} \\
&= \sup_{\theta\in\Theta} \Big\{ \Big| \sum_{z\in S} \mathcal{P}(\theta|f(z), g) - \sum_{z\in S_i} \mathcal{P}(\theta|f(z), g) \Big| \Big\} \\
&= \sup_{\theta\in\Theta} \Big\{ \Big| \sum_{z\in S\setminus S_i} \mathcal{P}(\theta|f(z), g) \Big| \Big\}
\end{aligned}$$

1. https://www.bookofproofs.org/branches/supremum-norm-and-uniform-convergence/proof/

Since $\mathcal{P}(\theta|f(z),g)$ is a probability function, it holds that $0 \leq \mathcal{P}(\theta|f(z),g) \leq 1$. Hence, the norm is bounded by the equation below

$$\begin{aligned} ||F(\theta) - F_i(\theta)||_\infty &= \sup_{\theta\in\Theta} \Big\{ \Big| \sum_{z\in S\setminus S_i} \mathcal{P}(\theta|f(z),g) \Big| \Big\} \\ &\leq \sup_{\theta\in\Theta} \Big\{ \sum_{z\in S\setminus S_i} \Big| \mathcal{P}(\theta|f(z),g) \Big| \Big\} \leq \Big| S \setminus S_i \Big| \end{aligned}$$

for $\Big|S \setminus S_i\Big|$ the cardinality of the set $S \setminus S_i$. As discussed before, by definition S_i converges to S for large values of i. According to the proposition above, therefore, we can prove that

$$||F(\theta) - F_i(\theta)||_\infty \xrightarrow{i\to\infty} 0$$

From this proof, it follows that when i approaches infinity the function $F_i(\theta)$ uniformly converges to $F(\theta)$

$$||F(\theta) - F_i(\theta)||_\infty \xrightarrow{i\to\infty} 0 \implies F_i(\theta) \rightrightarrows F(\theta) \tag{6}$$

■

As a consequence of this uniform convergence, two properties of the function F naturally arise. Firstly, $F_i(\theta)$ converges point-wise to $F(\theta)$. Secondly, $F(\theta)$ is a continuous function on Θ.

4.2 Parameter convergence

Let us now focus on the convergence of the copy parameters θ. As previously introduced, we can define the optimal copy parameters for a given value of i, and its corresponding function F_i, according to the following equation

$$\theta_i^* = \arg\max_{\theta\in\Theta} F_i(\theta) \tag{7}$$

for Θ the complete parameter set. We assume that this set is only well defined if F_i and F have a unique global maximum. This is a commonly made assumption in the literature. In addition, we also assume that Θ is compact.

From the definition of θ_i^* above it follows that

$$F_i(\theta_i^*) \geq F_i(\theta), \;\; \forall\theta \in \Theta, \; \forall i \in \mathbb{N}$$

and using the point-wise convergence of F_i we obtain

$$F(\hat{\theta}^*) \geq F(\theta), \;\; \forall\theta \in \Theta \tag{8}$$

where $\hat{\theta}^*$ is the limit of the sequence $\left(\theta_i^*\right)_i$. As a consequence of the compactness of Θ and the continuity of F, we can conclude that $\hat{\theta}^*$ must exist. Moreover, given our assumption that Θ is *well defined*, we can conclude that

$$\hat{\theta}^* = \theta^* = \arg\max_{\theta} F(\theta). \tag{9}$$

The proof above shows that we can approximate the true optimal parameters by sequentially estimating their value using a sequence of subsets S_i that uniformly converge to S. In other words, it demonstrates the feasibility of the sequential approach. This is a theoretical feasibility. Now, we discuss how this approach can be implemented in practice, by considering different optimizations that ensure that the process can be conducted within reasonable computational and memory requirements.

4.3 Implementation

In what follows, we bridge the gap between theory and practice by introducing the implementation details required to solve the previously described optimization problem following a sequential approach. For this purpose, this section introduces intuitions, as well as practical optimizations to build sequential copies. We begin by discussing how the sequence of subsets S_i can be constructed.

4.4 Building S_i

In the sequential approach, the optimal parameter values are approximated sequentially by updating their value according to a sequence of subsets $S_i \subseteq S_{i+1} \subseteq \cdots \subseteq S$. The first step is therefore generating the subsets so as to ensure a uniform convergence of the sequence.

Let us assume P_Z, an arbitrary generating probability distribution defined over the set S. We use this distribution to draw new independent samples $z \in S$. In particular, at each step in i of the sequential process we can use P_Z to generate N new samples such that

$$
\begin{aligned}
S_0 &= \left\{\emptyset\right\} \\
S_1 &= S_0 \cup \left\{z_1^1, \ldots, z_{N-1}^1, z_N^1\right\} \\
S_i &= S_{i-1} \cup \left\{z_1^i, \ldots, z_{N-1}^i, z_N^i\right\} = \left\{z_j^k\right\}_{j=1,\ldots,N}^{k=1,\ldots,i}
\end{aligned}
$$

where the condition that $S_{i-1} \subseteq S_i$ holds for all $i \geq 0$. The cardinality of the set S_i is then given by $\left|S_i\right| = i \cdot N$ and grows linearly with i. This approach ensures that in the limit where $i \longrightarrow \infty$ we recover the set S, since P_Z is defined over this set.

The procedure then works as follows. Once having generated the set S_i at each step we can label them using our objective function $f(\cdot)$ and obtain our best estimation to our objective function $g(\cdot, \theta_i^*)$ using information about the $i \cdot N$ elements considered for that iteration. Of course, this estimation can only be partial, since S_i only provides a partial access to the form of f over S. Of course, the larger the size of N the better our approximation to the true optimal will be, However, as discussed above for the single-pass approach, the bigger the cardinality of S_i the more costly it will be to handle in practice.

At this point it should already become clear to the reader that the number of points N drawn at each iteration plays a key role in the copying process. In addition, there's also the issue of selecting an appropriate generating probability distribution P_Z.

The latter involves having having access to the original training data distribution, or at least a certain intuition about its form. For starters, we need to define an attribute representation $\mathbb{R}^d$. This is a reasonable assumption, since we need to have an approximate

idea of the data points range to build meaningful synthetic samples. Secondly, another important issue is that of volume imbalance, which arises when one or more of the classes occupy a region of the space much smaller than the rest. These two issues, among others, should be understood and dealt with when selecting P_Z. In this work, however, we assume P_Z to take the form of an arbitrary probability distribution, a normal distribution for example, and generate the synthetic samples accordingly. We focus instead on developing heuristics to reduce the number of points $i \cdot N$ considered at each step. We choose to focus on this issue because the size of the different S_i has a direct impact on the computational resources required for copying.

To reduce the cardinality of the different subsets, we hypothesize that there exists an optimal set $S_i^* \subseteq S_i$ from which we can build a copy $g(z, \theta_i^*)$ where $z \in S_i^*$ that completely recovers the behavior of $f(\cdot)$ over S_i. This is, we hypothesize that there exists a lower cardinality set that S_i^* that can be used to carry information about the points in S_i to the next iteration. In this regard, we remind the reader that the copy is build sequentially by obtaining the best estimation of θ^* at each step and then adding an additional $i \cdot N$ data points at the next. Our hypothesis is therefore that the number of points in S_i that evolve to the next iteration $i+1$ can be reduced to a smaller subset S_i^*. In other words, that the cardinality of the set $\left|S_i\right|$ can be largely reduced at each iteration.

This hypothesis is based on the observation that learning in machine learning settings can be understood as a form of data compression. Most state-of-the-art machine learning models internally compress the data points they are trained with by memorizing them inside the model parameters. In the sequential setting, this means that a substantial part of the information contained in S_i is carried by θ_i^* to the next iteration. It is the additional information, which the copy has not memorized, that we need to store in S_i^* for the next iteration.

Testing this hypothesis in practice can be challenging. Here, we propose to use a *measure* about how well each data point, z_j^k, is compressed by the copy model g at each step. We can then use this measure to leave aside those points the copy has already learnt for the next iteration.

4.5 Introducing copy confidence

We refer the reader to Eqs. 3.1 and 3.1 to recall that both the original model f and the copy g are defined such that they return a n_c-vector. In particular, we define the model f to return a vector of hard labels and the copy g to return a vector of class probabilities. We introduce the *confidence* of the copy for a given point z as the normalized euclidean norm of the distance between the vectors output by model and copy for the considered point. We can compute this confidence value according to the following expression

$$\rho(z, \theta) = \frac{\left|g(z, \theta) - f(z)\right|}{\sqrt{n_c}} \in [0, 1] \tag{10}$$

A small value of ρ indicates that the copy has a strong confidence in the class prediction output for z, and that, in turn, z is properly classified. On the contrary, a large value of ρ indicates that while the copy model might have a strong confidence in the class predicted for z, this prediction is incorrect. Based on this definition, ρ can be interpreted as a measure

of how well the model g has learned the point z. A value of 0 means that g has perfectly learned the considered point, whereas a value of 1 corresponds to cases where z has been wrongly learned.

The parameter ρ as defined above is point-dependent. We can extend its definition to the sequential framework assuming. To do so, we assume the existence of the set S_{i-1} and the corresponding optimal parameter value estimation θ^*_{i-1}. Then, we can use these values to build S_i as

$$S_i = S_{i-1} \cup \left\{ z^i_1, \ldots, z^i_{N-1}, z^i_N \right\} = \left\{ z^k_j \right\}^{k=1,\ldots,i}_{j=1,\ldots,N}$$

and rewrite Eq. 10 as

$$\rho_i(z^k_j, \theta^*_{i-1}) = \frac{\left| g(z^k_j, \theta^*_{i-1}) - f(z^k_j) \right|}{\sqrt{n_c}}. \tag{11}$$

This formulation for ρ_i gives us information about how well the data points in S_i are compressed by the copy $g(\theta^*_i)$. We can use this confidence value as reference to decide whether to keep or drop these points by assigning them a weight accordingly. Before assigning these weights, however, we need to ensure that our previous proof about the convergence of $F(\theta)$ and θ^* holds in the presence of weighted data. In what follows, we show that adding a weight to each data point in S_i does

4.6 Proof of convergence with weighted data

Let us introduce confidence values for each data point at each step i as follows

$$z^k_j \longrightarrow \beta_i(z^k_j, \theta^*_{i-1}) = 1 - \rho_i(z^k_j, \theta^*_{i-1}) \tag{12}$$

Note that the weights β are introduced as a complementary value of the confidence ρ for each data point. This definition ensures that we give a higher weight to those samples that have been better learned by the copy. Introducing these weights for each data point we can rewrite, Eq. 4 as

$$F_i(\theta) = \sum_{z \in S_i} \beta_i(z, \theta^*_{i-1}) \mathcal{P}(\theta | f(z), g). \tag{13}$$

In what follows we study the convergence of F under the new form above. Fortunately, we know that $\beta_i(z)$ must converge to 1 as $i \to \infty$, since the copy converges to the original model. Then, the same limit function can be used to study the convergence of F_i

$$F(\theta) = \sum_{z \in S} \mathcal{P}(\theta | f(z), g)$$

and the convergence $F_i(\theta) \rightrightarrows F(\theta)$ can be proved by a simple modification of theorem 1

Theorem 3 *A sequence of functions $\left\{F_i\right\}_i$ defined as $F_i(\theta) = \sum_{z \in S_i} \beta_i(z, \theta^*_{i-1}) \mathcal{P}(\theta | f(z), g)$, uniformly converges to $F(\theta) = \sum_{z \in S} \mathcal{P}(\theta | f(z), g)$.*

Proof Let us start by taking the supremum norm

$$\begin{aligned}||F(\theta) - F_i(\theta)||_\infty &= \sup_{\theta\in\Theta}\Big\{\Big|F(\theta) - F_i(\theta)\Big|\Big\} \\ &= \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S}\mathcal{P}(\theta|f(z),g) - \sum_{z\in S_i}\beta_i(z,\theta^*_{i-1})\mathcal{P}(\theta|f(z),g)\Big|\Big\} \\ &= \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S\setminus S_i}\mathcal{P}(\theta|f(z),g) + \sum_{z\in S_i}\rho_i(z,\theta^*_{i-1})\mathcal{P}(\theta|f(z),g)\Big|\Big\}\end{aligned}$$

Using the triangular inequality we obtain

$$\begin{aligned}||F(\theta) - F_i(\theta)||_\infty &= \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S\setminus S_i}\mathcal{P}(\theta|f(z),g) + \sum_{z\in S_i}\rho_i(z,\theta^*_{i-1})\mathcal{P}(\theta|f(z),g)\Big|\Big\} \\ &\leq \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S\setminus S_i}\mathcal{P}(\theta|f(z),g)\Big| + \Big|\sum_{z\in S_i}\rho_i(z,\theta^*_{i-1})\mathcal{P}(\theta|f(z),g)\Big|\Big\} \\ &\leq \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S\setminus S_i}\mathcal{P}(\theta|f(z),g)\Big|\Big\} + \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S_i}\rho_i(z,\theta^*_{i-1})\mathcal{P}(\theta|f(z),g)\Big|\Big\}\end{aligned}$$

Proceeding as we did in the original proof, we can bound the supremum norm by

$$||F(\theta) - F_i(\theta)||_\infty \leq \Big|S\setminus S_i\Big| + \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S_i}\rho_i(z,\theta^*_{i-1})\mathcal{P}(\theta|f(z),g)\Big|\Big\}$$

for $\Big|S\setminus S_i\Big|$ the cardinality of the set $S\setminus S_i$. By definition, S_i converges to S for large values of i while $\rho_i \to 0$. According to the proposition we previously prove, as we have show that

$$||F(\theta) - F_i(\theta)||_\infty \xrightarrow{i\to\infty} 0$$

it follows that when i approaches infinity the function $F_i(\theta)$ uniformly converges to $F(\theta)$

$$||F(\theta) - F_i(\theta)||_\infty \xrightarrow{i\to\infty} 0 \implies F_i(\theta) \rightrightarrows F(\theta) \tag{14}$$

■

Having proved the converged of our sequence in the presence of weighted data, we now provide an intuitive interpretation for the parameter β in Eq. 13.

To do so, let us assume that we are at step i and that $g(\cdot,\theta^*_{i-1})$ is the best approximation to the original model considering the points in S_{i-1} in the previous iteration. At the current step i, we feed the copy a new set of N points through S_i. Let us further assume that the copy has a total confidence on its prediction for all the data points in this set except for one, for which it fails to correctly predict the class label. In this situation we can write

$$\rho(z^k_j,\theta^*_{i-1}) = \begin{cases}\rho_{fail} & \text{if } z^k_j = z_{fail} \\ 0 & \text{other}\end{cases} \implies \beta(z^k_j,\theta^*_{i-1}) = \begin{cases}\beta_{fail} & \text{if } z^k_j = z_{fail} \\ 1 & \text{other}\end{cases} \tag{15}$$

which, at step i gives us

$$F_i(\theta) = \sum_{z \neq z_{fail}} 1 \cdot \mathcal{P}(\theta|f(z), g) + \beta_{fail} \cdot \mathcal{P}(\theta|f(z_{fail}), g) \tag{16}$$

which is the function that needs to be maximized to obtain parameters θ. However, since the model has learned almost all the points with a perfect confidence, the value $\mathcal{P}(\theta|f(z), g)$ in the first term turns to be equals to 1

$$\mathcal{P}(\theta^*_{i-1}|f(z), g) = 1$$

for the optimal parameter in the previous step, so the only way to improve the result, this is, to get an even higher value for the sum, is to find a new optimal parameter that conserves the probabilities of 1 for all the learned points. This is because those points with a larger weight and a slightly decrease on their probability can dramatically affect the value of the total sum. In consequence, the copy can learn new data points but never at the expense of loss knowledge on the points it has already learned.

5. Experiments

This section provides an intuitive understanding of how the sequential approach works in practice on both a set of toy problems and on higher dimensional real-world datasets. To this aim, we describe the heuristics to sequentially approximate copy parameters using the optimizations described above and then report the results by comparing them with the obtained using the single-pass method. We begin by describing a version of the sequential approach in practice for creating a copy using a neural network.

5.1 A first approach to practice: linear training set growth

Let us assume the existence of a model f trained on dataset $\mathcal{D}$ to which we have no access, other than through a membership query interface that provides hard prediction outputs. We train a copy $f_{\mathcal{C}}$ by sequentially approximating the decision boundary learned by f as follows:

1. At each step i, we generate a set synthetic set S_i by randomly sampling N data points from a given probability distribution P_Z. For simplicity purposes, we consider P_Z to follow a normal distribution.

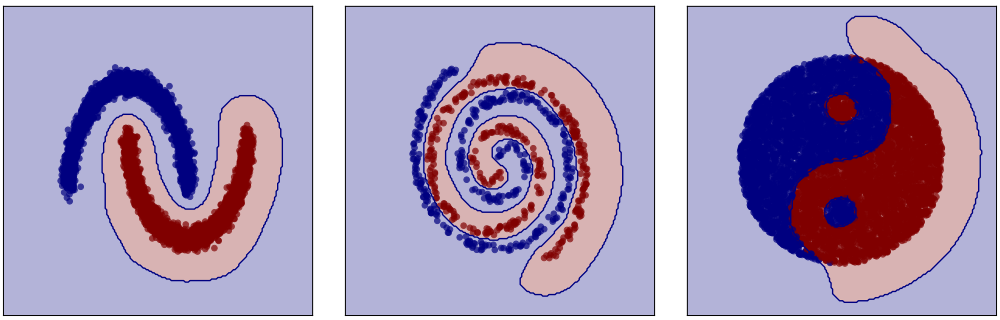
2. We label this set according to the hard prediction labels output by the model f for the different data points $z^i_j \in S_i$.

3. We use the labelled set to obtain an intermediate estimation of the optimal copy parameters θ^*_i.

Whenever we move to the next iteration, we generate a new set of N synthetic data points and add them to the already existing ones, such that $S_{i+1} = S_i \cup z^{i+1}_1, ..., z^{i+1}_N$. In this simple approach, the number of samples used for training grows linearly with the number of steps. This process is stopped when we reach the desired number of iterations or when the reported error is below a given threshold.

5.2 Solving toy problems with the linear training set growth

We assay this approach in three toy binary classification datasets. We learn these datasets using Gaussian kernel SVMs. In all cases, the models achieve perfect accuracy. We show the achieved decision boundaries in Figure 1. We copy these models using the linear training set growth employing a fully-connected neural network with three hidden *relu* layers and a softmax output. We sequentially train the neural network during 20 steps using the Adam optimizer and binary cross-entropy as our loss function. At each step we use 100 epochs and a batch size of 10 samples. Figures 7, 8 and 9 in the Appendix A show the decision functions learnt by instantaneous copy models for steps 1, 5, 10, 15 and 20 for different values of N for the *moons*, *spirals* and *yin-yang* datasets, respectively.



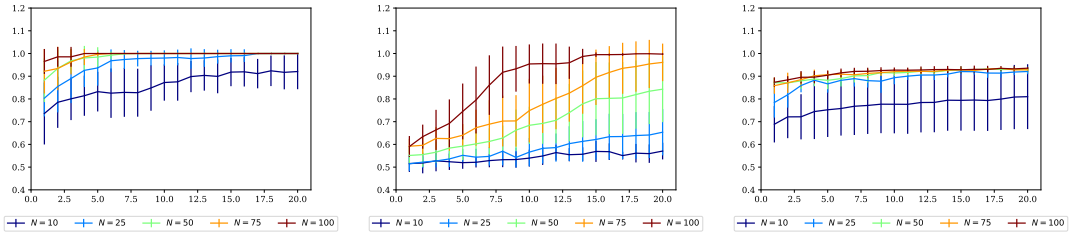
(a) (b) (c)

Figure 1: Decision boundaries learned by a gaussian kernel SVM for the (a) *moons*, (b) *spirals* and (c) *yin-yang* datasets. The first and third datasets include 10000 data points, while the second includes 500 samples.

In all cases, we observe that when we increase the number of steps, the fidelity of the copy increases. Note that in the current setting, the copy has access to new N synthetic data points at each iteration, meaning that weights are updated based on both the new and previous samples. In the final step, copies are trained using $20N$ samples. Note that the copy weights are randomly initialized at the beginning and sequentially updated at each step i starting from the values obtained at the end of the previous step $i-1$.

An additional observation is that every dataset has a different threshold for the parameter N. For instance, copies trained on the *moons* datasets with only $N = 25$ are already able to recover most of the original decision behaviour by step 10. However, for the *spirals* dataset, the original decision boundary is only captured when N is larger than 75. Finally, for *yin-yang* dataset, only iterations with $N = 100$ display the expected behaviour. A primary conclusion is that the number of new data points required to train sequential copies is highly dependent on the complexity of the considered decision boundary. However, as observed in this preliminary toy example, it is possible to obtain high-performance copy models with few samples for the three considered datasets.

To better compare the effect of sample size, we report the average copy accuracy for the different values of N and an increasing number of steps. Figure 2 shows the results for all three problems.



(b) (c)

Figure 2: Copy accuracy averaged over 10 independent runs for varying values of N for the (a) *moons*, (b) *spirals* and (c) *yin-yang* datasets. Error bars correspond to one standard deviation.

This section provides an intuitive understanding of how the sequential approach work in practice on a set of toy problems. We first describe the heuristics to sequentially approximate copy parameters using the optimizations above and then report our preliminary results and compare them with those obtained using the single-pass approach.

5.3 A second approach to practice: dropping training samples

To reduce the required training samples, we improve the previous approach by using the sample weights described in Section 4 to drop the unnecessary previous synthetic training samples. To include this improvement, we modify the previous approach as follows:

- At each step i we generate a set S_i by randomly sampling N data points from a given probability distribution P_Z. For simplicity purposes, we consider P_Z to follow a normal distribution
- We label this set according to the predictions of the model f for the different data points z.
- We used the labelled set to obtain an intermediate estimation of the optimal copy parameters θ_i^* by fitting the copy model to the synthetic data
- Before moving on to the next step $i+1$, we compress the S_i as follows: we compute the weights of all the samples $z \in S_i$ by querying the copy and calculating their confidence scores according to Eq.11. We compute the weights by taking the complementary of these values. At this point, we introduce a new parameter h that act as a threshold to decide which points to keep for the next iteration and which ones to drop. In other words, we build a compressed set S_i^* that includes only those points in S_i for which $\beta < h$.
- We move to next iteration and generated a new set of N synthetic data points, such that $S_{i+1} = S_i^* \cup z_1^{i+1}, ..., z_N^{i+1}$.

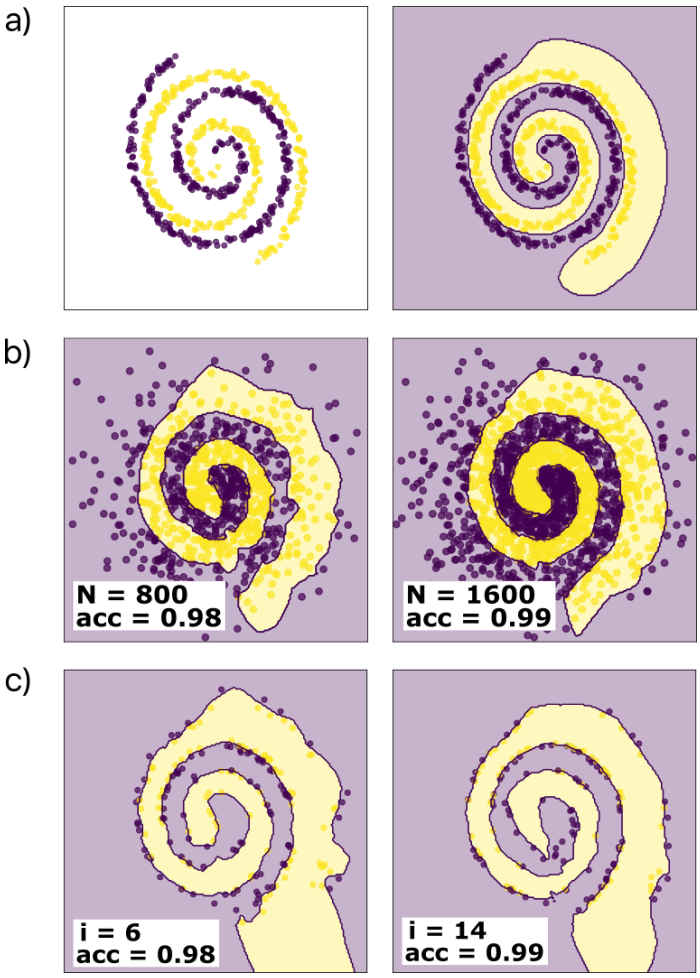


Figure 3: **Spirals dataset: summary.** a) Spiral dataset and the original model. b) Copy model for the *single-pass* approach for two different set sizes: N=800, 16000. c) Copy model for the *sequential* approach. Each image corresponds to a different step.

5.4 Solving toy problems with dropping training samples

We consider a binary classification problem in 2D. In particular, we use the dataset shown on the left side of Fig. 3a). We use this dataset to train a Gaussian kernel SVM. The right side of Fig. 3a) depicts the resulting decision boundary. Note that this model achieves maximum accuracy over the considered set. We copy this model using the same network architecture described in Section 5.2.

Initially, we assay the single-pass approach to train our copy with a fixed number of points. In particular, Figure 3b) shows the resulting copy decision boundaries for 800 and 1600 synthetic data points, respectively. In both cases, the resulting copies replicate the original model's behaviour with high accuracy (≈ 0.99).

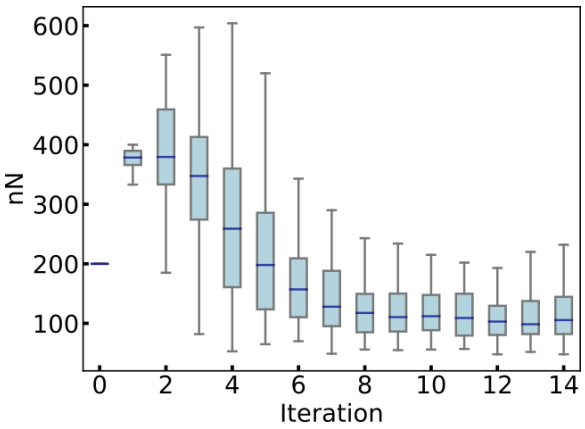


Figure 4: **Spirals: Number of points.** Average over 50 runs of the number of points used at each iteration in the sequential mode. The number of points shows a peak at the second step and then is drastically reduced.

Then, we train the same ANN sequentially as discussed in the previous section. We set the initial value of N to 200. We choose a value for the threshold h such that at the end of each iteration, we remove all the samples with $\rho(z) \leq 5 \cdot 10^{-5}$. Figure 3c) shows the resulting decision boundary for iterations 6 and 14 respectively. In addition, Fig. 4 shows the average number of data points used at each iteration. Note that this value increases at the initial steps when the copy weights are not fine-tuned for the given decision boundary. However, after a few iterations, the number of required data points decreased and stabilises around 150 points. Initially, we assay the single-pass approach to train our copy with a fixed number of points. In particular, Figure 3b) shows the resulting copy decision boundaries for 800 and 1600 synthetic data points, respectively. In both cases, the resulting copies replicate the original model's behaviour with high accuracy (≈ 0.99).

5.5 Considering multi-class datasets

In this subsection, we confront our sequential approach with real datasets with different number of attributes and instances and three labels extracted from the UCI data repository. Table 1 shows a brief summary of the used databases. See the *UCI dataset webpage* for additional information.

	Instances	Attributes	Classes
iris	150	4	3
wine	178	13	3

Table 1: Numeric description of the UCI databases used.

For this experiment, we use the sequential approach with dropping training samples. Initially, we train a k-nearest neighbours (k-NN) classifier as the original model to be replaced by a copy. The original k-NN classifier was trained with the default parameters of

the SciKit-learn Python library. Copies are fed with $N = 15$ new synthetic points at each iteration and the threshold to drop data points is 10^{-5}. Figure 5 shows the average value for the accuracy and the number of points at each iteration of 50 independent runs for the *iris* and *wine* datasets, respectively.

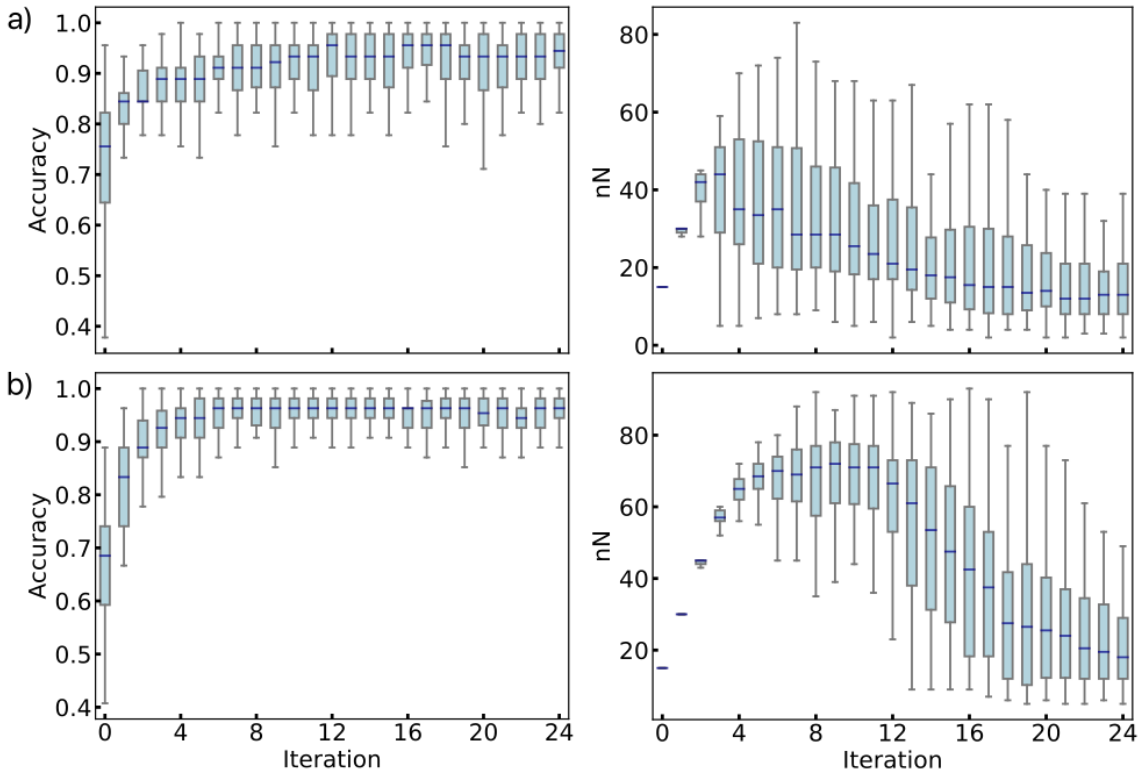


Figure 5: **UCI datasets.** a) Average values for the accuracy and number of points per iteration for the *iris* dataset. b) Average values for the accuracy and number of points per iteration for the *wine* dataset.

Even though both datasets do not present a complex attribute distribution, the copying accuracy in both cases is near 100% after a few training iterations. However, in both datasets, we can observe peaks in the required number of points in different time moments in the training phase. These peaks are related to the complexity of the datasets. From these plots, we can guess the following empirical idea: the fewer points required to train a copy, the easier the model to copy.

Observing the difference in the number of training points per iteration, we see that the copy needs to store more points during some iterations for the wine dataset. This additional memory is because the iris dataset outcomes are linearly separable while the wine decision boundary is not linear. Non-linearities require the copy to store some extra points during the training phase to allow the model to learn them.

5.6 Studing the high dimensional datasets

The previous UCI datasets had few dimensions allowing us to focus on the sequential learning process without worrying about the dimensionality issues related to the sampling. In this experiment, we use high dimensionality data, the Forest CoverType dataset, downloaded from the UCI data repository. The dataset contains 58,1012 samples, 54 attributes and seven possible output labels. We have used the same approach as before: the sequential approach with dropping and a k-nearest neighbours (k-NN) classifier as the original model.

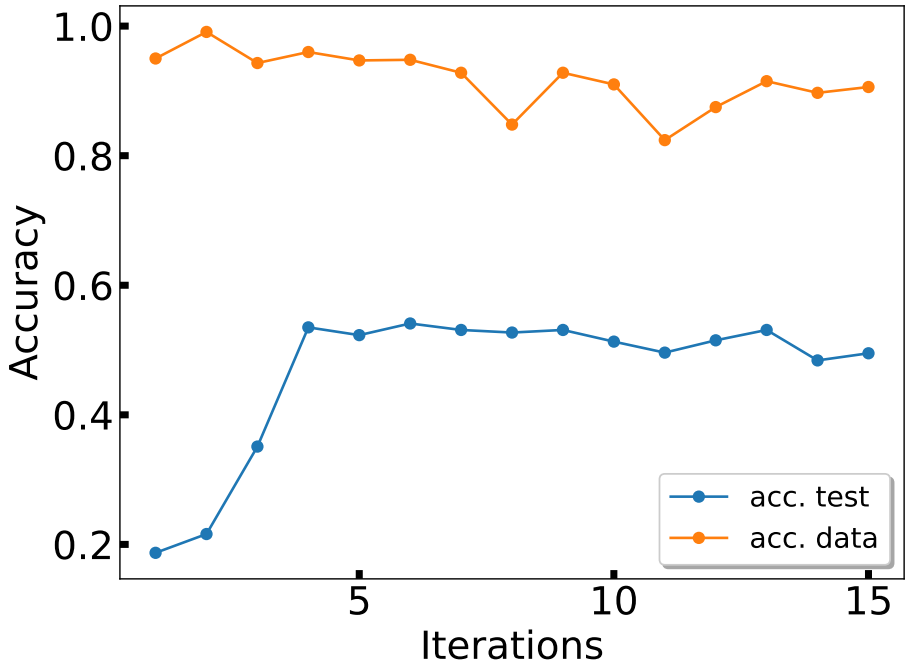


Figure 6: **Covertype dataset.** Comparing train and test accuracy per iteration.

As depicted in Figure 6, the achieved accuracy in each iteration is close to 100%. However, when observing the classification performance with the test data, it drops to near 60% over the seven possible outcomes. The low behaviour in the test data, similar to the classical overfitting problem of any machine learning problem, can be produced for several reasons, such as an inappropriate sampling method, an imbalanced space distribution of some outcomes, or a poor performance original model.

6. Conclusions

In this paper, we propose and validate a sequential approach to copy machine learning classifiers. We derived the theory for copying a machine learning classifier in a sequential manner, highlighting its differences from the single-pass technique, which requires generating and storing a massive synthetic database in a computer when the decision boundary of the classifier to copy is complex and intricate.

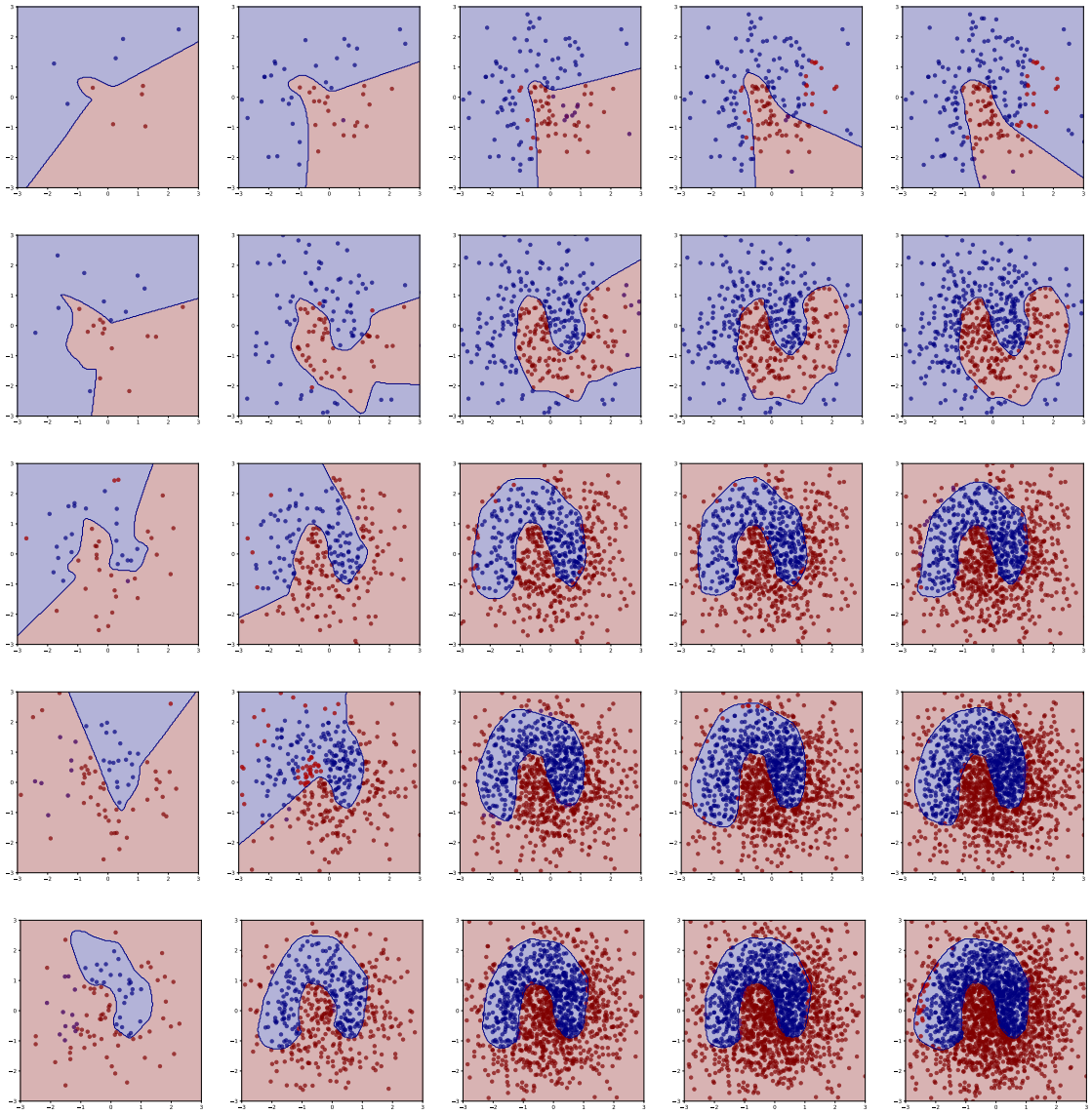
Besides, our experiments demonstrate that the sequential approach to copies is feasible. There are a few conclusions that we can extract from the described results. Firstly, the sequential approach achieves a comparable level of accuracy to that obtained using the single-pass method. It does so using a smaller amount of memory at each iteration. Moreover, these results also show that removing data points based on the level of confidence of the copy is a viable heuristic to ensure the necessary amount of information is carried from iteration to iteration.

In our future work, we want to study in depth the sampling problem in the copying framework to reduce the number of required iterations to copy a machine learning model.

Acknowledgments

This work was funded by the Huawei Technologies Duesseldorf GmbH under project TC20210924032.

Appendix A. Toy problems instantaneous decision boundaries



(u) (v) (w) (x) (y)

Figure 7: From top to bottom, instantaneous decision boundaries for $N = 10$, $N = 25$, $N = 50$, $N = 75$ and $N = 100$ at steps (a) 1, (b) 5, (c) 10, (d) 15 and (e) 20 for the *moons* dataset.

References

Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpreting black-box models via model extraction. arXiv:1705.08504, 2018. [Online] Available from: `https://arxiv.org/pdf/1705.08504.pdf`.

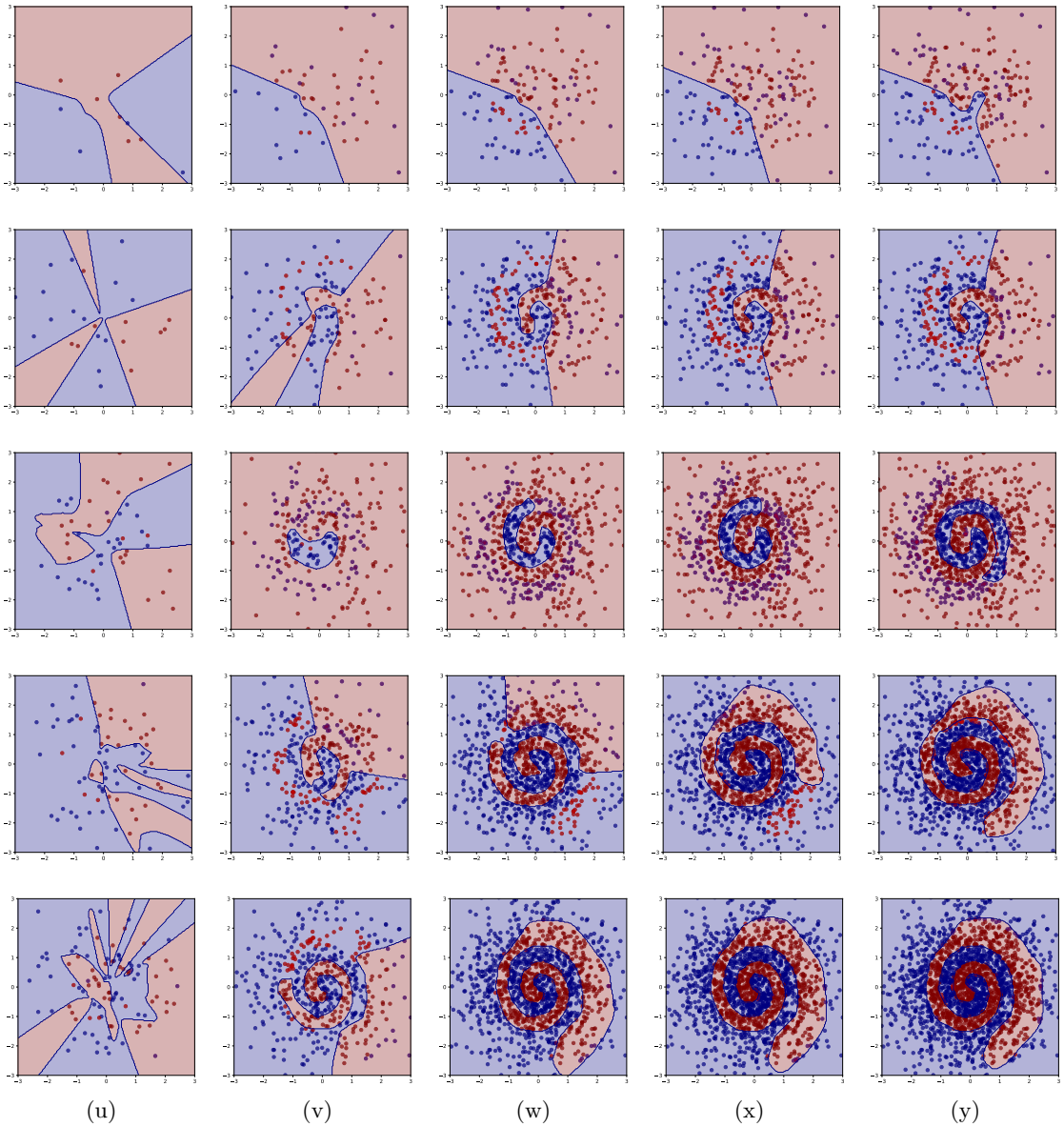
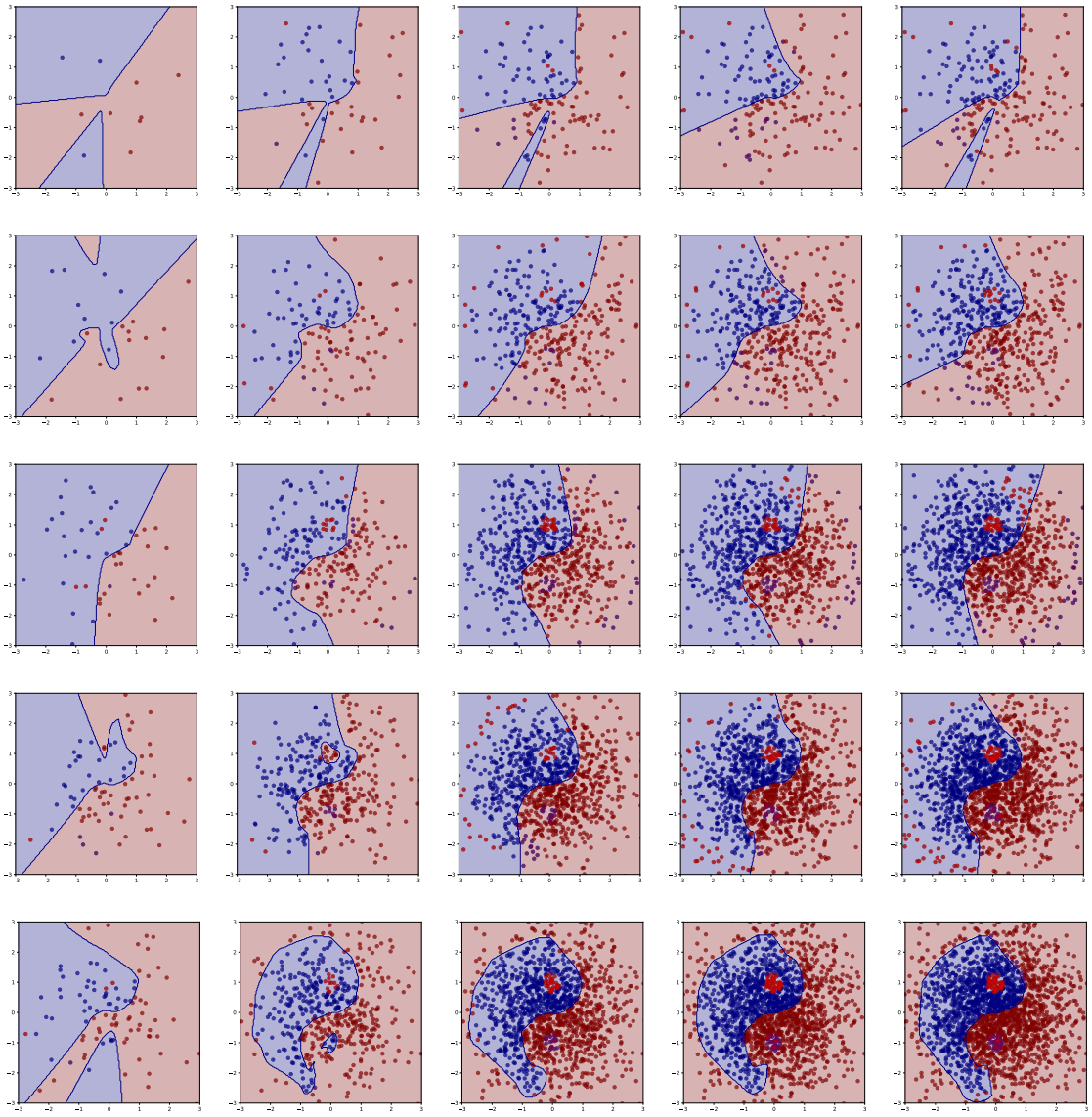


Figure 8: From top to bottom, instantaneous decision boundaries for $N = 10$, $N = 25$, $N = 50$, $N = 75$ and $N = 100$ at steps (a) 1, (b) 5, (c) 10, (d) 15 and (e) 20 for the *spirals* dataset.

Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, 2014. doi: 10.1109/TNNLS.2013.2293637.

Guido Bologna and Yoichi Hayashi. A comparison study on rule extraction from neural network ensembles, boosted shallow trees, and svms. *Applied Computational Intelligence*



(u) (v) (w) (x) (y)

Figure 9: From top to bottom, instantaneous decision boundaries for $N = 10$, $N = 25$, $N = 50$, $N = 75$ and $N = 100$ at steps (a) 1, (b) 5, (c) 10, (d) 15 and (e) 20 for the *yin-yang* dataset.

and Soft Computing, pages 1–20, 2018. doi: 10.1155/2018/4084850.

Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001. doi: 10.1214/ss/1009213726.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 535–541, Philadelphia, PA, USA, 2006. Association for Computing Machinery, New York, NY, USA. ISBN 1595933395. URL https://doi.org/10.1145/1150402.1150464. doi: 10.1145/1150402.1150464.

Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. Interpretable deep models for icu outcome prediction. In *Proceedings of the AMIA Annual Symposium*, pages 371–380, Chicago, IL, USA, 2016.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989. doi: 10.1007/BF02551274.

Nilesh Dalvi, Pedro Domingos, Mausam Sumit, and Sanghai Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 99–108, Seattle, WA, USA, 2004. Association for Computing Machinery, New York, NY, USA. ISBN 1581138881. doi: 10.1145/1014052.1014066.

Y.N. Dauphin and Y. Bengio. Big neural networks waste capacity. arXiv:1301.3583, 2013. [Online] Available from: https://arxiv.org/pdf/1301.3583.pdf.

D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. Understanding deep architectures using a recursive convolutional network. arXiv:1312.1847, 2013. [Online] Available from: https://arxiv.org/pdf/1312.1847.pdf.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

Tianzhen Hong, Zhe Wang, Xuan Luo, and Wanni Zhang. State-of-the-art on research and applications of machine learning in the building life cycle. *Energy and Buildings*, 212: 109831, 2020. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2020.109831. URL https://www.sciencedirect.com/science/article/pii/S0378778819337879.

Ruishan Liu, Nicolo Fusi, and Lester Mackey. Teacher-student compression with generative adversarial networks. arXiv:1812.02271, 2018a. [Online] Available from: https://arxiv.org/pdf/1812.02271.pdf.

Ruishan Liu, Nicolo Fusi, and Lester Mackey. Teacher-student compression with generative adversarial networks. arXiv:1812.02271, 2018b. [Online] Available from: https://arxiv.org/pdf/1812.02271.pdf.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, page 641–647, Chicago, IL, USA, 2005. Association for Computing Machinery, New York, NY, USA. ISBN 159593135X. doi: 10.1145/1081870.1081950.

Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. Challenges in deploying machine learning: A survey of case studies. *ACM Computing Surveys*, apr 2022. ISSN 0360-0300. doi: 10.1145/3533378. URL https://doi.org/10.1145/3533378.

Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxkijC5FQ.

Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proc. VLDB Endow.*, 11(12):1781–1794, aug 2018. ISSN 2150-8097. doi: 10.14778/3229863.3229867. URL https://doi.org/10.14778/3229863.3229867.

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf.

Irene Unceta, Jordi Nin, and Oriol Pujol. Copying machine learning classifiers. *IEEE Access*, 8(11):160268–160284, 2020a. doi = 10.1109/ACCESS.2020.3020638.

Irene Unceta, Jordi Nin, and Oriol Pujol. Environmental adaptation and differential replication in machine learning. *Entropy*, 22(10), 2020b. doi = 10.3390/e22101122.

Dongdong Wang, Yandong Li, Liqiang Wang, and Boqing Gong. Neural networks are more productive teachers than human raters:active mixup for data-efficient knowledge distillation from a blackbox model. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1498–1507, Seattle, WV, USA, 2020. IEEE Computer Society, Washington, DC, USA.