
Generative Trees: Adversarial and Copycat

Richard Nock¹ Mathieu Guillaume-Bert¹

Abstract

While Generative Adversarial Networks (GANs) achieve spectacular results on unstructured data like images, there is still a gap on *tabular data*, data for which state of the art *supervised learning* still favours decision tree (DT)-based models. This paper proposes a new path forward for the generation of tabular data, exploiting decades-old understanding of the supervised task's best components for DT induction, from losses (properness), models (tree-based) to algorithms (boosting). The *properness* condition on the supervised loss – which postulates the optimality of Bayes rule – leads us to a variational GAN-style loss formulation which is *tight* when discriminators meet a calibration property trivially satisfied by DTs, and, under common assumptions about the supervised loss, yields "one loss to train against them all" for the generator: the χ^2 . We then introduce tree-based generative models, *generative trees* (GTs), meant to mirror on the generative side the good properties of DTs for classifying tabular data, with a boosting-compliant *adversarial* training algorithm for GTs. We also introduce *copycat training*, in which the generator copies at run time the underlying tree (graph) of the discriminator DT and completes it for the hardest discriminative task, with boosting compliant convergence. We provide experiments on tasks including fake/real distinction and missing data imputation.

1. Introduction

Generative Adversarial Networks have early established a gold standard for both neural networks as generative models *and* the loss to train generative models via a variational measure-based distortion (Goodfellow et al., 2014; Nowozin et al., 2016; Nock et al., 2017). While they have achieved spectacular results on a variety of unstructured data (Ni

et al., 2021), the quality of outcomes on *tabular data* is still lagging behind with the sentiment that new approaches are needed (Camino et al., 2020). This is an important problem: recently, tabular data was still representing the most prevalent data type in real world AI (Chui et al., 2018, pp. 15). Interestingly, this chasm separating astonishing generation on unstructured data to suboptimal generation on tabular data mirrors another one, *on the supervised side*, where neural nets can achieve superhuman recognition on unstructured data (Linsley et al., 2021) but require massive amounts of sophistication to compete against standard libraries using decision-tree (DT) based models on tabular data (Arik & Pfister, 2021). DT induction has been perfected over decades, starting with core supervised loss functions known as *proper* (Savage, 1971; Reid & Williamson, 2011), using particularly fit and simple graph-based *tree models* (Breiman et al., 1984; Quinlan, 1993), and culminating with a powerful algorithmic machinery to learn them, *boosting* (Kearns & Mansour, 1996; Friedman et al., 2000; Schapire & Singer, 1998). One would expect that potential generative approaches for tabular data would "mirror" those three key components on the generative side, but to our knowledge, none has been achieved. Such is our objective, and our paper thus contains three main technical contributions:

On losses, the GAN approach formulates the generator's loss from a variational measure-based divergence, unveiling the discriminator's loss (Nowozin et al., 2016). Instead, we start from the *discriminator's* side and a general proper loss, *i.e.* a loss for which Bayes prediction is optimal, which is standard for DT induction since Breiman et al. (1984). We relate the corresponding information (De Groot, 1962) to a GAN-style formulation which provides us with the generator's loss. A difference with GANs' variational formulation is there is *no slack* in the characterisation if the discriminator meets a calibration condition trivially satisfied by DTs: unlike *e.g.* Nowozin et al. (2016, Ineq. (4)), we get identities all the way through. A surprising corollary follows. If the discriminator's *partial losses* meet a property that most popular choices meet, then to minimize the generator's loss, it is *sufficient* to minimize the χ^2 between real and fake data: we get *one* loss to "train generators against them all". This first contribution *is not specific to DTs* as it holds for all calibrated discriminators in the properness framework.

On models, we introduce generative trees (GTs). In the same way as generator and discriminator in GANs include

¹Google Research. Correspondence to: Richard Nock <richardnock@google.com>.

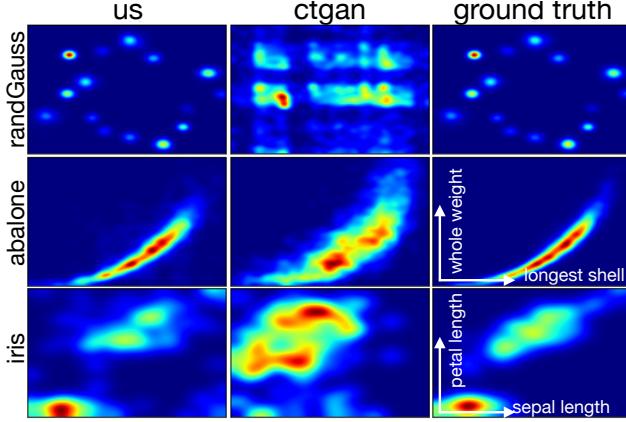


Figure 1. 2D density plots for our generative trees (us) vs CTGANs on 2 UCI + 1 simulated domain. See Section 6 for details. Note how GTs improve on CTGANs.

a similar functional form (a neural net), our GTs include a tree (graph) structure like DTs, differences being stochastic activations at the arcs and leaf-dependent data generation.

On algorithms, we propose a top-down induction algorithm to adversarially train GTs with provable boosting-compliant geometric convergence of the χ^2 , the weak generative learning assumption being a weak statistical dependence between the generator and discriminator. We propose a second way to train generative trees, extremely efficient and that we think has no equivalent yet in neural networks. In this setting, that we nickname *copycat*, the generator tracks and copies the discriminator's tree (graph) at training time, and completes it for the hardest generative model given the discriminator¹. The geometric convergence in density ratio loss of the generator (Menon & Ong, 2016) directly follows from a seminal result of Kearns & Mansour (1996).

From an **experimental** standpoint, we have performed four main series of experiments on missing data imputation, training from generated data, fake/real discrimination and generated data augmentation, against each experiment's state of the art contender (van Buuren & Groothuis-Oudshoorn, 2011; Xu et al., 2019). Figure 1 provides a glimpse of generated data's quality for three domains, against neural nets (CTGANs, Xu et al. (2019)). The domains we used included simulated domains and domains from the UCI, Kaggle and the Stanford Open Policing project. The experiments display that GTs can be very efficient contenders against sophisticated state of the art methods: on training from generated data, fake/real discrimination and generated data augmentation, GTs tend to get better results than neural nets, sometimes with tiny models relative to domain size; on missing data imputation, GTs can beat on low-dimensional problems the mice approach (van Buuren & Groothuis-Oudshoorn, 2011), even when mice relies on tree-based

¹For the informed reader, the generator learns boosting's balanced distribution of Kearns & Mansour (1996).

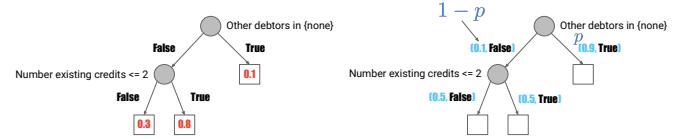


Figure 2. Decision tree (left) and generative tree (right) with the same underlying tree, for the UCI german credit domain.

imputation using thousands+ of tree models – against a GT essentially relying on a *single one*. Experiments are given *in extenso* in an Appendix (App), also containing all proofs.

2. Basic definitions

$\forall k \in \mathbb{N}_*$, we let $[k] \doteq \{1, 2, \dots, k\}$. \mathcal{X} denotes a domain, $\mathcal{S} \doteq \{\mathbf{x}_i : i \in [m]\} \subset \mathcal{X}$ is a sample of real observations. The associated supervised learning problem is a binary labeled problem where labels $\mathcal{Y} \doteq \{-1, 1\} \doteq \{\text{fake}, \text{real}\}$ distinguish between a fake (=generated) and a real observation. The objective of the supervised problem is to learn a *posterior* computing $\mathbb{P}[Y = 1|\mathcal{X}]$, denoted $\eta \in [0, 1]^{\mathcal{X}}$. With slight variations, many notations follow from (Reid & Williamson, 2011). $\pi \doteq \mathbb{P}[Y = 1]$ is the prior. In the generative game, the prior is user-fixed. (\mathcal{X}, P) and (\mathcal{X}, N) are measure spaces for 'positive/real' and 'negative/fake' observations respectively – to avoid notation overloads, we leave implicit the σ -algebra. $(\mathcal{X} \times \{-1, 1\}, D)$ is the product measure space of labeled examples following the (supervised) *binary task* (π, P, N) Reid & Williamson (2011, Section 4); we let $B \doteq (\pi, P, N)$ for short. We also have the *mixture* space (\mathcal{X}, M) with $M \doteq \pi \cdot P + (1 - \pi) \cdot N$. A posterior is particularly interesting for B , *Bayes posterior*, which is:

$$\eta^* = \pi \cdot \frac{dP}{dM}, \quad (1)$$

and is optimal for *proper losses* (more on this in Section 4).

3. Models

We present the tree-based models we use as architectures for both the discriminator and the generator.

Architectures: we start by the commonpoint between both, that we denote a tree for short.

Definition 3.1. A *tree* is a rooted, directed binary tree whose internal nodes are labeled with binary tests over observation variables and outgoing arcs are labeled with truth values. For any internal node, the left outgoing arc is labeled with truth value *false* and the right outgoing arc is labeled with *true*. Leaves are blank nodes.

Definition 3.2. A *decision tree* (DT) is a tree with leaves labeled in $[0, 1]$. A *generative tree* (GT) is a tree in which truth values at arcs are associated to Bernoulli events $B(\cdot)$.

Figure 2 presents examples of DT and GT with the same underlying tree. We assume without loss of generality that

trees are binary but our definitions could trivially be extended to trees of any arity. Hereafter, low caps like h are used to represent DTs while high-caps like G are used to represent GTs. $\Lambda(\cdot)$ denotes the set of leaves of a tree.

Access routines: an important routine needed for a DT h is, for any observation x , the leaf $\lambda(x) \in \Lambda(h)$ reached by x . This is the leaf whose path from the root involves tests satisfied by x . If x contains no unknown feature values, this path is unique. The main access routine for a GT G is the generation of an observation. To do so, we simply stochastically traverse the tree using the Bernoulli events at the internal nodes. Once a leaf $\lambda \in \Lambda(G)$ is reached, sampling an observation is done by a uniform sampling in the complete domain *that satisfies the tests traversed* to reach λ . In Figure 2, the center leaf λ of the GT G is reached with probability $q = 0.1 \cdot 0.5 = 0.05$. If we do reach it, according to the UCI german credit data domain (Dua & Graff, 2021), then we sample uniformly at random an observation for which attribute 'Number existing credits' is in $\{0, 1, 2\}$ and 'Other debtors' is in $\{\text{co-applicant, guarantor}\}$ and all other attributes are chosen uniformly at random in their full domain, since they do not appear in the path to λ .

Remark 3.3. Uniform sampling imposes a finite length domain for real or integer features, which is a reasonable assumption for standard features like e.g. age, salary. Alleviating the constraint can be done using specific transformations, such as the Box-Muller transform, generating Normal deviates from uniform distributions (Box & Muller, 1958).

4. Loss functions involved

Departing from (W)GAN-style approaches, we design the losses involved from the discriminator's.

Calibrated posteriors: for any function $f \in \mathbb{R}^X$ and measure Q over measurable space (\mathcal{X}, Ω) , Q_f is the restriction of Q to the sub- σ -algebra Ω_f induced by the level set of f . A similar notation Q_f with identical definition is used in van Erven & Harremoës (2014, Section II). It can be interpreted as the marginal of Q on the subset of events of Ω_f , each of which is a union of events from Ω having the same f -value. We now define a property of a posterior for class probability estimation that shall be fundamental to analyse our losses.

Definition 4.1. Posterior $\tilde{\eta}$ is said calibrated with task B (or just calibrated for short) iff $\tilde{\eta} = \pi \cdot \frac{dP_{\tilde{\eta}}}{dM_{\tilde{\eta}}}$, and we let $B_{\tilde{\eta}} \doteq (\pi, P_{\tilde{\eta}}, N_{\tilde{\eta}})$.

There are three important examples of calibrated posteriors: (i) the constant posterior $\eta_\pi \doteq \pi$ is (the only constant) calibrated (posterior). To see it, it yields $\Omega_{\eta_\pi} = \{\emptyset, \mathcal{X}\}$. The RHS in Def. (4.1) gives $\pi \cdot dP_{\eta_\pi}(\mathcal{X})/dM_{\eta_\pi}(\mathcal{X}) \doteq \pi \cdot \int_{\mathcal{X}} dP / \int_{\mathcal{X}} dM \doteq \pi \cdot 1/1 = \pi$ and we have for this posterior $\mathbb{P}[Y|X] = \mathbb{P}[Y|X \in \mathcal{X}] = \eta_\pi \doteq \pi$; (ii) Bayes posterior η^* is calibrated; it follows from (1); (iii) let h be a DT. Any DT induces a partition of \mathcal{X} at its leaves, $\{\mathcal{X}_\lambda : \lambda \in \Lambda(h)\}$;

Loss	$\ell_{-1}(u)$	$\ell^{\text{DR}}(\rho), \rho \geq 0$	
		Eq.	cvx
Log	$-\log_2(1-u)$	$\log_2\left(1 + \frac{1}{\rho}\right)$	✓ ✓
Square	$\frac{u^2}{2}$	$\frac{1}{2} \cdot \left(\frac{1}{1+\rho}\right)^2$	✓ ✓
Matusita	$\sqrt{\frac{u}{1-u}}$	$\frac{1}{\sqrt{\rho}}$	✓ ✓
Jeffreys	$2 \cdot \left(\log\left(\frac{u}{1-u}\right) + \frac{1}{1-u}\right)$	$2 \cdot \left(\log\left(\frac{1}{\rho}\right) + \frac{1}{\rho} + 1\right)$	✓ ✓
KL	$2 \cdot \left(2\log(1-u) + \frac{u}{1-u}\right)$	$2 \cdot \left(2\log\left(\frac{u}{1+\rho}\right) + \frac{1}{\rho}\right)$	✓ ✓
Normalized* χ^2	$2 \cdot \frac{u^2}{(1-u)^2}$	$\frac{2}{\rho^2}$	✓ ✓

Table 1. Differentiable partial losses for class -1 for symmetric losses, along with their corresponding ℓ^{DR} and its properties (see text; cvx = convex). (*) add +2 to Pearson χ^2 to have $\ell_{-1}(0) = 0$.

suppose without loss of generality that all leaves' predictions are different, and consider $\Omega_h = \{\emptyset\} \cup \{\mathcal{X}_\lambda : \lambda \in \Lambda(h)\}$. Without further correction, *the posterior prediction* at a leaf λ of h is classically computed as the ratio of the total weight of real observations reaching λ over the total weight of observations reaching λ . In mathematical form, we have here $\mathbb{P}[Y|X] = \mathbb{P}[Y|X \text{ reaches } \lambda] = \pi dP_h/dM_h$, which is by definition the prediction of η at λ and shows that the posterior prediction of any DT is calibrated. We note that (i) is a particular case of (iii) when h is reduced to its root, and if the domain \mathcal{X} is finite, then (ii) is a particular case of (iii) for h being *any* complete (finite) DT. Hereafter, a tilda like $\tilde{\eta}$ denotes a calibrated posterior. Notation η denotes any posterior, disregarding eventual additional properties.

Losses for class probability estimation: a loss for class probability estimation, $\ell : \mathcal{Y} \times [0, 1] \rightarrow \mathbb{R}$, is expressed as

$$\ell(y, u) \doteq [y = 1] \cdot \ell_1(u) + [y = -1] \cdot \ell_{-1}(u), \quad (2)$$

where $[.]$ is Iverson's bracket (Knuth, 1992). Functions ℓ_1, ℓ_{-1} are called *partial losses*. A loss is *symmetric* when $\ell_1(u) = \ell_{-1}(1-u), \forall u \in [0, 1]$ (Nock & Nielsen, 2008) and *differentiable* when both partial losses are differentiable. Table 1 presents examples partial losses of symmetric losses. The pointwise conditional risk of posterior $\eta \in [0, 1]$ with respect to ground truth $\eta^* \in [0, 1]$ is $L(\eta, \eta^*) \doteq \mathbb{E}_{Y \sim B(\eta^*)} [\ell(Y, \eta)]$, i.e.,

$$L(\eta, \eta^*) = \eta^* \cdot \ell_1(\eta) + (1 - \eta^*) \cdot \ell_{-1}(\eta). \quad (3)$$

$B(\cdot)$ denotes a Bernoulli for picking label $Y = 1$. The associated (pointwise) *Bayes* risk is

$$\underline{L}(\eta^*) \doteq \inf_{\eta} L(\eta, \eta^*). \quad (4)$$

The interesting case is when the argument of the inf reduces to $\{\eta^*\}$, because then minimizing (3) for η 'encourages' to pick ground truth η^* . Formally, when (i) $\underline{L}(\eta) = L(\eta, \eta), \forall \eta \in [0, 1]$ and (ii) $L(\eta, \eta^*) > \underline{L}(\eta^*), \forall \eta \neq \eta^*$, we say that the loss is *strictly proper*, and *proper* when (i) holds. The population version of (3), when both $\eta, \eta^* \in [0, 1]^{\mathcal{X}}$, is the (*full*) risk (Reid & Williamson, 2011, pp 747),

$$\underline{L}(\eta, \eta^*, M) \doteq \mathbb{E}_{X \sim M} [L(\eta(X), \eta^*(X))]. \quad (5)$$

We now assume that all losses for class probability estimation used hereafter are strictly proper, symmetric and differentiable (SPSD) and satisfy the additional technical assumption that $\ell_{-1}(0) = 0$ (all but Jeffreys in Table 1 are SPSD), which makes η^* Bayes posterior in (1).

Definition 4.2. The **information** of calibrated $\tilde{\eta} \in [0, 1]^\mathcal{X}$ is: $\Delta\mathbb{L}(\tilde{\eta}, M) \doteq \mathbb{L}(\eta_\pi, \eta_\pi, M) - \mathbb{L}(\tilde{\eta}, \tilde{\eta}, M) = L(\pi) - \mathbb{L}(\tilde{\eta}, \tilde{\eta}, M)$.

This definition is a convenient restriction to calibrated posteriors of the original definition in De Groot (1962, Eq. (2.2)) and Reid & Williamson (2011, Eq. (20)). It represents how much ‘information’ $\tilde{\eta}$ brings compared to the constant calibrated posterior η_π . Decision tree induction would traditionally maximize $\Delta\mathbb{L}(\tilde{\eta}, M)$ via the minimisation of some $\mathbb{L}(\tilde{\eta}, \tilde{\eta}, M)$, where $\tilde{\eta}$ is the calibrated posterior at the leaves of the decision tree: CART’s uses the square loss (Breiman et al., 1984), C4.5 uses the log-loss (Quinlan, 1993), etc. .

Losses for measure estimation and binary task information: a substantial body of work has tightened the GAN loss to variational f -divergences (Nowozin et al., 2016; Nock et al., 2017). Here, we are also interested in such a formulation *but* for a very specific set of f introduced decades ago (Österreicher & Vajda, 1993, Theorem 2):

$$f^\pi(t) \doteq L(\pi) - (\pi t + 1 - \pi) \cdot \underline{L}\left(\frac{\pi t}{\pi t + 1 - \pi}\right), \forall t \in \mathbb{R}_+,$$

which involves prior π (controlled in the generative game).

Definition 4.3. The **information** of binary task $B \doteq (\pi, P, N)$, $\mathbb{I}(B)$, is the f^π -divergence

$$\mathbb{I}(B) \doteq \mathbb{I}_{f^\pi}(P, N), \quad (6)$$

where we recall $\mathbb{I}_{f^\pi}(P, N) \doteq \int f^\pi \left(\frac{dP}{dN} \right) dN$.

Given $\mathbb{I}(B)$, we could directly dig into the variational formulation of the f^π -divergence to design the generative modelling game and loss at the expense of an eventual slack due to the variational argument (Nowozin et al., 2016, Ineq. (4)). We avoid the slack *via* a trick using calibrated posteriors.

Losses for the adversarial generative game: we need to define two additional functions, for any posterior η :

$$\rho \doteq \frac{1 - \eta}{\eta} \quad ; \quad \varrho \doteq \frac{1}{\rho} \cdot \frac{1 - \pi}{\pi}. \quad (7)$$

We call ρ the **density ratio** and ϱ the **likelihood ratio**, following conventions in Reid & Williamson (2011, pp 746)². To take an example, if we consider Bayes posterior η^* in (1), then it follows $\varrho^* = dP/dN$, justifying the name. Let $f^*(z) \doteq \sup_t \{zt - f(t)\}$ denote the convex conjugate of f . We note that for any SPSD loss ℓ , f^π is differentiable.

Definition 4.4. Let B and ϱ be any binary task and likelihood ratio. Let $G_\ell(z) \doteq (f^\pi)^* \circ f^{\pi'}(z)$ and

$$\mathbb{G}_\ell(N|\varrho) \doteq -\mathbb{E}_{X \sim N}[G_\ell(\varrho(X))], \quad (8)$$

$$\mathbb{D}_\ell(\varrho|B) \doteq -(\mathbb{E}_{X \sim P}[f^{\pi'} \circ \varrho(X)] + \mathbb{G}_\ell(N|\varrho)) \quad (9)$$

²Names can otherwise vary in the literature.

respectively denote the **generator and discriminator risks**.

For any f -divergence, we have the f -GAN defining inequality (Nowozin et al., 2016, eqs. (4-6)) similar to -(9):

$$\mathbb{I}_f(P, N) \geq \sup_{\varrho} \{\mathbb{E}_{X \sim P}[f' \circ \varrho(X)] - \mathbb{E}_{X \sim N}[f^* \circ f' \circ \varrho(X)]\}, \quad (10)$$

so both (8) and (9) define the corresponding functions to minimise for the generator and discriminator in this variational inequality after the change $f \rightarrow f^\pi$. While the change is anecdotal with respect to the inequality (10), it conceptually operates a radical shift with respect to classical (f)-GANs: the generator’s loss is completely determined in our case by *the loss of the discriminator* as it appears in f^π , a loss whose design heavily relies on properness. The change also has a key fortunate mathematical consequence: we can replace the inequality (10) by a *chain of equalities* involving all key risks, as we now show.

Theorem 4.5. For any SPSD loss ℓ , any binary task B , any calibrated posterior $\tilde{\eta}$ whose likelihood ratio is denoted $\tilde{\varrho}$, the following holds:

$$\Delta\mathbb{L}(\tilde{\eta}, M_{\tilde{\eta}}) = -\mathbb{D}_\ell(\tilde{\varrho}|B_{\tilde{\eta}}) = \mathbb{I}(B_{\tilde{\eta}}). \quad (11)$$

Furthermore, we have the expression for function G_ℓ in Definition 4.4 (with $\ell^{\text{DR}}(\rho) \doteq \ell_{-1}(1/(1+\rho))$):

$$G_\ell(\varrho) = -L(\pi) + (1 - \pi) \cdot \ell^{\text{DR}}(\rho), \quad (12)$$

and the transformation $\varrho \leftrightarrow \rho$ is obtained via (7).

The proof (in Appendix, Section I.1) also provides the conjugate $(f^\pi)^*$, of potential independent interest.

Remark 4.6. Since f -divergences satisfy the data processing inequality, we also have for any calibrated posterior $\tilde{\eta}$, $\mathbb{I}(B_{\tilde{\eta}}) \leq \mathbb{I}(B)$. Together with (11), this gives a precise way of how the GAN game operates with calibrated posteriors and proper losses: training a discriminator to maximise its statistical information $\Delta\mathbb{L}(\tilde{\eta}, M_{\tilde{\eta}})$, e.g. as done with DT induction algorithms, increases as well the information of the binary task $\mathbb{I}(B_{\tilde{\eta}})$. On the other hand, training in turn the generator to minimize $\mathbb{G}_\ell(N_{\tilde{\eta}}|.)$ reduces the information of the binary task $\mathbb{I}(B_{\tilde{\eta}})$. In the case of DT algorithms, as the tree grows, its calibrated posterior $\tilde{\eta}$ converges to an ‘empirical Bayes’ best posterior (based on training real data). Disregarding generalisation issues, as long as the generator ‘stands’ the growth of the discriminator by keeping $\mathbb{I}(B_{\tilde{\eta}})$ small enough, it is guaranteed to improve with iterations.

A **generative loss ‘to learn against them all’ (almost)** Table 1 shows that ℓ^{DR} has several invariant properties for the losses shown. We formalise some of them.

Lemma 4.7. For any ℓ proper symmetric and differentiable, (i) ℓ^{DR} is decreasing and (ii) ℓ^{DR} is convex ρ or in $1/\rho$, $\forall \rho$.

Proof in Appendix, Section I.2. In the examples of Table 1, the ‘or’ in Lem. 4.7 is in fact an ‘and’. Convexity is

important because it yields a single loss to efficiently train the generator against any 'proper' trained discriminator: the χ^2 , i.e. the f -divergence whose generator is $f(t) \doteq (t-1)^2$.

Lemma 4.8. *For any PSD loss ℓ for which ℓ^{DR} is convex, any binary task B, calibrated posterior $\tilde{\eta}$ (likelihood ratio $\doteq \tilde{\rho}$), the following bound holds on the generator's risk \mathbb{G}_ℓ :*

$$\begin{aligned} & \mathbb{G}_\ell(N_{\tilde{\eta}} | \tilde{\rho}) \\ & \leq \underline{L}(\pi) - (1-\pi) \cdot \ell_{-1} \left(\frac{\pi}{1 + (1-\pi) \cdot \chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}})} \right) \quad (13) \end{aligned}$$

Proof in Appendix, section I.3. For any PSD loss, ℓ_{-1} is increasing (*Cf* proof of Theorem 4.5). Therefore, if we train the generator to reduce $\chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}})$, it reduces the RHS in (13) and provides a smaller bound on the generator's risk, *regardless* of the proper loss used as long as ℓ^{DR} is convex.

5. Training h and G

One way to train both the DT h and the GT G would be to proceed as in generative adversarial networks (Goodfellow et al., 2014). The DT can be trained using any commercial package (Breiman et al., 1984; Quinlan, 1993) or more generally any greedy induction of a tree minimizing a PSD loss with convex ℓ^{DR} . We can then train the adversarial GT by minimizing $\chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}})$ and alternate between phases of training the DT and training the GT. We call this setting '*adversarial*' for short. Due to the architecture of the models, there is a more specific training available for generative trees, more constrained than the adversarial setting but with a straightforward implementation and direct convergence guarantees coming from the convergence of the DT training. In this case, the GT *copies* the tree architecture of the DT and fits the probabilities to keep $\chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}}) = 0$. We call this setting the '*copycat*' setting. We detail them.

5.1. Adversarial training of the generator

We adopt a greedy induction of the GT. The current calibrated posterior of the generator h is $\tilde{\eta}$. Let λ denote a general leaf of h . \circledS denotes the current sampling node at the generator G that we are going to split to create a subtree with two sampling leaves and associated Bernoulli probability p to compute the new arcs at \circledS . Figure 3 provides an overview of the process, pointing to a new variable, τ , which is the local (relative) proportion of examples generated from the right sub-domain at the candidate split. For any $\lambda \in \Lambda(h)$ in the discriminator and candidate split at leaf $\circledS \in \Lambda(T)$ in the generator, we define:

- p_λ , the total weight of real examples reaching λ ;
- $n_\lambda \doteq \int_{\mathcal{X}(\lambda)} dN_{\tilde{\eta}}$, the theoretical proportion of fake examples reaching λ , where $\mathcal{X}(\lambda)$ is the subset of \mathcal{X} of observations that reach λ in h ;
- n_λ^0 , the total weight of fake examples reaching λ *but*

generated by $\Lambda(G) \setminus \{\circledS\}$ – these weights do not change after the split at \circledS ;

→ n_λ^l , the total weight of fake examples reaching λ , generated by \circledS *and* whose value for attribute X_i (the one considered for the split) is in X^l ;

→ n_λ^r , the total weight of fake examples reaching λ , generated by \circledS *and* whose value for attribute X_i (the one considered for the split) is in X^r .

It is worth noticing that $n_\lambda, n_\lambda^0, n_\lambda^l, n_\lambda^r$ can all be calculated *exactly* from the trees of h and G . After the split, 'only' the proportions in $\cup_\lambda \{n_\lambda^l\} \cup_\lambda \{n_\lambda^r\}$ are potentially changed by the split. We can compute τ as a function of these quantities:

$$\tau = \frac{\sum_{\lambda \in \Lambda(h)} n_\lambda^r}{\sum_{\lambda \in \Lambda(h)} n_\lambda^l + n_\lambda^r}. \quad (14)$$

Define three more quantities:

$$l_\lambda \doteq n_\lambda^0 + \frac{n_\lambda^l}{1-\tau}; r_\lambda \doteq n_\lambda^0 + \frac{n_\lambda^r}{\tau}; \delta_\lambda \doteq r_\lambda - l_\lambda.$$

These quantities are interpretable as follows: l_λ would be the new n_λ after split *if* we were to pick $p = 0$; r_λ would be the new n_λ after split *if* we were to pick $p = 1$ and δ_λ quantifies the difference in generation between these two extreme strategies. These strategies *are* extreme because for example if we choose $p = 0$, then we discard the support at \circledS covering observations whose value for X_i is in X^r . Some coefficients are particularly important to compute p :

$$\mu_{LL} \doteq \sum_{\lambda \in \Lambda(h)} \frac{l_\lambda^2}{p_\lambda}; \mu_{RR} \doteq \sum_{\lambda \in \Lambda(h)} \frac{r_\lambda^2}{p_\lambda}; \mu_{LR} \doteq \sum_{\lambda \in \Lambda(h)} \frac{l_\lambda r_\lambda}{p_\lambda}. \quad (15)$$

These are also interpretable: if we let $\chi^2(N'_{\tilde{\eta}}(p) || P_{\tilde{\eta}})$ denote the new χ^2 after the split at \circledS with Bernoulli p , then $\mu_{LL} = 1 + \chi^2(N'_{\tilde{\eta}}(0) || P_{\tilde{\eta}})$, $\mu_{RR} = 1 + \chi^2(N'_{\tilde{\eta}}(1) || P_{\tilde{\eta}})$, and μ_{LR} is a correlation between both strategies. The proof of Lemma 4.8 shows those identities. Algorithm TD-GEN summarizes the steps to split one leaf, without giving specific constraint on the choice of leaf to split \circledS , feature i , and split parameters (X^l, X^r) . We leave these open because general convergence rates can be obtained for TD-GEN that do not constraint those choices. Function CLAMP(z) is defined as $\text{CLAMP}(z) \doteq \max\{\min\{z, 1\}, 0\}$. We have two different regimes for the convergence of the χ^2 , depending on whether $p \in (0, 1)$ or $p \in \{0, 1\}$ (that latter case means that we *discard support* for the generation of examples). We give those results in two different Theorems. For our first Theorem, let $\mathbb{I}_\delta \doteq [0, \delta]$ denote a range of 'acceptable' values for the χ^2 's. The question we ask is what is the guaranteed convergence rate when we are *not* in this favorable case, that is, when the χ^2 's (before and after update) are not in \mathbb{I}_δ , a situation we refer to as TD-GEN being

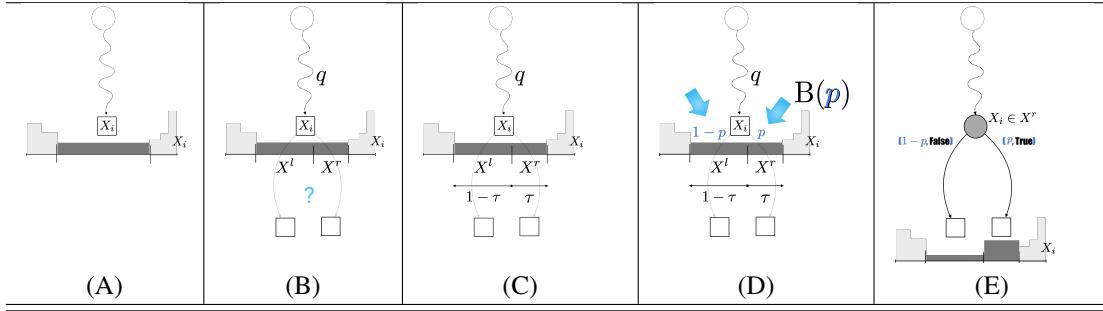


Figure 3. Splitting a sampling leaf in G to create a subtree with two new sampling leaves. (A) we pick a current leaf and decide on a variable X_i whose local density (therefore uniform, in dark gray) is going to be split in two at the new leaves. (B) a potential split creates two local intervals X^l, X^r , and we can compute the relative local proportion of examples that would be generated from X^r (τ) and from X^l ($1 - \tau$). (C) Finally, we compute Bernoulli's p (D). Note that τ does not appear after split (E), it is just used to compute p .

Algorithm 1 TD-GEN(G, h)

Input: current generator G , current discriminator h ;
Output: G with a new split;
Step 1 : pick $\circledcirc \in \Lambda(G)$, $i \in [d]$; // leaf and variable for the current split
Step 2 : choose (X^l, X^r) and compute τ ; // split choice
Step 3 : compute p as

$$p \leftarrow \text{CLAMP} \left(\frac{\mu_{LL} - \mu_{LR}}{\mu_{LL} + \mu_{RR} - 2\mu_{LR}} \right); \quad (16)$$

Step 4 : replace \circledcirc by a split as designed in Steps 1,2 w/ Bernoulli probability p as in (16);

$n_\lambda^l/(1 - \tau) = n_\lambda^r/\tau, \forall \lambda \in \Lambda(h)$, which brings after solving for τ , the relationships $n_\lambda^r/(n_\lambda^r + n_\lambda^l) = n_{\lambda'}^r/(n_{\lambda'}^r + n_{\lambda'}^l)$ for any two leaves in $\Lambda(h)$, and after simplifying, $n_\lambda^r \cdot n_{\lambda'}^l = n_\lambda^l \cdot n_{\lambda'}^r, \forall \lambda, \lambda' \in \Lambda(h)$. Filling any 2×2 contingency table with 'destination' leaves in the discriminator (λ, λ') versus 'provenance' in the generator (l, r) immediately leads to a Pearson's $\chi^2 = 0$. What the WGA prevents is thus the extreme independence where the generator's examples would be randomly attributed to the leaves in the discriminator.

5.2. Copycat training of the generator

Algorithm: when using a (decision) tree as discriminator, both the GT and DT have an underlying tree (graph). Copycat training takes advantage of this scenario as the GT G copies the tree of the DT h as it is learned: if h involves the usual top-down induction scheme, after each of the new splits in h , the generative tree G replicates the same split in its tree, computing the Bernoulli probabilities in such a way that the new proportion of fake observations is going to be the same as that of real observations at the new leaves of h . In other words, after the update of G , the new h performs as badly as a fair coin.

We see two substantial downsides to copycat vs adversarial training: the generator 'peeks' in the discriminator's tree, which can be a problem for privacy or fairness issues, and it has zero freedom to grow its own tree. There is, however, a major upside of copycat training over adversarial training: it requires no additional expensive computation for the new Bernoulli's p in G . Denote m_λ the number of positive examples at the leaf $\lambda \in \Lambda(h)$ to be split in h , and m_λ^r the number of positive examples ending up in its right sub-leaf after split. Then in G we have at the same λ : $p = m_\lambda^r/m_\lambda$.

Convergence: there is another benefit of copycat training: in the boosting model of Kearns & Mansour (1996, Section 5.1), the convergence rates for G towards the distribution of observed real data directly follow from the boosting rates of h on the supervised task. To show this, we proceed in two steps: the first introduces and conveniently decomposes a risk quantifying the discrepancy between measures in the

'outside regime \mathbb{I}_δ '. We show TD-GEN exhibits geometric convergence rate related to δ and the proximity of p to τ .

Theorem 5.1. Suppose $p \in (0, 1)$. For any $\varepsilon > 0, \delta > 0$, if (i) TD-GEN is outside regime \mathbb{I}_δ and (ii) p in Step 3 satisfies $|\tau - p| \geq \varepsilon$, then after one iteration of TD-GEN, we have:

$$\chi^2(N'_\eta(p) || P_\eta) \leq \frac{1}{1 + \delta\varepsilon^2} \cdot \chi^2(N_\eta || P_\eta). \quad (17)$$

Condition (ii) does make sense because if $p = \tau$, then there is no change in the χ^2 as $\chi^2(N'_\eta(\tau) || P_\eta) = \chi^2(N_\eta || P_\eta)$. To cover the case $p \in \{0, 1\}$, we need an additional assumption that mirrors the weak learning assumption that governs the convergence of the discriminator h in the boosting framework. We call it a *weak generating assumption*.

Definition 5.2. (δ-WGA) Let $\delta > 0$ be a constant. We say that the split at \circledcirc meets the δ -Weak Generating Assumption iff $\mu_{DD} \geq \delta \cdot \max\{\mu_{LL}, \mu_{RR}\}$, where $\mu_{DD} = \sum_{\lambda \in \Lambda(h)} \delta_\lambda^2 / p_\lambda$.

Theorem 5.3. Suppose $p \in \{0, 1\}$ and the δ -WGA holds. Then after one iteration of TD-GEN, we have:

$$\chi^2(N'_\eta(p) || P_\eta) \leq \frac{1}{1 + \delta(\tau + (1 - 2\tau)p)^2} \cdot \chi^2(N_\eta || P_\eta).$$

Remark 5.4. Definition 5.2 is 'weak' in a generative sense. Indeed, the only case where $\mu_{DD} = 0$ is when

density ratio model (Menon & Ong, 2016) – for this objective, we introduce indexes in notations, and let h_T denote the generator h after T splits, so that h_0 is the single-root DT. Similarly, we let $\tilde{\eta}_T$ denote the corresponding calibrated posterior and P^T denote the measure induced on \mathcal{X} by P and h_T by (i) ensuring it is locally uniform at each leaf and (ii) it locally sums to the local weight of P . In equation, it satisfies, U denoting the uniform measure,

$$\frac{dP^T}{dU}(\mathbf{x}) = \frac{\int_{\lambda} dP}{\int_{\lambda} dU}, \forall \mathbf{x} \text{ reaching } \lambda. \quad (18)$$

Denote $p_{\lambda} \doteq \int_{\lambda} dP$ and $u_{\lambda} \doteq \int_{\lambda} dU$. For differentiable and convex $F : \mathbb{R} \rightarrow \mathbb{R}$, the Bregman divergence with generator F is $B_F(z \| z') \doteq F(z) - F(z') - (z - z')F'(z')$. Given function $g : \mathbb{R} \rightarrow \mathbb{R}$, the generalized perspective transform of F given g is (Maréchal, 2005a;b; Nock et al., 2016) $\check{F}(z) \doteq g(z) \cdot F\left(\frac{z}{g(z)}\right)$. g is implicit in notation \check{F} .

Definition 5.5. The **Likelihood ratio risk** of P^T with respect to P for PSD loss ℓ is (with $g(z) \doteq z + (1 - \pi)/\pi$): $\mathbb{B}_{\ell}(P, P^T) \doteq \pi \cdot \mathbb{E}_U \left[B_{(-\check{L})} \left(\frac{dP}{dU} \middle\| \frac{dP^T}{dU} \right) \right]$.

Risks expressed as in Def. 5.5 have a history in density ratio estimation (Menon & Ong, 2016) (and references within).

Lemma 5.6. \forall PSD loss ℓ and DT h_T , $(-\check{L})$ is strictly convex and $\mathbb{B}_{\ell}(P, P^T) = \mathbb{I}_{f^{\pi}}(P, U) - \mathbb{I}_{f^{\pi}}(P_{\tilde{\eta}_T}, U_{\tilde{\eta}_T})$.

The proof is given in Appendix, Section I.5. Strict convexity is crucial: in such a case a Bregman divergence zeroes iff its two arguments are equal, implying at the risk level in Def. 5.5 that $\mathbb{B}_{\ell}(P, P^T) = 0$ iff $P = P^T$ almost everywhere. $\mathbb{I}_{f^{\pi}}(P, U)$ is a constant and the data processing inequality satisfied by f -divergences brings $0 = \mathbb{I}_{f^{\pi}}(P_{\tilde{\eta}_0}, U_{\tilde{\eta}_0}) \leq \dots \leq \mathbb{I}_{f^{\pi}}(P_{\tilde{\eta}_T}, U_{\tilde{\eta}_T}) \leq \dots \leq \mathbb{I}_{f^{\pi}}(P, U)$, so regardless of the top-down induction algorithm used for h , Lemma 5.6 shows a form of convergence of P^T towards P as accounted by the likelihood ratio risk $\mathbb{B}_{\ell}(P, P^T)$. The last part of copycat training’s convergence is to make those inequalities strict with guaranteed slack: this is achieved using the boosting analysis of Kearns & Mansour (1996) as is.

We do not put iteration indexes in G , assuming the one we consider is the one after the update of the last discriminator h_T . Denote λ a leaf to be split in h and P_{λ}, U_{λ} the distributions conditioned to reaching λ ; we denote as ‘uniformly generated’ the observations sampled from U_{λ} . Define the local mixture $M_{\lambda} \doteq \pi \cdot P_{\lambda} + (1 - \pi) \cdot U_{\lambda}$ and the balanced mixture, M'_{λ} , is defined as $M_{\lambda} \doteq (1/2) \cdot P_{\lambda} + (1/2) \cdot U_{\lambda}$. Let t denote the predicate value of a split chosen for λ .

Definition 5.7. (Kearns & Mansour, 1996) Fix $\delta > 0$. Predicate t at leaf λ satisfies the δ -Weak Hypothesis Assumption (WHA) iff $\Pr_{M'_{\lambda}}[t(X) \neq Y] \leq 1/2 - \delta$.

It turns out that the balanced mixture is the one against which each new split in h is evaluated after the generator

is updated in copycat training (the modifications at G are local since the underlying tree defines a partition of \mathcal{X}). We use the WHA to ensure that the split chosen at any leaf during copycat training complies with Definition 5.7. Using a result of Kearns & Mansour (1996), this brings guaranteed rates for the maximisation of the information of its calibrated posterior (Definition 4.2). Theorem 4.5 then directly yields rates for the maximisation of $\mathbb{I}_{f^{\pi}}(P_{\tilde{\eta}_T}, U_{\tilde{\eta}_T})$, and Lemma 5.6 translates them to convergence for $P^T = G$ towards P , where G denotes the measure induced by generator G (equality $P^T = G$ is guaranteed by copycat training). We make those convergence rates explicit for the boosting-optimal splitting criterion, Matusita’s loss (Table 1), for which $\underline{L}(u) = 2\sqrt{u(1-u)}$. For any $\varepsilon \in [0, 1]$, we abbreviate $\underline{L}_{\varepsilon}(B) \doteq \varepsilon \cdot \underline{L}(\pi) + (1 - \varepsilon) \cdot \underline{L}(\eta^*, \eta^*, M)$.

Theorem 5.8. Define the binary task $B = (\pi, P, U)$, M . Suppose PSD loss ℓ is Matusita’s loss and the WHA is satisfied at each split of h . Then for any $\varepsilon \in [0, 1]$, if the number of splits in h satisfies $T \geq \left(\frac{1}{\underline{L}_{\varepsilon}(B)}\right)^{\frac{32}{\delta^2}}$, then the likelihood ratio risk achieved by generator G with respect to the distribution of real observations satisfies $\mathbb{B}_{\ell}(P, G) \leq \varepsilon \cdot \mathbb{I}_{f^{\pi}}(P, U)$.

The proof directly follows from the proof of Kearns & Mansour (1996, Theorem 10), using Defns in (4.2), (5.5), Thm 4.5 and Lem. 5.6 to calibrate the bound that is needed on the information of $\tilde{\eta}_T$ to guarantee the bound on $\mathbb{B}_{\ell}(P, G)$.

6. Experiments

We carried out experiments on four topics: missing data imputation, training on generated data (training on fakes vs training on real), fake-real discrimination (distinguishing fakes from real) and generated augmentation (adding fakes to real for training) on a total of 11 readily available datasets, from the UCI (Dua & Graff, 2021), Kaggle and the Stanford Open Policing project, to which we added 4 simulated datasets. For simplicity, all GT experiments use copycat training, implemented in Java. Due to the lack of space, we refer to App, Section II for all details. To get the simple 2D heatmaps in Figure 1, we trained generative models with the full domain (1K epochs for neural nets, 10K splits for GTs) and generated the same amount of data as in the training sample. An interesting observation, quite intuitive considering copycat training’s properties and visible on all 2D heatmaps (see also App), is the virtual absence of mode collapse during training: modes that appears gets refined during induction but do not disappear at some point.

6.1. Missing data imputation (‘IMPUTE’)

Objective and experimental setting A GT G is not just useful to generate data: it can trivially be used for missing data imputation (Muzellec et al., 2020). For this, we con-

Us vs Mice #trees p. fold	NORM N/A	CART 10	RF 1 000				led	CART 35	RF 3 500	led24	CART 125	RF 12 500
(q=5%)	U(0.003)	U	U(0.07)					U	U		M	U
10%	U(0.03)	U(0.002)	U(0.007)					U	M		U	M
20%	U(0.0005)	U(0.001)	U(0.001)					U(0.05)	U		U	M
50%	U(0.04)	U	U					M(0.02)	M(0.06)		M(0.01)	M(0.07)

Table 2. Best method (**U**(s) vs **M**(ice); p -values shown if $p \leq 0.1$) on IMPUTE. circGauss: domain (green dots) vs imputed data (red).

U(.)	TRAIN-GEN			GEN-DISCRIM		
	10*	300*	1K**	10	300	1K
10	1/7/2	3/4/3	2/3/5	7/4/2	6/5/2	6/5/2
300	8/1/1	8/1/1	4/5/1	8/4/1	8/3/2	8/3/2
MAX	8/1/1	8/1/1	7/2/1	8/4/1	8/3/2	8/3/2

Table 3. TRAIN-GEN (left table) & GEN-DISCRIM (right table): statistical wins / ties / statistical losses for us (**U()**) vs CTGAN(**C()**). Statistical = significant for $p \leq 0.01$. For example, **a/b/c** means we statistically win **a** times, lose **b** times and there is no statistical difference **c** times. On TRAIN-GEN, each red star (*) indicates a domain for which the related technique performed statistically worse than uniform sampling (UNIF) for $p \leq 0.05$.

strain the support of the tree to the observed variables and then sample in region(s) of maximal density. This costs no more than $\mathcal{O}(|\Lambda(G)|)$ per observation. We compared against a few powerful alternatives (mostly tree-based) from the R `mice` package (van Buuren & Groothuis-Oudshoorn, 2011). Such methods rely on round-robin prediction of missing values: after having initialized them, one circles several times (5 in our experiments) through predicting each column from all others using trained models from a specific `method`. We used `method` ∈ {NORM, CART, RF} (RF = random forests with 100 trees each, CART learns regression / decision trees (Breiman et al., 1984)). It is important to realise that even on a domain like `led24` with 25 variables, 5 round-robin iterations with RF implies using no less than **12 500** trees *per fold* when we rely on **1** in our GT. We grow the GT to its MAX size (limit: 10 000 nodes) and prevent splits with $p \in \{0, 1\}$, thus avoiding discarding support for data generation. We generate Missing Completely At Random (MCAR, van Buuren (2018)) data by removing a fixed proportion $q \in \{5\%, 10\%, 20\%, 50\%\}$, embedded in a 5-fold cross validation for each q . Imputing a complete dataset on each fold, we judge imputation’s quality with optimal transport’s Wasserstein’s W_2^2 (Muzellec et al., 2020).

Results Table 2 summarises 3 domains (more in App) giving a good panorama of our observations, the first of which being the fact that our simple approach can in fact *beat* `mice` on problems with restricted number of variables (but the picture is reversed on domains with large number of variables, App). Imputations’ quality wrt `mice` is clear from Table 2 ($q = 20\%$). While our objective was not to beat such fit-for-purpose imputation methods relying on a comparatively huge number of models, we remark that our results can serve as basis for using GTs in more sophisticated approaches.

6.2. Training on generated data ('TRAIN-GEN')

Objective and experimental setting In this basic experiment, we seek to answer whether generated data can be used *in lieu* of the original data to solve the original data’s supervised / regression problem (*e.g.* predicting the variety for `iris`). We use a 5-folds CV experiment where on each fold a supervised classifier is trained on fake or real data and then used to classify the (fresh) real data’s fold. Fake data is obtained from a generator trained on the training data, then sampled for the same data size. We consider 3 GTs with different sizes, with 10, 300 and up to MAX splits (same as in Section 6.1). Our contender is the state of the art CTGAN(Xu et al., 2019), trained for a number of epochs in {10, 300, 1K}; the original data’s supervised problem is then solved by training RFs and gradient boosted decision trees (GBDT) on real or fake data, and comparing the output accuracy / RMSE (details in App, Section II.3).

Results Table 3 (left) provides a summary of the results on the 10 total domains considered, from which it emerges that when GTs have 600+ nodes (300+ splits), we tend to beat neural networks (NNs, regardless of the number of epochs considered). What is worse for NNs is that in a total of 4 cases, they are statistically beaten by a *uniform sampling* of the training data, which means they fail at learning the domain’s characteristics. This never happens to GTs. Detailed 2D plots display that GTs tend to better learn domain-specific features. Also, the final size of the GT can be tiny compared to the training data, *e.g.* less than 0.5% on `dna` and `open policing` (see App, Section II.6).

6.3. Fake-real discrimination ('GEN-DISCRIM')

Objective and experimental setting While the objective fits in a simple question (*can the generated data look like real data ?*), its treatment necessitated a complex pipeline, in particular to avoid rewarding generators whose output would be a mere copy their training sample. The complete pipeline is detailed in App (Section II.7); very briefly, it starts by shuffling a 3-partition of the training data in a $3! = 6$ -fold CV and ends up with supervised RF / GBDT classifiers (same as in Section 6.2) for a 2-class supervised problem of fakes vs real distinction. The smaller their accuracy, the *better* is the generator. We use CTGANs as contenders; all parameters (GTs, CTGANs) are the same as in Section 6.2.

Results Table 3 (right) provides a summary of the results on 13 total domains considered. They display that GTs (regardless of their sizes) achieved a better job at fooling

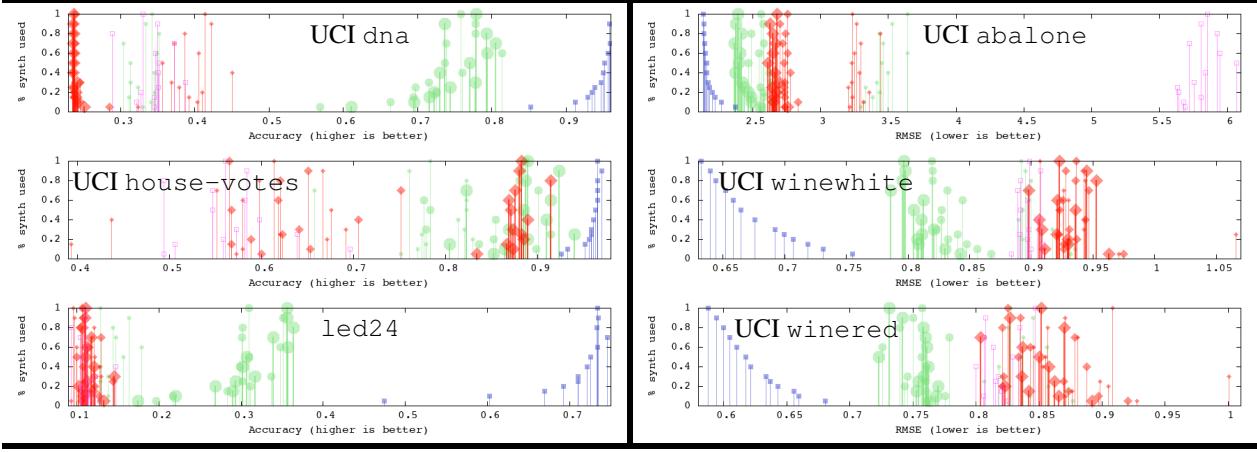


Table 4. Experiment GEN-AUG: detailed results on six domains (more in App, Section II.8). On the *left* pane, the metric is accuracy (higher = better); on the *right* pane, the metric is RMSE (lower = better). In each plot, the *x* value of a vertical bar indicates a method’s metric and the height along the *y* axis indicates the % of real data that represents generated data used to train the final classifier (up to 100%). Green filled circles are GTs results, the size of the circle indicating the number of splits in the GTs ($\bullet = 10$, $\circ = 300$, $\bullet\bullet = 10K$). Red filled diamonds are CTGANs results, the size of the diamond indicating the number of epochs ($\cdot = 10$, $\diamond = 300$, $\blacklozenge = 1K$). Finally, empty pink squares (\square) are UNIF’s results and filled blue squares (\blacksquare) are those of COPY (the “optimal” reference from the standpoint of the task).

classifiers than neural nets. Much more interesting is perhaps the fact that GTs managed, on 3 (simulated) datasets, to better fool classifier than the *original real data itself*. This never happened for CTGANs. However, there is still a gap to fill for all techniques: CTGANs do a statistically worse job at fooling classifiers than *uniformly generated data* on 6 domains while GTs do statistically worse on 3 domains.

6.4. Generated augmentation (‘GEN-AUG’)

Objective and experimental setting Supplementing real data with generative data is a particular case of data augmentation, and this setting is a variation on the TRAIN-GEN experiment. Two key factors under control are (i) the technique used to supplement the additional data and (ii) the proportion of additional data with respect to the training set size. In the case of (i), we do not just include data generated by our technique or GANs but also consider adding purely UNIFormly generated data and also adding real data from COPY itself. A generative model is all the better as its performances come closer to the COPY metrics.

Results Table 4 displays results on 6 domains, with a clear advantage to GTs on 5 out of the 6. On dna, GTs improve CTGANs by up to more than 50% accuracy, as neural nets clearly overfit the domain, being beaten by UNIF; on house-votes, CTGANs trained with the largest number of epochs get good results, though GTs are the only one managing to *beat* COPY with 5% or 10% additional real data. On 4 out of 6 domains, more training neural nets does not improve results, while learning bigger GTs *does* improve results in general, as well as does training with more generated data. Such experiments tend to indicate that training GTs can be quite resilient to overfitting.

7. Discussion and Conclusion

Our contributions have different application spectra: while our models are tree-based, our contribution on losses has wider applicability to any calibrated classifiers. While copycat training is specific to a tree vs tree training procedure, our adversarial algorithm could be used to train generative trees against any calibrated classifier. GTs have advantages that neural nets do not necessarily have: they provide us with an *exact and cheap to compute* expression of the measure learned, they can easily be used for *missing data imputation*, and they also collect many benefits of DTs: interpretability (of the measure); they can be trained using various feature types (numeric, nominal, ordinal, etc.); and they can be trained from data with missing values. They also share some downsides of DTs, such as the fact that the underlying tree graph induces an axis-parallel partition of the measure’s support. Tricks used to alleviate DTs downsides could also be used for GTs, though maybe in a non-trivial way (Heath et al., 1993). Important open problems include scaling the benefits of generative trees to *ensembles* of generative trees. We believe our work brings new tools for models, losses and algorithms to train powerful generative models tailored to tabular data, and hope it contributes to fill the gap in data generation quality for tabular data noted in recent work.

Acknowledgments

The authors thank Ehsan Amid, Sercan Arik, Olivier Bousquet, Julie Josse, Yishay Mansour, Aditya Krishna Menon, Madeleine Udell, Jean-Philippe Vert, Manfred Warmuth, Bob Williamson and the reviewers for many comments, suggestions and stimulating discussions.

References

- Arik, S.-Ö. and Pfister, T. TabNet: Attentive interpretable tabular learning. In *AAAI'21*, pp. 6679–6687, 2021.
- Box, G.-E.-P. and Muller, M.-E. A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, 29(2):610–611, 1958.
- Breiman, L., Freidman, J. H., Olshen, R. A., and Stone, C. J. *Classification and regression trees*. Wadsworth, 1984.
- Camino, R.-D., State, R., and Hammerschmidt, C.-A. Over-sampling tabular data with deep generative models: Is it worth the effort? In *I Can't Believe It's Not Better Workshop (ICBINB@NeurIPS 2020)*, 2020.
- Chui, M., Manyika, J., Miremadi, M., Henke, N., Nel, R. C. P., and Malhotra, S. *Notes from the AI frontier*. McKinsey Global Institute, 2018.
- De Groot, M.-H. Uncertainty, information, and sequential experiments. *Annals of Mathematical Statistics*, 33(2): 404–419, 1962.
- Dua, D. and Graff, C. UCI machine learning repository, 2021. URL <http://archive.ics.uci.edu/ml>.
- Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A.-C. Adversarially learned inference. In *ICLR'17*. OpenReview.net, 2017.
- Friedman, J., Hastie, T., and Tibshirani, R. Additive Logistic Regression : a Statistical View of Boosting. *Ann. of Stat.*, 28:337–374, 2000.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NIPS*27*, pp. 2672–2680, 2014.
- Heath, D., Kasif, S., and Salzberg, S. Learning oblique decision trees. In *Proc. of the 13th International Joint Conference on Artificial Intelligence*, pp. 1002–1007, 1993.
- Kearns, M. and Mansour, Y. On the boosting ability of top-down decision tree learning algorithms. In *Proc. of the 28th ACM STOC*, pp. 459–468, 1996.
- Knuth, D.-E. Two notes on notation. *The American Mathematical Monthly*, 99(5):403–422, 1992.
- Linsley, J.-W., Linsley, D.-A., Lamstein, J., Ryan, G., Shah, K., Castello, N.-A., Oza, V., Kalra, J., Wang, S., Tokuno, Z., Javaherian, A., Serre, T., and Finkbeiner, S. Superhuman cell death detection with biomarker-optimized neural networks. *Science Advances*, 7(50):eabf8142, 2021.
- Maréchal, P. On a functional operation generating convex functions, part 1: duality. *J. of Optimization Theory and Applications*, 126:175–189, 2005a.
- Maréchal, P. On a functional operation generating convex functions, part 2: algebraic properties. *J. of Optimization Theory and Applications*, 126:375–366, 2005b.
- Menon, A. and Ong, C.-S. Linking losses for density ratio and class-probability estimation. In *33rd ICML*, pp. 304–313, 2016.
- Muzellec, B., Josse, J., Boyer, C., and Cuturi, M. Missing data imputation using optimal transport. In *37th ICML*, volume 119, pp. 7130–7140, 2020.
- Ni, Y., Koniusz, P., Hartley, R., and Nock, R. Manifold learning benefits GANs. *CoRR*, abs/2112.12618, 2021.
- Nock, R. and Nielsen, F. On the efficient minimization of classification-calibrated surrogates. In *NIPS*21*, pp. 1201–1208, 2008.
- Nock, R., Menon, A.-K., and Ong, C.-S. A scaled Bregman theorem with applications. In *NIPS*29*, pp. 19–27, 2016.
- Nock, R., Cranko, Z., Menon, A.-K., Qu, L., and Williamson, R.-C. f -GANs in an information geometric nutshell. In *NIPS*30*, 2017.
- Nowozin, S., Cseke, B., and Tomioka, R. f -GAN: training generative neural samplers using variational divergence minimization. In *NIPS*29*, pp. 271–279, 2016.
- Österreicher, F. and Vajda, I. Statistical information and discrimination. *IEEE Trans. IT*, 39(3):1036–1039, 1993.
- Quinlan, J. R. *C4.5 : programs for machine learning*. Morgan Kaufmann, 1993.
- Reid, M.-D. and Williamson, R.-C. Information, divergence and risk for binary experiments. *JMLR*, 12:731–817, 2011.
- Rob, P. and Coronel, C. *Database systems - design, implementation, and management*. Boyd and Fraser, 1995.
- Rockafellar, R. T. *Convex Analysis*. Princeton University Press, 1970.
- Savage, L.-J. Elicitation of personal probabilities and expectations. *J. of the Am. Stat. Assoc.*, pp. 783–801, 1971.
- Schapire, R. E. and Singer, Y. Improved boosting algorithms using confidence-rated predictions. In *9th COLT*, pp. 80–91, 1998.
- Sypherd, T., Nock, R., and Sankar, L. Being properly improper. In *39th ICML*, 2022.

van Buuren, S. *Flexible Imputation of Missing Data*. Chapman & Hall / CRC, 2018.

van Buuren, S. and Groothuis-Oudshoorn, K. mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011.

van Erven, T. and Harremoës, P. Rényi divergence and kullback-leibler divergence. *IEEE Trans. IT*, 60:3797–3820, 2014.

Xiao, C., Zhong, P., and Zheng, C. BourGAN: Generative networks with metric embeddings. In *NeurIPS’18*, pp. 2275–2286, 2018.

Xu, L., Skouliaridou, M., Cuesta-Infante, A., and Veeramachaneni, K. Modeling tabular data using conditional GAN. In *NeurIPS*32*, pp. 7333–7343, 2019.

Yoon, J., Jordon, J., and van der Schaar, M. GAIN: missing data imputation using generative adversarial nets. In *35th ICML*, volume 80, pp. 5675–5684, 2018.

Appendix

Abstract

This is the Appendix to Paper "Generative Trees: Adversarial and Copycat" by R. Nock and M. Guillame-Bert, appearing in ICML'22. To differentiate with the numberings in the main file, the numbering of Theorems is letter-based (A, B, ...).

Table of contents

Supplementary material on proofs _____ Pg 13

→ Proof of Theorem 4.5	Pg 13
→ Proof of Lemma 4.7	Pg 16
→ Proof of Lemma 4.8	Pg 17
→ Proof of Theorems 5.1 and 5.3	Pg 18
→ Proof of Lemma 5.6	Pg 21

Supplementary material on experiments _____ Pg 23

→ Examples of generative trees	Pg 23
→ Domains	Pg 23
→ Data generation experiments	Pg 25
→ 'Missing data imputation' experiments (IMPUTE)	Pg 35
→ 'Training on generated data' experiment (TRAIN-GEN)	Pg 38
→ 'Fake-real discrimination' experiment (GEN-DISCRIM)	Pg 42
→ 'Generated augmentation' experiment (GEN-AUG)	Pg 44

I. Appendix on proofs

I.1. Proof of Theorem 4.5

We proceed in several steps.

▷ **Proof of the identity between the external elements in (11)** – We write, as in [Reid & Williamson \(2011, Appendix A.3\)](#), the second equality of

$$\begin{aligned} \mathbb{I}(B_{\tilde{\eta}}) &\doteq \mathbb{I}_{f^\pi}(P_{\tilde{\eta}}, N_{\tilde{\eta}}) \\ &= \underline{L}(\pi) - \int \underline{L}\left(\pi \cdot \frac{dP_{\tilde{\eta}}}{dM_{\tilde{\eta}}}\right) dM_{\tilde{\eta}} \\ &= \underline{L}(\pi) - \int \underline{L}(\tilde{\eta}) dM_{\tilde{\eta}} \end{aligned} \quad (19)$$

$$\begin{aligned} &= \underline{L}(\pi) - \mathbb{E}_{X \sim M_{\tilde{\eta}}} [\underline{L}(\tilde{\eta})] \\ &= \underline{L}(\pi) - \mathbb{E}_{X \sim M_{\tilde{\eta}}} [\underline{L}(\tilde{\eta}(X), \tilde{\eta}(X))] \end{aligned} \quad (20)$$

$$= \underline{L}(\pi) - \underline{L}(\tilde{\eta}, \tilde{\eta}, M_{\tilde{\eta}}) \quad (21)$$

$$= \Delta \underline{L}(\tilde{\eta}, M_{\tilde{\eta}}), \quad (22)$$

where (19) holds because $\tilde{\eta}$ is calibrated and (20) holds because the loss is proper and $\tilde{\eta}$ matches Bayes posterior on $B_{\tilde{\eta}}$.

▷ **Proof of the rightmost identity in (11)** – We first prove several helper results. We first remark that ℓ being strictly proper differentiable implies f^π strictly convex and differentiable. We show these results for completeness, starting by showing \underline{L} strictly concave: otherwise, we write $\underline{L}(\eta_{1:2}) = (1/2) \cdot (\underline{L}(\eta_1) + \underline{L}(\eta_2))$ for $\eta_{1:2} \doteq (1/2) \cdot (\eta_1 + \eta_2)$. If we had both

$$L(\eta_1, \eta_{1:2}) > L(\eta_{1:2}, \eta_{1:2}) \doteq \underline{L}(\eta_{1:2}) = (1/2) \cdot (\underline{L}(\eta_1) + \underline{L}(\eta_2)), \quad (23)$$

$$L(\eta_2, \eta_{1:2}) > L(\eta_{1:2}, \eta_{1:2}) \doteq \underline{L}(\eta_{1:2}) = (1/2) \cdot (\underline{L}(\eta_1) + \underline{L}(\eta_2)), \quad (24)$$

then making the average of both yields $(1/2) \cdot (\underline{L}(\eta_1) + \underline{L}(\eta_2)) > (1/2) \cdot (\underline{L}(\eta_1) + \underline{L}(\eta_2))$, a contradiction, and yielding for example $L(\eta_1, \eta_{1:2}) \leq L(\eta_{1:2}, \eta_{1:2})$, contradicting the strict properness of the loss in $\eta_{1:2}$. Strict concavity of \underline{L} implies strict convexity of f^π from its definition in (6). Also, the differentiability of the partial losses imply the differentiability of f^π .

For any strictly convex differentiable function f , we have $f^*(z) = z f'^{-1}(z) - f(f'^{-1}(z))$ and $(f^*)' = f'^{-1}$, and if it is lower semicontinuous then $f^{**} = f$. We check that f^π is indeed lower semicontinuous. Because \underline{L} is continuous ([Sypherd et al., 2022, Lemma 3.1](#)), we study the set

$$\mathcal{J}(\alpha) \doteq \left\{ t : (\pi t + 1 - \pi) \cdot \underline{L}\left(\frac{\pi t}{\pi t + 1 - \pi}\right) \geq \underline{L}(\pi) - \alpha \right\}, \quad (25)$$

for $\alpha \in \mathbb{R}$. Denote for short $g(t) \doteq (\pi t + 1 - \pi) \cdot \underline{L}\left(\frac{\pi t}{\pi t + 1 - \pi}\right)$, which is continuous and also concave [Reid & Williamson \(2011, Appendix A.3\)](#). Recalling $B_F(z \| z') \doteq F(z) - F(z') - (z - z')F'(z')$ the Bregman divergence with generator F . We have

$$\begin{aligned} g'(t) &= \pi \cdot \underline{L}'\left(\frac{\pi t}{\pi t + 1 - \pi}\right) + \frac{\pi(1-\pi)}{\pi t + 1 - \pi} \cdot \underline{L}'\left(\frac{\pi t}{\pi t + 1 - \pi}\right) \\ &= \pi \cdot \left((-\underline{L})(1) - (-\underline{L})\left(\frac{\pi t}{\pi t + 1 - \pi}\right) - \left(1 - \frac{\pi t}{\pi t + 1 - \pi}\right) \cdot (-\underline{L}')\left(\frac{\pi t}{\pi t + 1 - \pi}\right) \right) \end{aligned} \quad (26)$$

$$= \pi \cdot B_{-\underline{L}}\left(1 \middle\| \frac{\pi t}{\pi t + 1 - \pi}\right) \quad (27)$$

$$= \pi \cdot \ell_1\left(\frac{\pi t}{\pi t + 1 - \pi}\right), \quad (28)$$

which, since $\ell_1(1) = 0$, shows $\lim_{+\infty} g' = 0^+$; g being concave, g' is decreasing (which also shows ℓ_1 decreasing and ℓ_{-1} increasing). To conclude, g is increasing; hence, (when it is not empty) $\mathcal{J}(\alpha) = [t_u, +\infty)$ for some finite t_u , and so

$\overline{\mathcal{I}(\alpha)} = (-\infty, t_u)$ is open, showing the closedness of $\mathcal{I}(\alpha)$ and the closedness of $\{t : f^\pi(t) \leq \alpha\}$, and we get e.g. from Rockafellar (1970, Theorem 7.1, point (b)) that f^π is lower semicontinuous and thus $(f^\pi)^{**} = f^\pi$. We complete the proof of the identities: (26) holds because $\ell_{-1}(0) = 0$ implies $\underline{L}(0) \doteq 0 \cdot \ell_1(0) + 1 \cdot \ell_{-1}(0) = 0$ and because ℓ is also symmetric, then $\underline{L}(1) \doteq 1 \cdot \ell_1(1) + 0 \cdot \ell_{-1}(1) = \ell_1(1) = \ell_{-1}(0) = 0$. Hence, $\underline{L}(0) = \underline{L}(1) = 0$. We state (28) as a standalone Lemma.

Lemma A. Suppose ℓ is proper differentiable and satisfies $\ell_{-1}(0) = \ell_1(1) = 0$. Then

$$\begin{aligned} B_{-\underline{L}}(0||u) &= \ell_{-1}(u). \\ B_{-\underline{L}}(1||u) &= \ell_1(u). \end{aligned}$$

Proof: The following two relationships hold because ℓ is proper, $\forall u \in [0, 1]$ (using (4)):

$$\underline{L}(u) = u \cdot \ell_1(u) + (1 - u) \cdot \ell_{-1}(u), \quad (29)$$

$$u \cdot \ell'_1(u) + (1 - u) \cdot \ell'_{-1}(u) = 0. \quad (30)$$

The second identity expresses the fact that properness implies (from (3)),

$$\left. \frac{\partial}{\partial v} L(v, u) \right|_{v=u} = 0. \quad (31)$$

We derive $\forall u \in [0, 1]$, using $\underline{L}(0) = 0$,

$$\begin{aligned} B_{-\underline{L}}(0||u) &= \underline{L}(u) - u \underline{L}'(u) \\ &= u \cdot \ell_1(u) + (1 - u) \cdot \ell_{-1}(u) - u \cdot (\ell_1(u) + u \cdot \ell'_1(u) - \ell_{-1}(u) + (1 - u) \cdot \ell'_{-1}(u)) \\ &= u \cdot \ell_1(u) + (1 - u) \cdot \ell_{-1}(u) - u \cdot (\ell_1(u) - \ell_{-1}(u)) \\ &= \ell_{-1}(u), \end{aligned}$$

as anticipated. We also have $\forall u \in [0, 1]$, using $\underline{L}(1) = 0$,

$$\begin{aligned} B_{-\underline{L}}(1||u) &= \underline{L}(u) + (1 - u) \underline{L}'(u) \\ &= \underline{L}(u) - u \underline{L}'(u) + \underline{L}'(u) \\ &= u \cdot \ell_1(u) + (1 - u) \cdot \ell_{-1}(u) - u \cdot (\ell_1(u) - \ell_{-1}(u)) + \ell_1(u) - \ell_{-1}(u) \\ &= \ell_{-1}(u) + \ell_1(u) - \ell_{-1}(u) \\ &= \ell_1(u), \end{aligned}$$

which completes the proof of the Lemma. □

We now finish the proof of (11). Since $\tilde{\eta}$ is calibrated, then the corresponding likelihood satisfies

$$\tilde{\varrho} \doteq \frac{\tilde{\eta}}{1 - \tilde{\eta}} \cdot \frac{1 - \pi}{\pi} \quad (32)$$

$$\begin{aligned} &= \frac{\pi \cdot \frac{dP_{\tilde{\eta}}}{dM_{\tilde{\eta}}}}{(1 - \pi) \cdot \frac{dN_{\tilde{\eta}}}{dM_{\tilde{\eta}}}} \cdot \frac{1 - \pi}{\pi} \\ &= \frac{dP_{\tilde{\eta}}}{dN_{\tilde{\eta}}}. \end{aligned} \quad (33)$$

Putting all this together, we get:

$$\begin{aligned}
 -\mathbb{D}_\ell(\tilde{\varrho}|B_{\bar{\eta}}) &\doteq \mathbb{E}_{X \sim P_h} [f^{\pi'} \circ \tilde{\varrho}(X)] - \mathbb{E}_{X \sim N_{\bar{\eta}}} [G_\ell(\tilde{\varrho}(X))] \\
 &= \mathbb{E}_{X \sim P_{\bar{\eta}}} [f^{\pi'} \circ \tilde{\varrho}(X)] - \mathbb{E}_{X \sim N_{\bar{\eta}}} [(f^\pi)^* \circ f^{\pi'} \circ \tilde{\varrho}(X)] \\
 &= \int \frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \cdot f^{\pi'} \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \right) dN_{\bar{\eta}} - \int (f^\pi)^* \circ f^{\pi'} \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \right) dN_{\bar{\eta}} \\
 &= \int \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \cdot f^{\pi'} \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \right) - (f^\pi)^* \circ f^{\pi'} \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \right) \right) dN_{\bar{\eta}} \\
 &= \int \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \cdot (f^{\pi'*})^{-1} \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \right) - (f^\pi)^* \circ (f^{\pi'*})^{-1} \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \right) \right) dN_{\bar{\eta}} \\
 &= \int (f^\pi)^{**} \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \right) dN_{\bar{\eta}} \\
 &= \int f^\pi \left(\frac{dP_{\bar{\eta}}}{dN_{\bar{\eta}}} \right) dN_{\bar{\eta}} \\
 &= \mathbb{I}_{f^\pi}(P_{\bar{\eta}}, N_{\bar{\eta}}) \\
 &= \mathbb{I}(B_{\bar{\eta}}),
 \end{aligned} \tag{34}$$

as claimed. The key to avoiding the inequality (10) is (34), which holds only because when the σ -algebras of the measures involved are coarsened with the level set of a calibrated posterior, it becomes Bayes posterior in the measure spaces obtained. This ends up the proof of (11).

▷ **Proofs of (12) and expression of $(f^\pi)^*(z)$** – We now compute $(f^\pi)^*(z)$. Let

$$t \doteq \frac{p(1-\pi)}{(1-p)\pi}. \tag{35}$$

We have:

$$\begin{aligned}
 (f^\pi)^*(z) &\doteq \sup_{t \geq 0} \left\{ tz - \underline{L}(\pi) + (\pi t + 1 - \pi) \cdot \underline{L} \left(\frac{\pi t}{\pi t + 1 - \pi} \right) \right\} \\
 &= -\underline{L}(\pi) + \sup_{t \geq 0} \left\{ tz + (\pi t + 1 - \pi) \cdot \underline{L} \left(\frac{\pi t}{\pi t + 1 - \pi} \right) \right\} \\
 &= -\underline{L}(\pi) + \sup_{p \in [0,1]} \left\{ \frac{p(1-\pi)}{(1-p)\pi} \cdot z + \frac{1-\pi}{1-p} \cdot \underline{L}(p) \right\} \\
 &= -\underline{L}(\pi) + \frac{1-\pi}{\pi} \cdot \sup_{p \in [0,1]} \left\{ \frac{pz + \pi \cdot \underline{L}(p)}{1-p} \right\}.
 \end{aligned}$$

We see that $(f^\pi)^*$ is unbounded if $z > 0$. Otherwise, we have

$$\begin{aligned}
 \frac{\partial}{\partial p} \left(\frac{pz + \pi \cdot \underline{L}(p)}{1-p} \right) &= \frac{(z + \pi \cdot \underline{L}'(p))(1-p) + (pz + \pi \cdot \underline{L}(p))}{(1-p)^2} \\
 &= \frac{z + \pi \cdot (\underline{L}(p) + (1-p)\underline{L}'(p))}{(1-p)^2} \\
 &= \frac{z + \pi \cdot (-\underline{L}(1) - (-\underline{L}(p)) - (1-p)(-\underline{L})'(p))}{(1-p)^2} \\
 &= \frac{z + \pi \cdot B_{-\underline{L}}(1\|p)}{(1-p)^2} \\
 &= \frac{z + \pi \cdot \ell_1(p)}{(1-p)^2},
 \end{aligned}$$

where we have used the fact that $\underline{L}(1) = \ell_{-1}(0) = 0$ and. Zeroing the derivative, we thus seek $p_\pi(z)$ such that

$$\ell_1(p_\pi(z)) = -\frac{z}{\pi}, \tag{36}$$

and since ℓ is strictly proper, ℓ_1 is invertible and we get

$$p_\pi(z) = \ell_1^{-1} \left(-\frac{z}{\pi} \right), \quad (37)$$

and so

$$\begin{aligned} (f^\pi)^*(z) &= -\underline{L}(\pi) + \frac{1-\pi}{\pi} \cdot \frac{zp_\pi(z) + \pi\underline{L}(p_\pi(z))}{1-p_\pi(z)} \\ &= -\underline{L}(\pi) + (1-\pi) \cdot \frac{-p_\pi(z) \cdot B_{-\underline{L}}(1\|p_\pi(z)) + \underline{L}(p_\pi(z))}{1-p_\pi(z)} \\ &= -\underline{L}(\pi) + (1-\pi) \cdot \frac{-p_\pi(z) \cdot \underline{L}(p_\pi(z)) - p_\pi(z)(1-p_\pi(z)) \cdot \underline{L}'(p_\pi(z)) + \underline{L}(p_\pi(z))}{1-p_\pi(z)} \\ &= -\underline{L}(\pi) + (1-\pi) \cdot (\underline{L}(p_\pi(z)) - p_\pi(z)\underline{L}'(p_\pi(z))) \\ &= -\underline{L}(\pi) + (1-\pi) \cdot (-\underline{L}(0) - (-\underline{L})(p_\pi(z)) - (0-p_\pi(z))(-\underline{L})'(p_\pi(z))) \\ &= -\underline{L}(\pi) + (1-\pi) \cdot B_{-\underline{L}}(0\|p_\pi(z)) \\ &= -\underline{L}(\pi) + (1-\pi) \cdot \ell_{-1}(p_\pi(z)) \\ &= -\underline{L}(\pi) + (1-\pi) \cdot \ell_{-1} \circ \ell_1^{-1} \left(-\frac{z}{\pi} \right), \end{aligned}$$

This shows the expression of $(f^\pi)^*(z)$. To compute $(f^\pi)^* \circ f^{\pi'}(z)$, letting $\eta \doteq \pi z / (\pi z + 1 - \pi)$, we observe

$$\begin{aligned} -\frac{f^{\pi'}(v)}{\pi} &= \underline{L} \left(\frac{\pi z}{\pi z + 1 - \pi} \right) + \frac{1-\pi}{\pi z + 1 - \pi} \cdot \underline{L}' \left(\frac{\pi z}{\pi z + 1 - \pi} \right) \\ &= \underline{L}(\eta) + (1-\eta) \cdot \underline{L}'(\eta) \\ &= (-\underline{L})(1) - (-\underline{L})(\eta) - (1-\eta) \cdot (-\underline{L})'(\eta) \\ &= B_{-\underline{L}}(1\|\eta) \\ &= \ell_1(\eta), \end{aligned} \quad (38)$$

hence

$$(f^\pi)^* \circ f^{\pi'}(z) = -\underline{L}(\pi) + (1-\pi) \cdot \ell_{-1}(\eta) \quad (39)$$

$$= -\underline{L}(\pi) + (1-\pi) \cdot \ell_{-1} \left(\frac{\pi z}{\pi z + 1 - \pi} \right) \quad (40)$$

$$= -\underline{L}(\pi) + (1-\pi) \cdot \ell_{-1} \left(\frac{1}{1 + \frac{1-\pi}{\pi z}} \right), \quad (41)$$

and we remark that when $z \doteq \varrho$ is a likelihood ratio,

$$\frac{1-\pi}{\pi z} = \rho, \quad (42)$$

a density ratio, as claimed.

I.2. Proof of Lemma 4.7

From the chain rule and change of variable $\eta \doteq 1/(1+\rho)$, we get

$$\ell_{-1}'(\eta) = \frac{d\ell_{-1}(\eta)}{d\eta} \quad (43)$$

$$= \frac{d\rho}{d\eta} \cdot \frac{d\ell_{-1}(1/(1+\rho))}{d\rho} \quad (44)$$

$$= -\frac{1}{\eta^2} \cdot \ell^{\text{DR}'}(\rho) \quad (45)$$

$$= -(1+\rho)^2 \cdot \ell^{\text{DR}'}(\rho). \quad (46)$$

(28) shows that ℓ_1 is decreasing and by symmetry, ℓ_{-1} is increasing. So, $\ell_{-1}'(\eta) \geq 0$ and (46) shows then $\ell^{\text{DR}'}(\rho) \leq 0$, thereby proving ℓ^{DR} is decreasing. We get from (30) after working all parameters using 46,

$$\frac{1}{1+\rho} \cdot -\left(1 + \frac{1}{\rho}\right)^2 \cdot \ell^{\text{DR}'}\left(\frac{1}{\rho}\right) = \frac{\rho}{1+\rho} \cdot -(1+\rho)^2 \cdot \ell^{\text{DR}'}(\rho), \quad (47)$$

which becomes after simplification

$$\ell^{\text{DR}'}\left(\frac{1}{\rho}\right) = \rho^3 \cdot \ell^{\text{DR}'}(\rho), \forall \rho \geq 0. \quad (48)$$

Fix $\rho, \varepsilon > 0$, we also have

$$\ell^{\text{DR}'}\left(\frac{1}{\rho+\varepsilon}\right) = (\rho+\varepsilon)^3 \cdot \ell^{\text{DR}'}(\rho+\varepsilon), \quad (49)$$

From which we get

$$\frac{\ell^{\text{DR}'}(\rho+\varepsilon) - \ell^{\text{DR}'}(\rho)}{\varepsilon} = -\frac{1}{\varepsilon} \cdot \left(\frac{1}{\rho^3} \cdot \ell^{\text{DR}'}\left(\frac{1}{\rho}\right) - \frac{1}{(\rho+\varepsilon)^3} \cdot \ell^{\text{DR}'}\left(\frac{1}{\rho+\varepsilon}\right) \right). \quad (50)$$

Suppose the LHS is ≤ 0 , which indicates a secant with negative slope at the right of ρ . From the RHS, we then get the first inequality of

$$\begin{aligned} \ell^{\text{DR}'}\left(\frac{1}{\rho}\right) &\geq \frac{\rho^3}{(\rho+\varepsilon)^3} \cdot \ell^{\text{DR}'}\left(\frac{1}{\rho+\varepsilon}\right) \\ &\geq \ell^{\text{DR}'}\left(\frac{1}{\rho+\varepsilon}\right) \end{aligned} \quad (51)$$

and the second is due to the fact that $\ell^{\text{DR}'}(.) \leq 0$. Letting $\rho' \doteq 1/(\rho+\varepsilon)$ and $\varepsilon' \doteq \varepsilon/(\rho(\rho+\varepsilon))$, we then get

$$\frac{\ell^{\text{DR}'}(\rho'+\varepsilon') - \ell^{\text{DR}'}(\rho')}{\varepsilon'} \geq 0, \quad (52)$$

which indicates a secant with positive slope at the right of ρ' or equivalently at the left of $1/\rho$. Taking the limits for $\varepsilon \rightarrow 0$, we see that if the right derivative at ρ is negative, then the left derivative at $1/\rho$ is positive. Switching $\varepsilon < 0$ from (50) switches the directional derivatives and shows that if $\ell^{\text{DR}'}$ is not convex in ρ , then it is convex in $1/\rho$.

I.3. Proof of Lemma 4.8

The proof follows from Jensen's inequality: if ℓ^{DR} is convex then we have by definition of $\tilde{\varrho}$:

$$\mathbb{G}_\ell(N_{\bar{\eta}}|\tilde{\varrho}) \doteq \underline{L}(\pi) - (1-\pi) \cdot \mathbb{E}_{N_{\bar{\eta}}} \left[\ell_{-1} \left(\frac{1}{1 + \frac{1-\pi}{\pi} \cdot \frac{dN_{\bar{\eta}}}{dP_{\bar{\eta}}}} \right) \right] \quad (53)$$

$$= \underline{L}(\pi) - (1-\pi) \cdot \mathbb{E}_{N_{\bar{\eta}}} \left[\ell^{\text{DR}} \left(\frac{1-\pi}{\pi} \cdot \frac{dN_{\bar{\eta}}}{dP_{\bar{\eta}}} \right) \right] \quad (54)$$

$$\leq \underline{L}(\pi) - (1-\pi) \cdot \ell^{\text{DR}} \left(\frac{1-\pi}{\pi} \cdot \mathbb{E}_{N_{\bar{\eta}}} \left[\frac{dN_{\bar{\eta}}}{dP_{\bar{\eta}}} \right] \right), \quad (54)$$

and we check that

$$\begin{aligned} \mathbb{E}_{N_{\bar{\eta}}} \left[\frac{dN_{\bar{\eta}}}{dP_{\bar{\eta}}} \right] &= \mathbb{E}_P \left[\left(\frac{dN_{\bar{\eta}}}{dP_{\bar{\eta}}} \right)^2 \right] \\ &= \mathbb{E}_P \left[\left(\frac{dN_{\bar{\eta}}}{dP_{\bar{\eta}}} \right)^2 \right] - 2 \cdot \mathbb{E}_P \left[\frac{dN_{\bar{\eta}} dP_{\bar{\eta}}}{dP_{\bar{\eta}}^2} \right] + \mathbb{E}_P \left[\left(\frac{dP_{\bar{\eta}}}{dP_{\bar{\eta}}} \right)^2 \right] + 1 \\ &= \mathbb{E}_P \left[\left(\frac{dN_{\bar{\eta}} - dP_{\bar{\eta}}}{dP_{\bar{\eta}}} \right)^2 \right] + 1 \\ &= \chi^2(N_{\bar{\eta}} || P_{\bar{\eta}}) + 1, \end{aligned} \quad (55)$$

from which we get

$$\begin{aligned}
 \ell^{\text{DR}}\left(\frac{1-\pi}{\pi} \cdot \mathbb{E}_{N_{\tilde{\eta}}}\left[\frac{dN_{\tilde{\eta}}}{dP_{\tilde{\eta}}}\right]\right) &= \ell^{\text{DR}}\left(\frac{1-\pi}{\pi} \cdot (\chi^2(N_{\tilde{\eta}}||P_{\tilde{\eta}}) + 1)\right) \\
 &= \ell_{-1}\left(\frac{1}{1 + \frac{1-\pi}{\pi} \cdot (\chi^2(N_{\tilde{\eta}}||P_{\tilde{\eta}}) + 1)}\right) \\
 &= \ell_{-1}\left(\frac{\pi}{1 + (1-\pi) \cdot \chi^2(N_{\tilde{\eta}}||P_{\tilde{\eta}})}\right),
 \end{aligned} \tag{56}$$

and obtain the lowerbound of (13) after combining with (54).

I.4. Proof of Theorems 5.1 and 5.3

We start by a simple technical Lemma.

Lemma B. Let $f(u) \doteq au^2 + bu + c$; suppose $f(u) \geq 0, \forall u \in \mathbb{R}$, implying $a \geq 0, 4ac - b^2 \geq 0$. Consider $u^* \doteq \arg \min_{\mathbb{R}} f(u) = -b/(2a)$. Let $Q \doteq (4ac - b^2)/4a$. For any $\varepsilon > 0$ and any $v \in \mathbb{R}$,

$$|v - u^*| \geq \varepsilon \Rightarrow f(u^*) \leq \frac{1}{1 + \varepsilon^2 Q} \cdot f(v). \tag{57}$$

Proof: Fix $\delta \in [0, 1)$. We want to compute the v s such that $f(u^*) \leq (1 - \delta)f(v)$. Equivalently, we want

$$av^2 + bv + \frac{b^2 - 4\delta ac}{4(1 - \delta)a} \geq 0. \tag{58}$$

We have the discriminant

$$\Delta = b^2 - \frac{b^2 - 4\delta ac}{1 - \delta} = \frac{\delta(4ac - b^2)}{1 - \delta}, \tag{59}$$

and we have $\Delta \geq 0$ because of the constraints on f . The v s we seek therefore satisfy

$$|v - u^*| \geq \sqrt{\frac{\delta}{1 - \delta} \cdot \left(\frac{4ac - b^2}{4a}\right)} \tag{60}$$

Solving the RHS = ε for δ thus yields that if $|v - u^*| \geq \varepsilon$ then

$$f(u^*) \leq \frac{1}{1 + \varepsilon^2 Q} \cdot f(v), \tag{61}$$

where Q is defined in the statement of the Lemma. □

We have the expressions related to split of \mathbb{S} using feature X :

$$n_\lambda^0 = n_\lambda^0; \quad n_\lambda^l = n_\lambda^l \cdot \frac{1-p}{1-\tau}; \quad n_\lambda^r = n_\lambda^r \cdot \frac{p}{\tau}, \tag{62}$$

and $n'_\lambda = n_\lambda^0 + n_\lambda^l + n_\lambda^r$, $n_\lambda = n_\lambda^0 + n_\lambda^l + n_\lambda^r$. Figure 4 gives an example of the way the scaling factors are obtained on a simple example. These come from scaling the densities after the split (which partitions further the support) and the computation of the associated Bernoullis $B(p)$ (which defines the stochastic activation of the generative tree). Since this process does not change the way the support is split by the discriminator, the weights – integrals of these densities – are scaled by the same factors as depicted in (62). We recall notations

$$l_\lambda \doteq n_\lambda^0 + \frac{n_\lambda^l}{1-\tau}; \quad r_\lambda \doteq n_\lambda^0 + \frac{n_\lambda^r}{\tau}; \quad \delta_\lambda \doteq \frac{n_\lambda^r}{\tau} - \frac{n_\lambda^l}{1-\tau} = r_\lambda - l_\lambda.$$

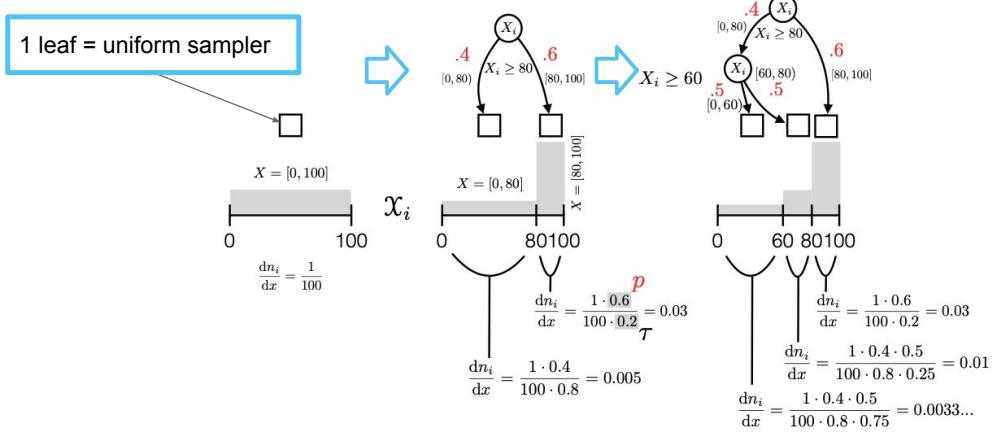


Figure 4. Explanation of (62) on a 1D example: two splits on the same variable in a generative tree creating a piecewise constant but non uniform density for the variable.

For any $p \in [0, 1]$, the new χ^2 after split of \mathbb{S} using feature X admits the simplified expression using (62) and $\mu_{\text{LL}} \doteq \sum_{\lambda \in \Lambda(h)} l_\lambda^2 / p_\lambda$, $\mu_{\text{DD}} \doteq \sum_{\lambda \in \Lambda(h)} \delta_\lambda^2 / p_\lambda$, $\mu_{\text{LD}} \doteq \sum_{\lambda \in \Lambda(h)} l_\lambda \delta_\lambda / p_\lambda$,

$$\begin{aligned} \chi^2(N'_{\bar{\eta}}(p) || P_{\bar{\eta}}) &= \sum_{\lambda \in \Lambda(h)} \frac{(p_\lambda - n_\lambda^0 - n_\lambda^l \cdot \frac{1-p}{1-\tau} - n_\lambda^r \cdot \frac{p}{\tau})^2}{p_\lambda} \\ &= -1 + \sum_{\lambda \in \Lambda(h)} \frac{(n_\lambda^0 + n_\lambda^l \cdot \frac{1-p}{1-\tau} + n_\lambda^r \cdot \frac{p}{\tau})^2}{p_\lambda} \\ &= -1 + \sum_{\lambda \in \Lambda(h)} \frac{(l_\lambda + p\delta_\lambda)^2}{p_\lambda} \end{aligned} \quad (63)$$

$$= -1 + \mu_{\text{LL}} + 2p\mu_{\text{LD}} + p^2\mu_{\text{DD}}, \quad (64)$$

where we put p in parameter of $N'_{\bar{\eta}}(\cdot)$. Note we can also take a fork at (65) and write instead:

$$\chi^2(N'_{\bar{\eta}}(p) || P_{\bar{\eta}}) = -1 + \sum_{\lambda \in \Lambda(h)} \frac{(r_\lambda - (1-p)\delta_\lambda)^2}{p_\lambda} \quad (65)$$

$$= -1 + \mu_{\text{RR}} - 2(1-p)\mu_{\text{RD}} + (1-p)^2\mu_{\text{DD}}. \quad (66)$$

Three values of p are of interest:

- for $p = \tau$, we get

$$\chi^2(N'_{\bar{\eta}}(\tau) || P_{\bar{\eta}}) = \chi^2(N_{\bar{\eta}} || P_{\bar{\eta}}), \quad (67)$$

and there is no change in χ^2 after split.

- for $p = 0$, we get

$$\chi^2(N'_{\bar{\eta}}(0) || P_{\bar{\eta}}) = -1 + \mu_{\text{LL}}, \quad (68)$$

which corresponds to discarding support on X^r and yields $n'_\lambda = l_\lambda$;

- for $p = 1$, we get

$$\chi^2(N'_{\bar{\eta}}(1) || P_{\bar{\eta}}) = -1 + \mu_{\text{RR}}, \quad (69)$$

which corresponds to discarding support on X^l and yields $n'_\lambda = r_\lambda$.

Case 1 – suppose $\mu_{DD} > 0$. Define f as in Lemma B using (64), with $a \doteq \mu_{DD}, b \doteq 2\mu_{LD}, c \doteq \mu_{LL} - 1$. We have

$$u^* = -\frac{\mu_{LD}}{\mu_{DD}} = \frac{\mu_{LL} - \mu_{LR}}{\mu_{LL} + \mu_{RR} - 2\mu_{LR}} \quad (70)$$

Case 1.1 – suppose in addition $\mu_{LL} \geq \mu_{LR}$ and $\mu_{RR} \geq \mu_{LR}$. We have $u^* \in [0, 1]$ and fix $p \doteq u^*$. For the choice $v = \tau$, Lemma B says that

$$|\tau - p| \geq \varepsilon \Rightarrow \chi^2(N'_{\tilde{\eta}}(p) || P_{\tilde{\eta}}) \leq \frac{1}{1 + \varepsilon^2 Q} \cdot \chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}}), \quad (71)$$

with

$$Q = \frac{\mu_{DD}(\mu_{LL} - 1) - \mu_{LD}^2}{\mu_{DD}} = -1 + \mu_{LL} - \frac{\mu_{LD}^2}{\mu_{DD}}. \quad (72)$$

We also remark that with the value of p as in (70), $\chi^2(N'_{\tilde{\eta}}(p) || P_{\tilde{\eta}})$ turns out to be

$$\chi^2(N'_{\tilde{\eta}}(p) || P_{\tilde{\eta}}) = -1 + \mu_{LL} - 2 \cdot \frac{\mu_{LD}}{\mu_{DD}} \cdot \mu_{LD} + \left(\frac{\mu_{LD}}{\mu_{DD}} \right)^2 \mu_{DD} = Q. \quad (73)$$

Hence, for any $\delta > 0$, as long as $\chi^2(N'_{\tilde{\eta}}(. || P_{\tilde{\eta}})) \geq \delta$, whenever $|\tau - p| \geq \varepsilon$, one step of TD-GEN achieves geometric convergence with rate $1/(1 + \delta\varepsilon^2)$.

Case 1.2 – suppose now $\mu_{LL} < \mu_{LR}$, which implies $u^* < 0$. Pick $p = 0$. From (68), (67) and (64), to get $\chi^2(N'_{\tilde{\eta}}(0) || P_{\tilde{\eta}}) \leq (1/(1 + \varepsilon)) \cdot \chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}})$, we equivalently need

$$\tau^2 \mu_{DD} + 2\tau \mu_{LD} - \varepsilon \mu_{LL} + \varepsilon \geq 0, \quad (74)$$

which, expressed using the fact that $\mu_{DD} = \mu_{RR} + \mu_{LL} - 2\mu_{LR}$ and $\mu_{LD} = \mu_{LR} - \mu_{LL}$, yields:

$$(\tau^2 - 2\tau - \varepsilon) \mu_{LL} + \tau^2 \mu_{RR} + 2\tau(1 - \tau) \mu_{LR} + \varepsilon \geq 0, \quad (75)$$

The Weak Generative Assumption, $\mu_{DD} \geq \delta \cdot \max\{\mu_{LL}, \mu_{RR}\}$ implies $\mu_{RR} \geq 2\mu_{LR} - \mu_{LL} + \delta \cdot \mu_{LL}$, and with Case 1.2's assumption, $\mu_{LR} > \mu_{LL}$, yields $\mu_{RR} \geq 2\mu_{LL} - \mu_{LL} + \delta \cdot \mu_{LL} = (1 + \delta) \cdot \mu_{LL}$, so we get

$$\begin{aligned} (\tau^2 - 2\tau - \varepsilon) \mu_{LL} + \tau^2 \mu_{RR} + 2\tau(1 - \tau) \mu_{LR} + \varepsilon &> (\tau^2 - 2\tau - \varepsilon) \mu_{LL} + (1 + \delta) \tau^2 \cdot \mu_{LL} \\ &\quad + 2\tau(1 - \tau) \mu_{LL} + \varepsilon \\ &= (\delta \tau^2 - \varepsilon) \cdot \mu_{LL} + \varepsilon. \end{aligned} \quad (76)$$

Fixing $\varepsilon = \delta \tau^2$ thus brings (74) and we conclude with

$$\chi^2(N'_{\tilde{\eta}}(0) || P_{\tilde{\eta}}) \leq \frac{1}{1 + \delta \tau^2} \cdot \chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}}). \quad (77)$$

Case 1.3 – suppose now $\mu_{RR} < \mu_{LR}$, which implies $u^* > 1$ (the denominator of u^* is always ≥ 0). Pick $p = 1$. From (69), (67) and (66), to get $\chi^2(N'_{\tilde{\eta}}(1) || P_{\tilde{\eta}}) \leq (1/(1 + \varepsilon)) \cdot \chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}})$, we equivalently need

$$(1 - \tau)^2 \mu_{DD} - 2(1 - \tau) \mu_{RD} - \varepsilon \mu_{RR} + \varepsilon \geq 0. \quad (78)$$

Using $\mu_{DD} = \mu_{RR} + \mu_{LL} - 2\mu_{LR}$ and $\mu_{RD} = \mu_{RR} - \mu_{LR}$, we break this down to:

$$(1 - \tau)^2 \mu_{LL} - (1 - \tau^2 + \varepsilon) \mu_{RR} + 2\tau(1 - \tau) \mu_{LR} + \varepsilon \geq 0, \quad (79)$$

The Weak Generative Assumption yields this time $\mu_{LL} \geq 2\mu_{LR} - \mu_{RR} + \delta \cdot \mu_{RR}$, which, together with Case 1.3's assumption, $\mu_{LR} > \mu_{RR}$, yields $\mu_{LL} \geq (1 + \delta) \cdot \mu_{RR}$, so we get this time

$$\begin{aligned} (1 - \tau)^2 \mu_{LL} - (1 - \tau^2 + \varepsilon) \mu_{RR} + 2\tau(1 - \tau) \mu_{LR} + \varepsilon &> (1 - \tau)^2 (1 + \delta) \cdot \mu_{RR} - (1 - \tau^2 + \varepsilon) \mu_{RR} \\ &\quad + 2\tau(1 - \tau) \mu_{RR} + \varepsilon \\ &= (\delta(1 - \tau)^2 - \varepsilon) \cdot \mu_{RR} + \varepsilon. \end{aligned} \quad (80)$$

Fixing $\varepsilon = \delta(1 - \tau)^2$ thus brings (74) and we conclude with

$$\chi^2(N'_{\tilde{\eta}}(0) || P_{\tilde{\eta}}) \leq \frac{1}{1 + \delta(1 - \tau)^2} \cdot \chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}}). \quad (81)$$

Cases 1.2 and 1.3 can be summarised as

$$\chi^2(N'_{\tilde{\eta}}(p) || P_{\tilde{\eta}}) \leq \frac{1}{1 + \delta(\tau + (1 - 2\tau)p)^2} \cdot \chi^2(N_{\tilde{\eta}} || P_{\tilde{\eta}}), p \in \{0, 1\}. \quad (82)$$

Case 2 – suppose $\mu_{DD} = 0$. We remark in this case that the optimal p to minimize $\chi^2(N'_{\tilde{\eta}}(p) || P_{\tilde{\eta}})$ as in (64) or (66) is in $\{0, 1\}$, which brings us to the case where the Weak Generative Assumption holds and therefore make that Case 2 does not happen for the analysis of TD-GEN.

I.5. Proof of Lemma 5.6

We note that we have for any $\lambda \in \Lambda(h_T)$ and x reaching λ ,

$$\frac{\pi dP^\top}{\pi dP^\top + (1 - \pi)dU}(x) = \frac{\pi p_\lambda}{\pi p_\lambda + (1 - \pi)u_\lambda}, \quad (83)$$

where we recall $p_\lambda \doteq \int_\lambda dP$ and $u_\lambda \doteq \int_\lambda dU$. We get from the scaled Bregman Theorem (Nock et al., 2016, Theorem 1) that since g is affine, the perspective transform $(\widetilde{-L})$ is convex and the first equality holds in:

$$\begin{aligned} & \pi \cdot \mathbb{E}_U \left[B_{(\widetilde{-L})} \left(\frac{dP}{dU} \middle\| \frac{dP^\top}{dU} \right) \right] \\ &= \pi \cdot \int_{\mathcal{X}} g \left(\frac{dP}{dU} \right) \cdot B_{-\underline{L}} \left(\frac{\frac{dP}{dU}}{g(\frac{dP}{dU})} \middle\| \frac{\frac{dP^\top}{dU}}{g(\frac{dP^\top}{dU})} \right) \cdot dU \\ &= \underbrace{\int_{\mathcal{X}} (\pi dP + (1 - \pi)dU) \cdot (-\underline{L}) \left(\frac{\pi dP}{\pi dP + (1 - \pi)dU} \right)}_{\doteq A} \\ &\quad - \underbrace{\int_{\mathcal{X}} (\pi dP + (1 - \pi)dU) \cdot (-\underline{L}) \left(\frac{\pi dP^\top}{\pi dP^\top + (1 - \pi)dU} \right)}_{\doteq B} \\ &\quad - \underbrace{\int_{\mathcal{X}} (\pi dP + (1 - \pi)dU) \cdot \left(\frac{\pi dP}{\pi dP + (1 - \pi)dU} - \frac{\pi dP^\top}{\pi dP^\top + (1 - \pi)dU} \right) \cdot (-\underline{L})' \left(\frac{\pi dP^\top}{\pi dP^\top + (1 - \pi)dU} \right)}_{\doteq C}, \end{aligned}$$

and the second equality follows from the definition of Bregman divergences. Since leaves in h_T induce a partition of \mathcal{X} , C simplifies to:

$$\begin{aligned} C &= \sum_{\lambda \in \Lambda(h_T)} (-\underline{L})' \left(\frac{\pi p_\lambda}{\pi p_\lambda + (1 - \pi)u_\lambda} \right) \cdot \left(\pi \int_\lambda dP - \frac{\pi p_\lambda}{\pi p_\lambda + (1 - \pi)u_\lambda} \cdot \int_\lambda (\pi dP + (1 - \pi)dU) \right) \\ &= \sum_{\lambda \in \Lambda(h_T)} (-\underline{L})' \left(\frac{\pi p_\lambda}{\pi p_\lambda + (1 - \pi)u_\lambda} \right) \cdot \underbrace{\left(\pi \int_\lambda dP - \pi p_\lambda \right)}_{=0} \\ &= 0. \end{aligned}$$

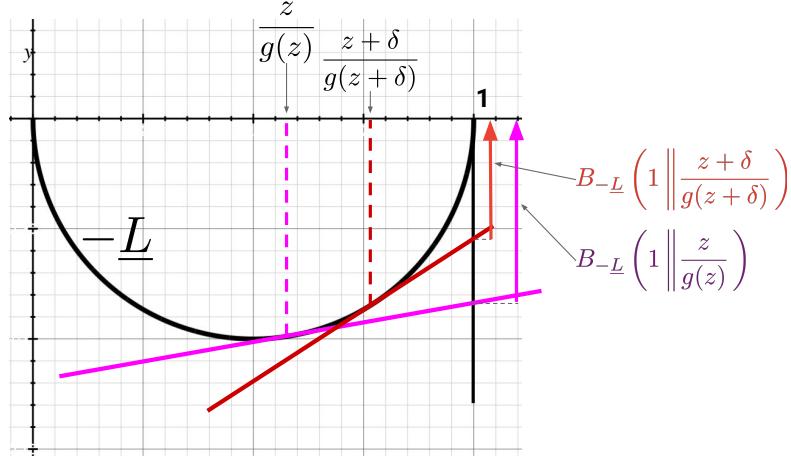


Figure 5. Depiction of the two Bregman divergences in (85).

We can also reformulate $A - B$:

$$\begin{aligned} A - B &= \underline{L}(\pi) - \int_{\mathcal{X}} (\pi dP + (1 - \pi)dU) \cdot \underline{L}\left(\frac{\pi dP}{\pi dP + (1 - \pi)dU}\right) \\ &\quad - \left(\underline{L}(\pi) - \sum_{\lambda \in \Lambda(h_T)} (\pi p_\lambda + (1 - \pi)u_\lambda) \cdot \underline{L}\left(\frac{\pi p_\lambda}{\pi p_\lambda + (1 - \pi)u_\lambda}\right) \right) \\ &= \mathbb{I}_{f^\pi}(P, U) - \mathbb{I}_{f^\pi}(P_{\bar{\eta}_T}, U_{\bar{\eta}_T}), \end{aligned}$$

which leads to the statement of the Lemma.

Remark: there is a simple argument to show the strict convexity of $(\widetilde{-L})$ that relates its derivative to negative a Bregman divergence:

$$\begin{aligned} (\widetilde{-L})'(z) &= (-L)\left(\frac{z}{g(z)}\right) + \left(1 - \frac{z}{g(z)}\right) \cdot (-L)' \left(\frac{z}{g(z)}\right) \\ &= (-L)(1) - \left((-L)(1) - (-L)\left(\frac{z}{g(z)}\right) - \left(1 - \frac{z}{g(z)}\right) \cdot (-L)' \left(\frac{z}{g(z)}\right)\right) \\ &= -B_{-L}\left(1 \middle\| \frac{z}{g(z)}\right), \end{aligned}$$

because in our case we have $(-L)(1) = 0$. Let for short $g(z) \doteq z + K$ for $K > 0$. We have $g(z) > z$ and so for any $\delta > 0$,

$$\frac{z + \delta}{g(z + \delta)} = \frac{z + \delta}{g(z) + \delta} > \frac{z}{g(z)}, \quad (84)$$

and since L is strictly concave and $z/g(z) < 1$ for $z > 0$, Bregman divergences lead to:

$$B_{-L}\left(1 \middle\| \frac{z + \delta}{g(z + \delta)}\right) < B_{-L}\left(1 \middle\| \frac{z}{g(z)}\right) \quad (85)$$

(See Figure 5 for a depiction of the quantities), hence $(\widetilde{-L})'(z + \delta) > (\widetilde{-L})'(z)$, showing the derivative of the perspective transform of negative the pointwise Bayes risk is strictly increasing and the function is therefore strictly convex.

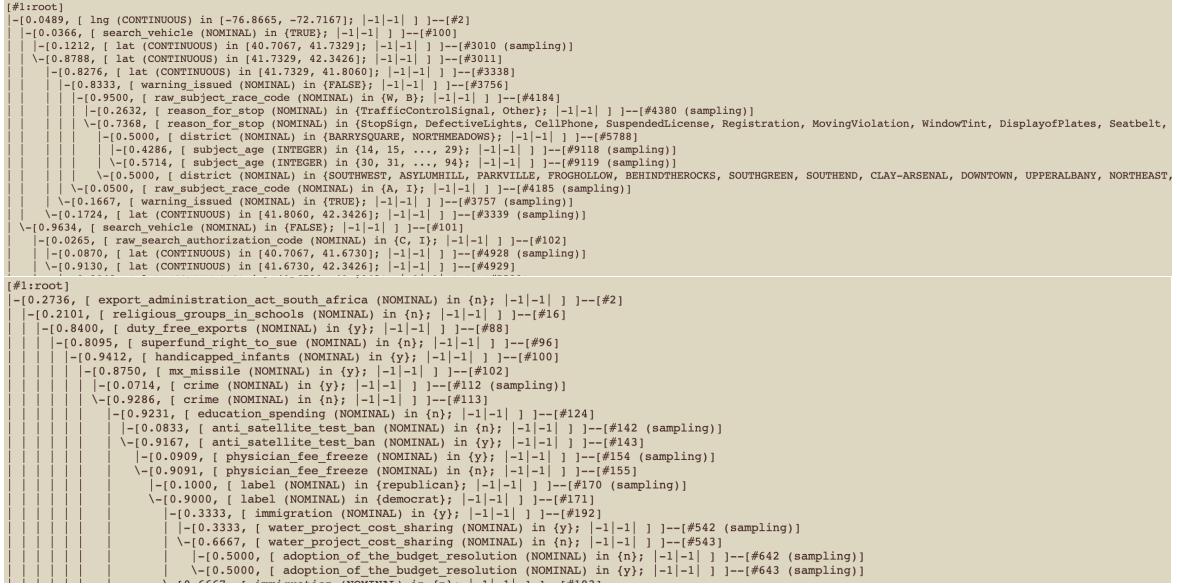


Figure 6. Generative trees (crop) learned on a fold of Stanford open policing data for Hartford (top) and UCI house votes (bottom).

II. Appendix on experiments

II.1. Examples of generative trees

Figure 6 provides examples of subsets of generative trees learned on Stanford open policing data and UCI house votes. Each node takes the form

```
[prob value, [variable (nominal) in {set of nominal values}]; ... ]--[#node name]
[prob value, [variable (continuous) in [continuous interval]]; ... ]--[#node name]
[prob value, [variable (integer) in {int value n, n+1, ..., m}]; ... ]--[#node name]
```

prob value is the Bernoulli probability associated to the arc pointing to the node. If a node appears as

```
[ ... ]--[#node name (sampling)]
```

then it is a leaf (sampling) node. The rest of the Figures should be self-explanatory. In open policing, notice sampling nodes #9118, #9119, inducing a higher probability of sampling a young person (age within 14 and 29) in the related part of the domain.

II.2. Domains

ringGauss is the seminal 2D ring Gaussians appearing in numerous GAN papers (Xiao et al., 2018); those are eight (8) spherical Gaussians with equal covariance, sampling size and centers located on sightlines regularly spaced (2-2 angular distance) and at equal distance from the origin. gridGauss was generated as a decently hard task from (Dumoulin et al., 2017): it consists of 25 2D mixture spherical Gaussians with equal variance and sampled sizes, put on a regular grid. circGauss is a Gaussian mode surrounded by a circle, from (Xiao et al., 2018). randGauss is a substantially harder version of ringGauss with 16 mixture components, in which covariance, sampling sizes and distances on sightlines from the origin are all random, which creates very substantial discrepancies between modes.

II.3. Algorithms configuration and choice of parameters

GTs We only report comparisons with the copycat approach in which the discriminator is Kearns & Mansour (1996)'s greedy induction algorithm optimising Matusita's loss; our implementation is in Java. The input of our algorithm to train a generator is a .csv file containing the training data *without any further information*. In particular, each feature's domain is learned from the training data only; while this could surely and trivially be replaced by a user-informed domain for improved

Domain	Source	Missing data ?	m	d	# Nom.	# Num.
iris	UCI	No	150	5	1	4
*ringGauss	—	No	1 600	2	—	2
*circGauss	—	No	2 200	2	—	2
*gridGauss	—	No	2 500	2	—	2
house-votes'84	UCI	Yes	435	16	16	—
*randGauss	—	No	3 800	2	—	2
led	UCI	No	1 000	8	—	8
tictactoe	UCI	No	958	9	9	—
winered	UCI	No	1 599	12	1	11
led24	—	No	1 000	25	—	25
abalone	UCI	No	4 177	9	1	3
winewhite	UCI	No	4 898	12	1	11
sigma-cabs	Kaggle	Yes	5 000	13	5	8
open-policing	SOP**	Yes	18 419	20	16	4
dna	UCI	No	3 186	181	181	—

Table A5. Public domains considered in our experiments (m = total number of examples, d = number of features), ordered in increasing $m \times d$. "Nom." is a shorthand for nominal / ordinal / binary; "Num." stands for integers / reals. (*) = simulated, (**) = Hartford data from the Stanford Open Policing Project, <https://openpolicing.stanford.edu/>) (see text).

results (e.g. indicating a proportion's domain as [0%, 100%], informing the complete list of socio-professional categories, etc.) — and is in fact standard in some ML packages like weka's ARFF files, we did not pick this option to alleviate all side information available to the GT learner. Technical details are:

1. features' domains are computed from training data; our software automatically recognizes three types of variables: nominal, integer and floating point represented³;
2. the discriminator's training follows the top-down induction blueprint in Kearns & Mansour (1996);
3. we do not accept splits that will incur a branching probability $p \in \{0, 1\}$ to prevent discarding support; leaves are split from the heaviest first; when a real examples with missing values branch in the decision tree on one of its missing values, the probability of following the left / right arc is *not* 1/2 but takes into account the length of the left / right domains of the variable at the split (for example, if the left branch's variable domain is $\{A, B, C\}$ and the right one is $\{D\}$ for a nominal variable, then the left branching probability is 3/4);
4. we consider three basic sizes of GTs, corresponding to 10, 300 and a MAX = 10 000 nodes splits (thus with total number of nodes equal to 21, 601 and no more than 20 001). In the IMPUTE experiment, we only use the MAX size. Note that the MAX size usually is smaller than 20 001 on small datasets because of the support constraint in [2].

mice We have used the R mice package V 3.13.0 with three choices of methods for the round robin (column-wise) prediction of missing values: cart (Breiman et al., 1984), norm and random forests (rf) (van Buuren & Groothuis-Oudshoorn, 2011). In that last case, we have replaced the default number of trees (10) by a larger number (100) to get better results. We use the default number of round-robin iterations (5).

CTGAN We have used the Python implementation⁴ with default values.

TENSORFLOW To learn the additional Random Forests and Gradient Boosted Decision Trees involved in experiments TRAIN-GEN, GEN-DISCRIM and GEN-AUG, we used Tensorflow Decision Forests library⁵. The important points are:

³This is a difference with mice for which categorical variables need to be explicitly stated.

⁴<https://github.com/sdv-dev/CTGAN>

⁵<https://github.com/google/yggdrasil-decision-forests/blob/main/documentation/learners.md>

- for Random Forests, we use 300 trees with max depth 16. Attribute sampling: $\text{sqrt}(\text{number attributes})$ for classification problems, $\text{number attributes} / 3$ for regression problems (Breiman rule of thumb);
- for Gradient Boosted Decision Trees, we use max 300 trees, with 10% of the training dataset for validation and early stopping. Max depth is 6 and there is no attribute sampling;
- in both cases, the min #examples per leaf is 5, we use CART to find splits on numerical and categorical features (*i.e.* we don't use one-hot encoding for categorical features). Induction is top-down.

Computers used We ran part of the experiments on a Mac Book Pro 16 Gb RAM w/ 2 GHz Quad-Core Intel(R) Core i5(R) processor, and part on a desktop Intel(R) Xeon(R) 3.70GHz with 12 cores and 64 Gb RAM.

II.4. Data generation experiments

Figures 7, 8, 9, 10, 11, 12, 13, 14, 15 present 2D heatmap of density learned by generative trees with shown total number of nodes. We have run a simple 10-fold CV experiment, each plot being the generator that minimizes over all folds an empirical χ^2 between the training data and a set of generated data. For UCI domains, the variables plotted are indicated and we indicate the number (m') of examples generated to build the plots.

Warning: each of those plots records 1-of-10 results minimizing an empirical χ^2 given the current size of the GT, so the related GT could be *different* between any two iterations: sudden changes in densities do not stem from any instability of training GTs but just from different GTs being selected. For example, in Figure 8, the GTs kept in the pairs of $\#\text{nodes} = 7$ and 11, or $\#\text{nodes} = 35$ and 39, or $\#\text{nodes} = 55$ and 59, are *different* from each other (*e.g.* densities do not suddenly rotate by $\pm 90^\circ$ after few iterations).

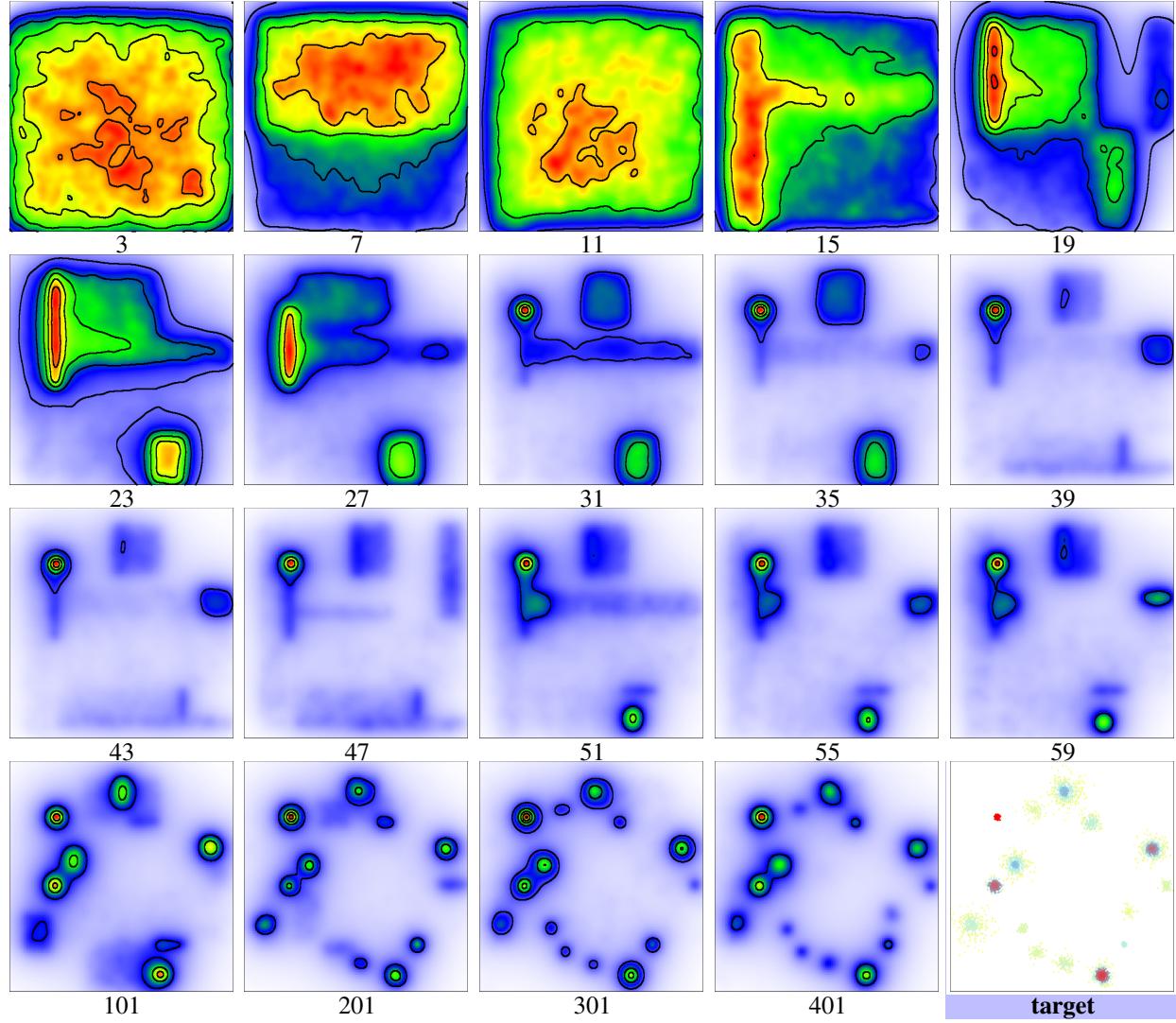


Figure 7. Results on the `randGauss` simulated data (target in the bottom-right, $m = 3800$; colors indicate sampled density), for copycat training. Numbers are the total number of nodes of the generators; generators sampled for $m' = 4000$ points each, each plot shows results for one of the ten generators in the CV folds (not necessarily from the same fold); training method = copycat.

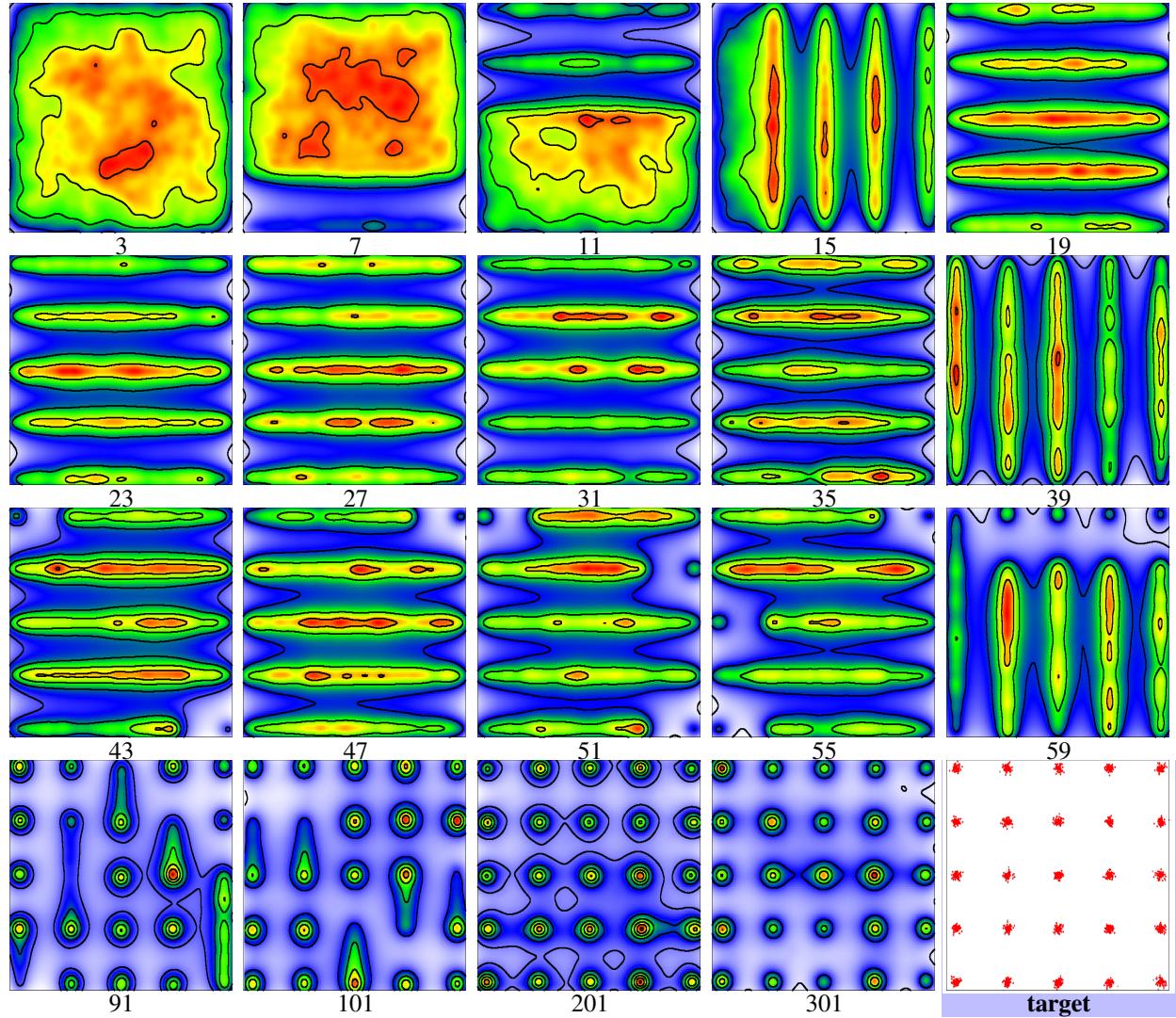


Figure 8. Results on the `gridGauss` simulated data (with $m = 2500$, $m' = 4000$), convention follows Fig. 7. Check the warning in Section II.4 for an accurate interpretation of the plots (any two can refer to *different* generative trees).

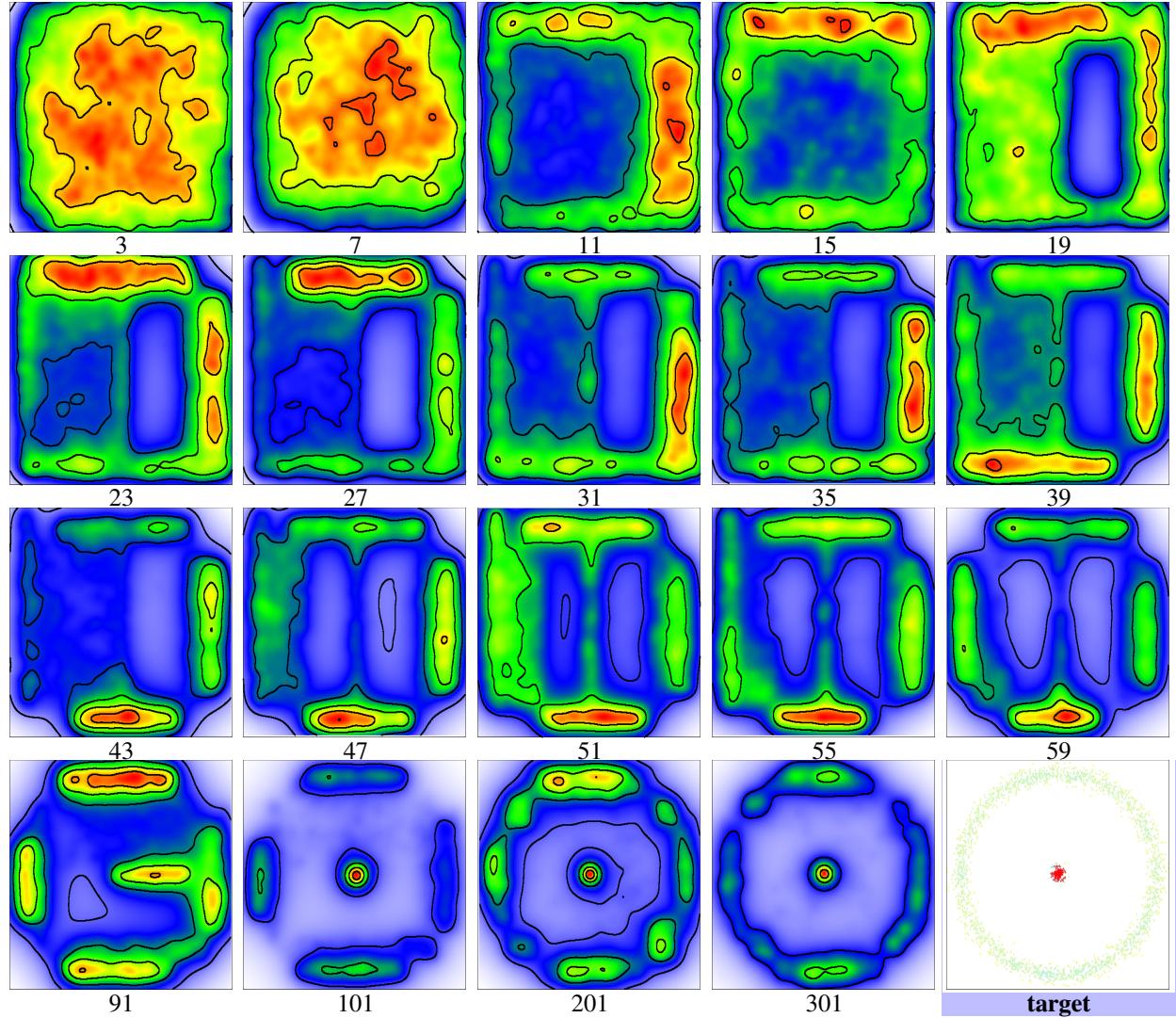


Figure 9. Results on the `circGauss` simulated data (with $m = m' = 2200$), convention follows Fig. 7.

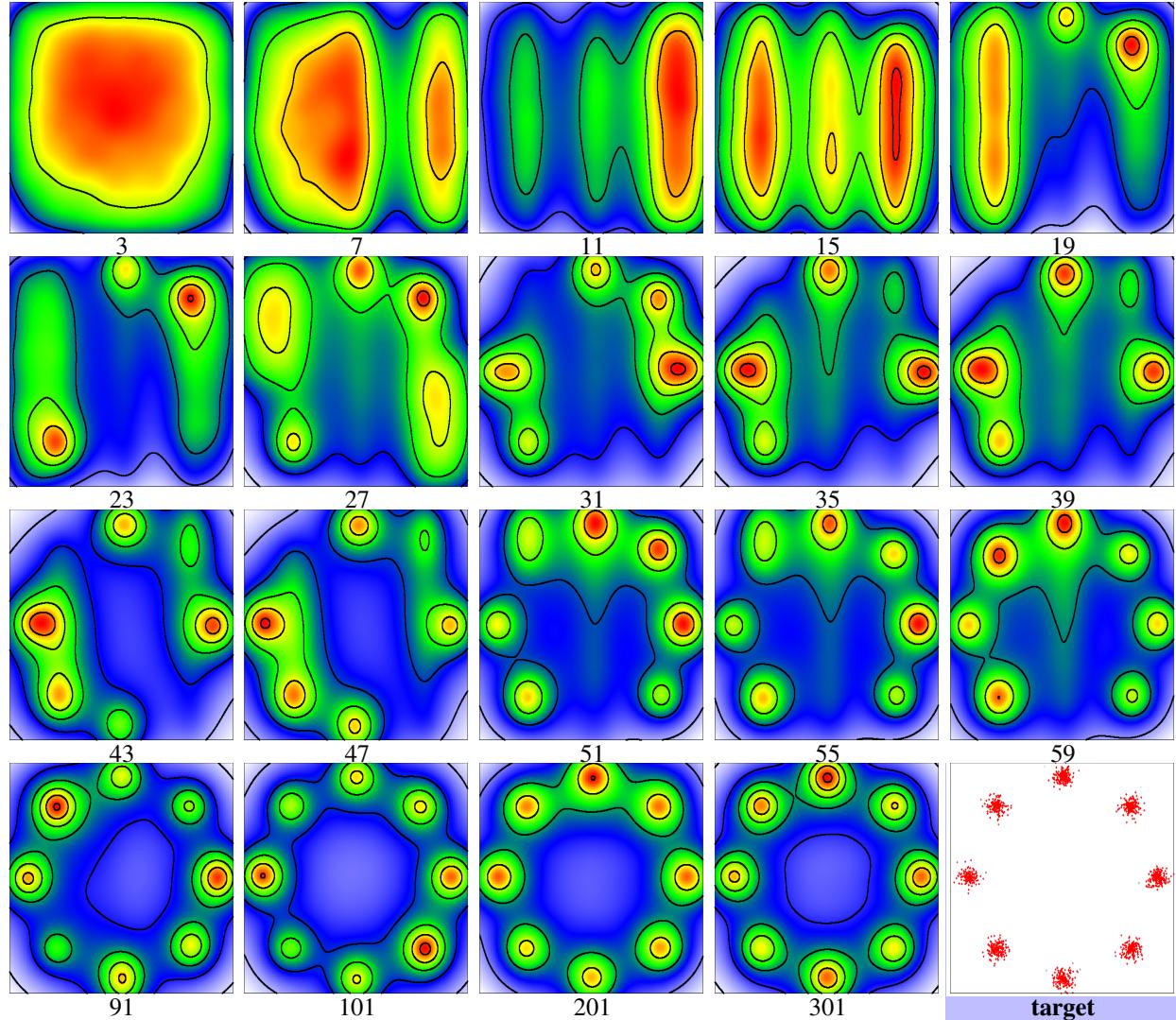


Figure 10. Results on the `ringGauss` simulated data (with $m = 1500, m' = 4000$), convention follows Fig. 7.

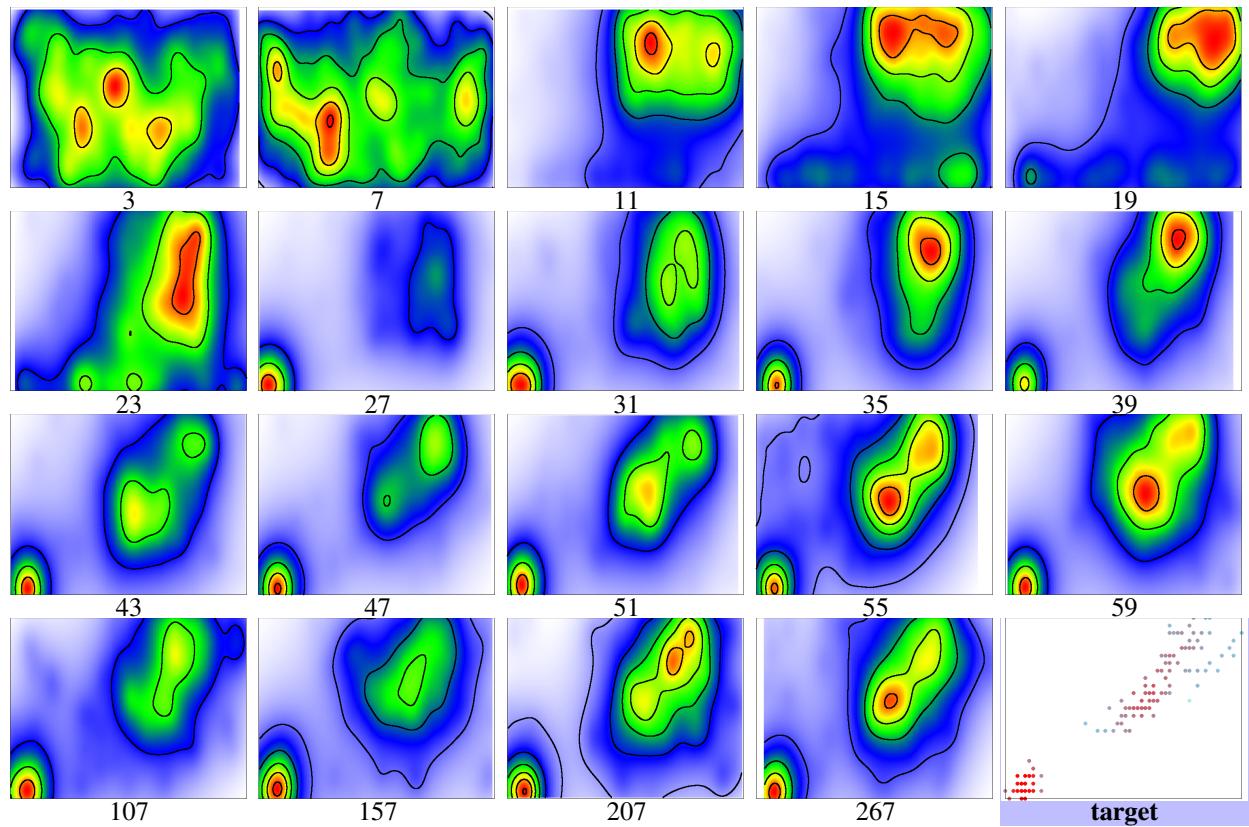


Figure 11. Results on UCI *iris* domain (with $m = 150, m' = 150$) for the 2D plane petal-length \times petal-width, convention follows Fig. 7.

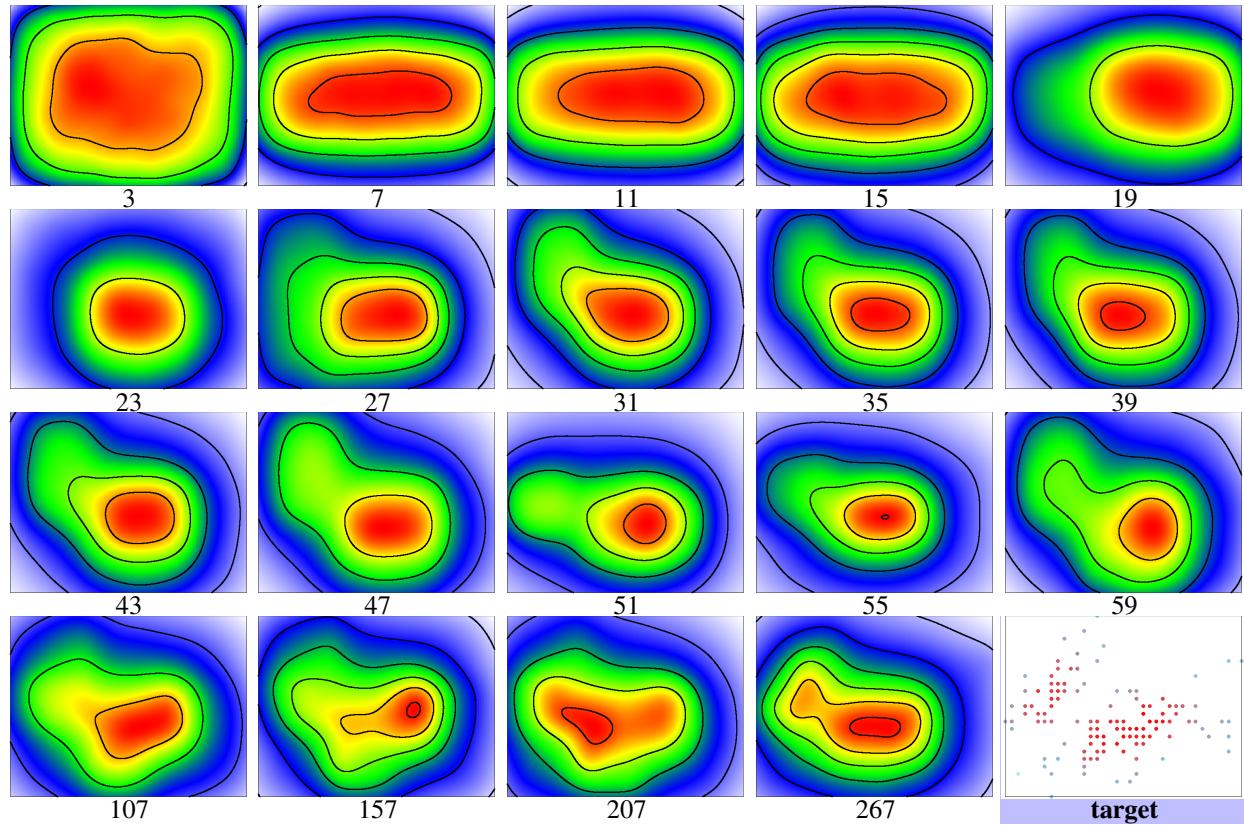


Figure 12. Results on UCI `iris` domain (with $m = 150$, $m' = 4000$) for the 2D plane `sepal-length` \times `sepal-width`, convention follows Fig. 7.

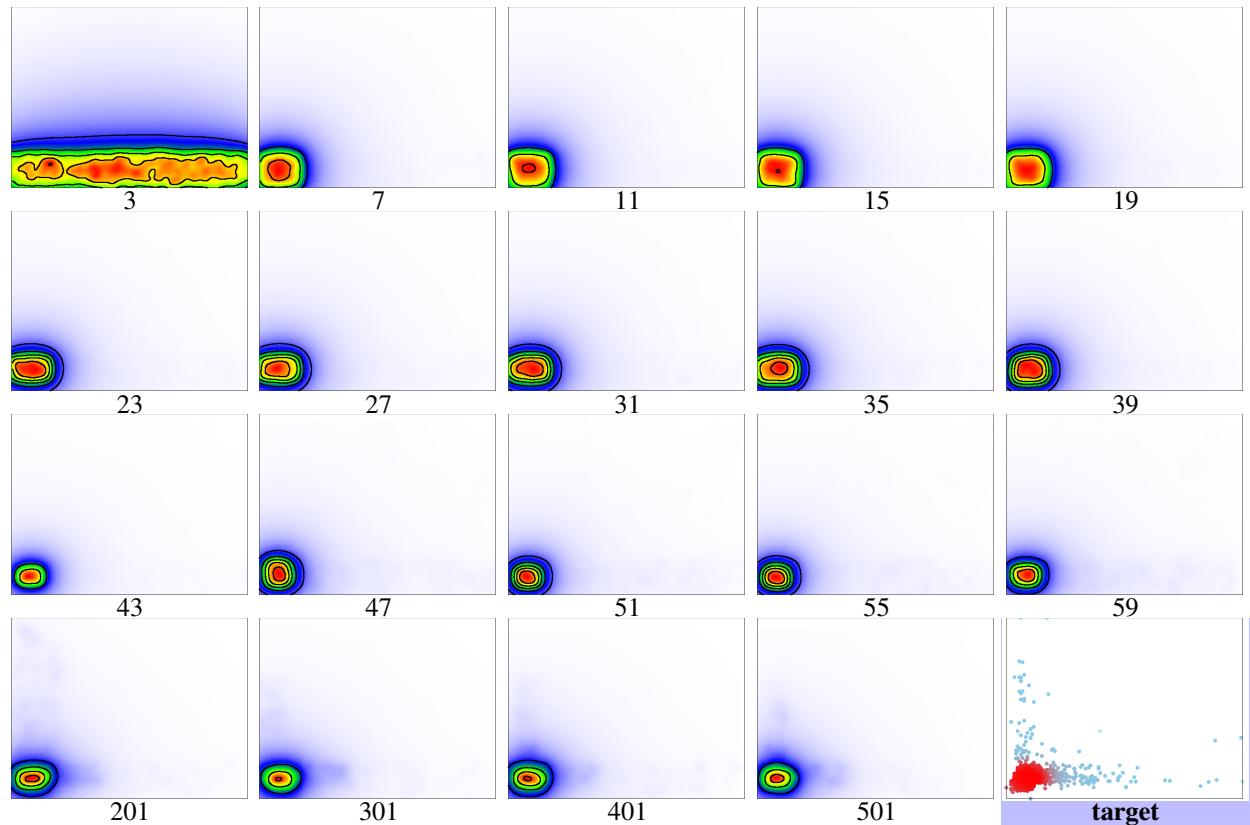


Figure 13. Results on UCI wine-red domain (with $m = m' = 1599$) for the 2D plane residual-sugar \times chlorides, convention follows Fig. 7.

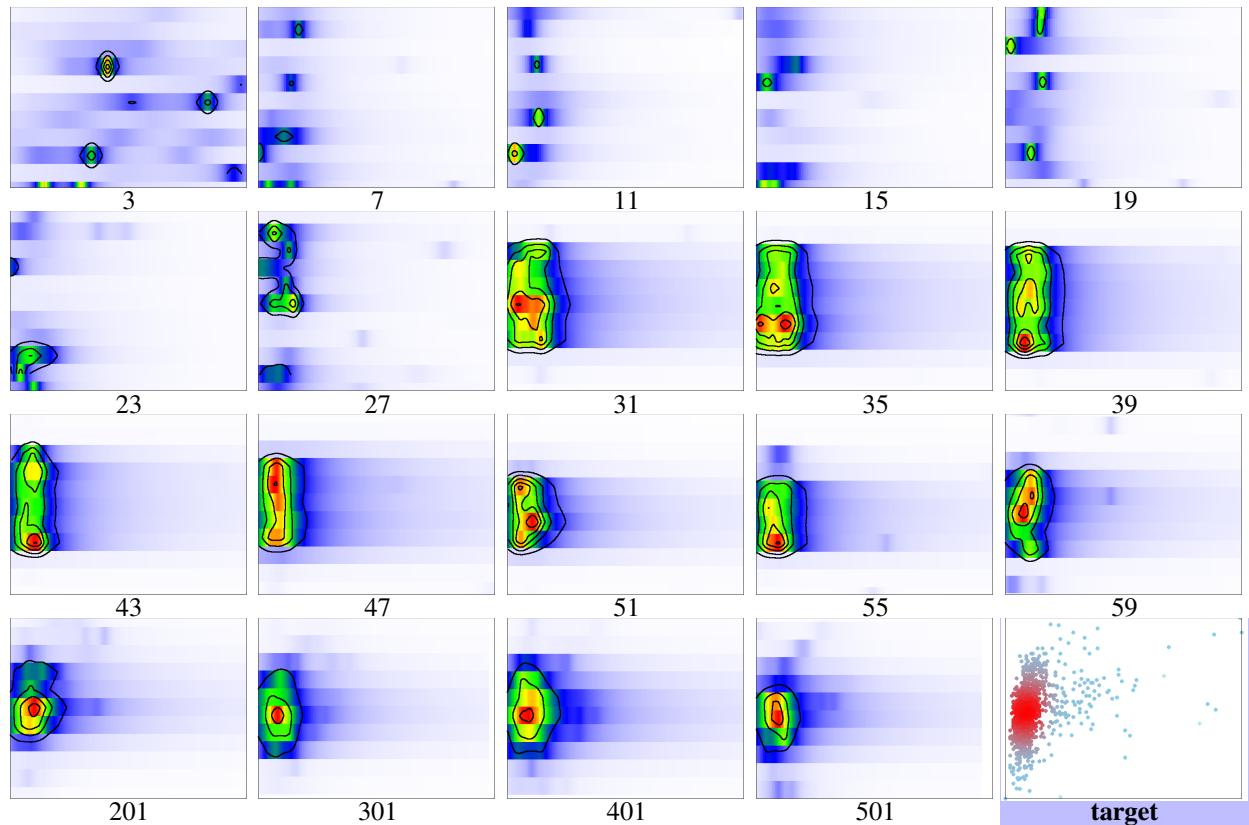


Figure 14. Results on UCI wine-red domain (with $m = m' = 1599$) for the 2D plane residual-sugar \times density, convention follows Fig. 7. The discontinuous strip-look is due to the huge difference in scales for the variables.

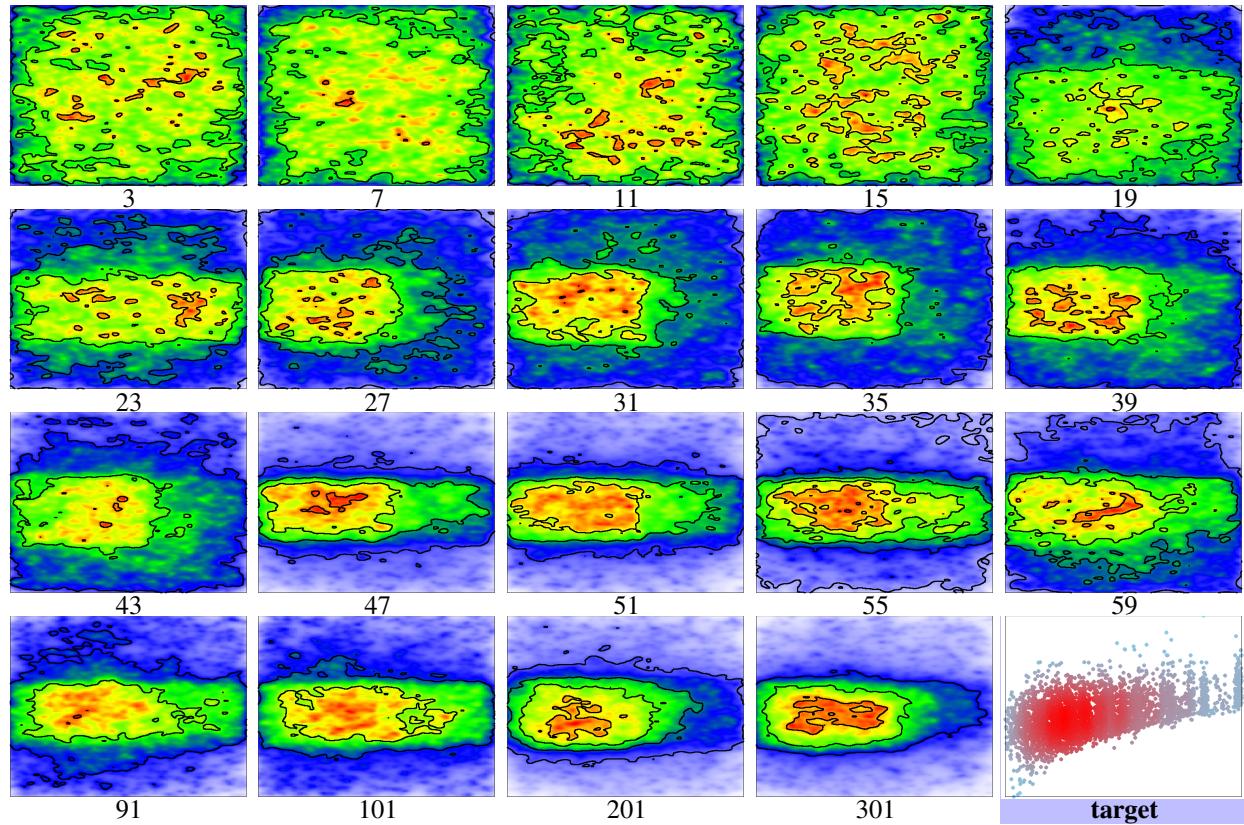


Figure 15. Results on UCI sigma-cabs domain (with $m = m' = 5000$) for the 2D plane trip-distance \times life-style-index, convention follows Fig. 7.

II.5. Missing data imputation experiments (IMPUTE)

Objective Imputing missing values in data is an important process (Muzellec et al., 2020). Classically, imputation methods are *specifically* designed for the task (van Buuren, 2018), even when they rely on generative models (Yoon et al., 2018). In our case, a general purpose GT G can trivially be used for missing data imputation: we constrain the support of the tree to the observed variables and then sample in the region(s) of maximal density, for a process that takes no more than $\mathcal{O}(|\Lambda(G)|)$ per observation. We have tested this simple procedure against the state of the art tree-based methods in the `mice` R package (van Buuren & Groothuis-Oudshoorn, 2011) (and one non-tree based but known to be a good fit for normal data). The methods we have considered in `mice` all have a commonpoint: they carry out round-robin imputation (Muzellec et al., 2020). After having imputed the missing values with an initial guess, they circle round the attributes, repeatedly updating the revision of one attribute by predicting it from all the current others. Notice the potentially huge number of classifiers used. In our case, on a domain like `dna` with 181 variables, the default number of iterations (5) with random forests of 100 trees each means imputing a dataset necessitates no less than **90 500** trees. In comparison, we rely on a **single** tree-based model to simultaneously impute all values. In particular on such domains, we cannot hope to beat such approaches, but our approach was rather to compare with SOTA over ranges of problem complexity, variable diversity and have specific simulated domains to further scrutinise differences for GTs used as ‘basic’ components of imputation methods.

Experimental setting We consider copycat training against a discriminator minimizing Matusita’s loss (Kearns & Mansour, 1996). We grow the GT to a MAX size (with limit 10 000 nodes, see Section II.6) and prevent splits with $p \in \{0, 1\}$, thus avoiding discarding support for data generation. For each domain, we generate data that is Missing Completely At Random (MCAR, van Buuren (2018)) by removing a fixed proportion of modalities $q \in \{5\%, 10\%, 20\%, 50\%\}$, embedded in a 5-fold cross validation for each q . In each fold, we thus impute a complete dataset and compare the resulting imputations to the observed values. In such a setting, classical per-observation metrics like RMSE are not necessarily the best choices: if after removing MCAR features two observations were then the same for the resulting features, a perfect imputation of the missing values resulting in a permutation of the observations in the dataset would incur non-zero RMSE, yet would arguably be correct. Similarly to Muzellec et al. (2020), we have thus opted for an optimal transport metric, Wasserstein’s W_2^2 . We use `mice` with method oracles in {CART, NORM, RANDOM FORESTS (RF) (100 trees)}. NORM is not tree-based but a good alternative on normal data (van Buuren, 2018). Notice that we have not used the default number of trees for random forests (10), which we considered too small for our purpose. Due to the difficulty of aggregating different types of variables to compute the Wasserstein distance without accidentally dimming the contribution of some to the total, we consider here only domains for which all variables have the same or closely-related types (*e.g.* categorical with a close number of modalities). We compute our metric using the squared errors normalized to the variable domain for continuous variables, and the error (0/1 loss, multiclass single valued) of prediction for nominal variables (in all cases, the contribution to the distance of each variable is in $[0, 1]$).

Results A key part of our experiments was to compare approaches on simulated data since we then know the ground truth, including data with non-trivial structure. Table A6 provides all results on our simulated domains. Several conclusions can be drawn: first, our GTs appear to be winning on at least half of the total domain \times MCAR% combinations, regardless of the `mice` contender, even when heavy disparities appear depending on the domain. On `gridGauss` for example, we become competitive for large % MCAR (though not statistically significantly) while on `circGauss` we statistically significantly beat all `mice` contenders on *all but one run*. We can also notice the quality of the imputations from the plots: NORM in `mice` is clearly failing to impute mostly on the heavy dense regions of the domain. Our results are visually much closer to those of CART and RF. We suspect that our method has a different imputation ‘quality’ pattern vs CART and RF: such methods typically successfully impute near the modes while we can get a more balanced allocation of data *among* modes. On domains like `circGauss`, we suspect this is the source of our better results. We then have tested what happens when the number of variables increases: to assess this, we used different kind of data (boolean / trinary valued) and domains with an increasing number of variables (from 8 to 181). Results are in Table A7, from which it comes that we are competitive on problems on up to 25 variables and while we are significantly beaten by `mice` on the largest problem, one has to keep in mind that on `dna`, we compete on imputation with a single tree model per fold when random forests aggregate **90 500** of them to do the same task. One may expect that this has impact as well on the time to complete the task. This turns out to be true: on `dna`, it takes less than 5 minutes to impute a fold with GTs (taking into account the training of the GT), while `mice` requires more than *two hours* for the same task on random forests. The implementations of our algorithms (Java) and `mice`’s (R) require caution in comparing times, but we can safely say that on such domains with relatively large number of

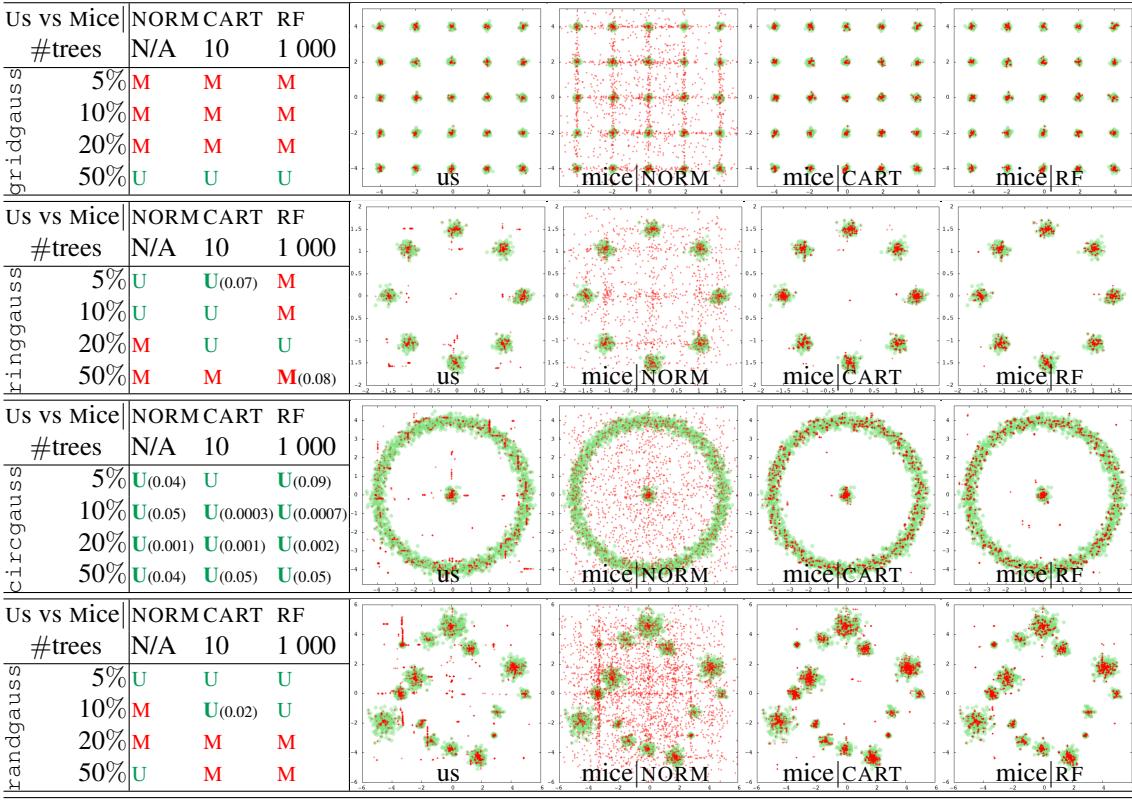


Table A6. Experiments IMPUTE on simulated domains gridgauss, ringgauss, circgauss and randgauss: comparison of copycat GT induction with maximal size (us) vs mice (van Buuren & Groothuis-Oudshoorn, 2011) with three prediction methods: NORM, CART and Random Forests. *Left table*: for each % of missing completely at random (MCAR) variables (in {5, 10, 20, 50}), we indicate which of us (**U**) or mice (**M**) achieves the lowest average Wasserstein metric W_2^2 . **Bold faces** indicate when a Student paired *t*-test returns a *p*-val < 0.1, in which case the *p*-values are indicated in parenthesis. *Right plots*: examples imputation results (red dots) on top of the domain's data (green) for 50% MCAR.

variables, our approach takes much less time to complete the task. Finally, mice is specifically designed for imputation while our GTs can be used for other purposes than just imputation itself.

	Us vs Mice		CART RF		Us vs Mice		CART RF		Us vs Mice		CART RF		Us vs Mice		CART RF	
	#trees	40	4 000	#trees	45	4 500	#trees	125	12 500	#trees	905	90 500	#trees	905	90 500	
led	5%	U(0.04)	U	5%	U	M(0.07)	5%	M	U	5%	M(0.03)	M(0.02)	led24	dna	M(0.02)	
	10%	U(0.04)	M	10%	U	M	10%	U	M	10%	M(0.02)	M(0.004)				
	20%	U	U	20%	M	M(0.05)	20%	U	M	20%	M(0.001)	M(0.002)				
	50%	M	U	50%	U	M	50%	M(0.01)	M(0.07)	50%	M(0.002)	M(0.001)				
tictactoe		led24		dna		M(0.001)		M(0.002)		M(0.001)		M(0.002)		M(0.001)		

Table A7. Experiments IMPUTE on binary/trinary valued domains led, tictactoe, led24, dna, with increasing number of nominal description variables. Notations follow Table A6 (NORM not shown as it does not impute all NAs). While we are competitive on domains with the smallest number of variables, we are clearly beaten by mice when the number of variables substantially increases like for dna. Those numbers have to be read keeping in mind the number of trees involved in imputing a single fold: while we compete with 1 tree against 40 (CART) and 4 000 (RF) on led, we compete with 1 tree against **905** (CART) and **90 500** (RF) on dna.

Domain	#1	#2	#3	#4	#5	#6	#7	#8
abalone	COPY	U(MAX)	U(300)	C(1K)	C(300)	C(10)	U(10)	UNIF
dna	COPY	U(300)	U(MAX)	UNIF	C(10)	U(10)	C(1K)	C(300)
house votes	COPY	U(300)	U(MAX)	C(1K)	U(10)	C(10)	C(300)	UNIF
iris	COPY	C(1K)	U(300)	U(MAX)	C(10)	C(300)	U(10)	UNIF
led24	COPY	U(300)	U(MAX)	C(10)	U(10)	C(1K)	UNIF	C(300)
led	COPY	U(300)	U(MAX)	C(1K)	U(10)	C(10)	C(300)	UNIF
winered	COPY	U(300)	U(MAX)	C(1K)	UNIF	U(10)	C(300)	C(10)
winewhite	COPY	U(MAX)	U(300)	U(10)	UNIF	C(300)	C(10)	C(1K)
sigma-cabs	COPY	C(1K)	C(300)	C(10)	U(MAX)	U(300)	UNIF	U(10)
open-policing	COPY	U(MAX)	C(1K)	U(300)	C(300)	C(10)	U(10)	UNIF

Table A8. Ranking results on experiment TRAIN-GEN, showing for each domain the order (left to right: from best to worst) of the three sizes of runs of our GTs (U(.), number in parenthesis = number of splits; MAX = up to 10 000 splits), the three runs of CTGAN with different epoch numbers (C(.)), the COPY approach (we use the original data) and the UNIF(orm) approach (we use a random sample). Those two last methods have their cells shaded to locate them.

COPY	U(300)	U(MAX)	C(1K)	C(10)	U(10)	C(300)	UNIF
1	2.9	3	4.5	5.7	6	6.2	6.8

Table A9. Average rank for each approach in the TRAIN-GEN, as collected in Table A8, ordered in increasing average rank.

II.6. 'Training on generated' experiment (TRAIN-GEN)

Objective In the context of tabular data, there can be several reasons to replace a training sample with a data generator: (i) the storing size of the generated model, in particular for big databases and / or with substantial redundancy (Rob & Coronel, 1995), (ii) privacy (disregarding the privacy model), (iii) security (with respect to classical databases), etc. . The objective of this experiment is the replacement of the training data by a data generator *to address the supervised learning problem* related to the training data. For example, for domain *iris*, the objective is the prediction of a flower's variety among three.

Experimental setting For each domain, we carry out a 5-fold CV, leaving 20% of the data for testing and the rest for training. We train a data generator with the training data, then generate a training sample the size of the generator's training

		C(.)	10*	300*	1K**
		U(.)			
10		1 / 7 / 2	3 / 4 / 3	2 / 3 / 5	
300		8 / 1 / 1	8 / 1 / 1	4 / 5 / 1	
MAX		8 / 1 / 1	8 / 1 / 1	7 / 2 / 1	

Table A10. Experiment TRAIN-GEN: statistical wins / ties / statistical losses for us (U()) vs CT-GAN (C()). Statistical = significant for $p \leq 0.01$. For example, $a / b / c$ means we statistically win a times, lose c times and there is no statistical difference b times. Each red star (*) indicates a domain for which the related technique performed statistically *worse* than uniform sampling (UNIF) for $p = 0.05$ (See Table A8 to spot those domains).

Domain	$\text{U}(10)$	$\text{U}(300)$	$\text{U}(\text{MAX})$	$\text{C}(10)$	$\text{C}(300)$	$\text{C}(1K)$
abalone	23	90	193	14	62	215
dna	2	7	38	143	1 051	3 128
house-votes	ε	ε	ε	7	20	51
iris	ε	ε	ε	7	17	42
led24	ε	1	2	6	22	62
led	ε	1	1	6	14	46
winered	3	8	14	13	24	62
winewhite	17	46	131	17	52	210
sigma-cabs	19	60	286	13	86	419
open-policing	168	311	933	24	338	1 316

Table A11. Average training times to get the generated training sample on experiment TRAIN-GEN, in seconds, rounded to the nearest second. ' ε ' means average < 0.5s.

Domain	$ \text{GT} / \mathcal{X} \cdot 100 (\%)$
abalone	3.99
dna	0.26
house-votes	21.57
iris	200.13
led24	6.00
led	18.76
winered	7.82
winewhite	2.55
sigma-cabs	2.31
open-policing	0.41

Table A12. Experiment TRAIN-GEN: average sizes of the GT obtained using $\text{U}(300)$ relative to the domain size (see text).

Generative Trees: Adversarial and Copycat

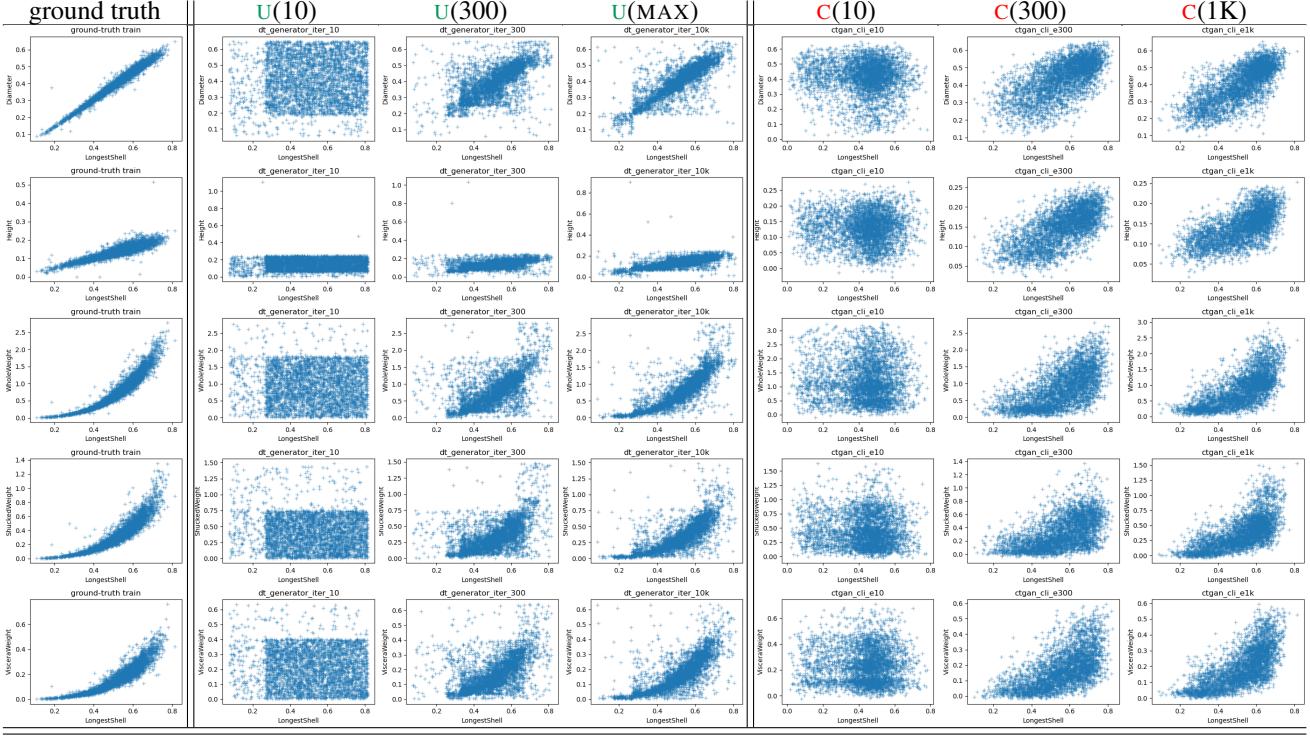


Figure 16. Experiment TRAIN-GEN: 2D distribution plots for domain abalone (warning: scales vary).

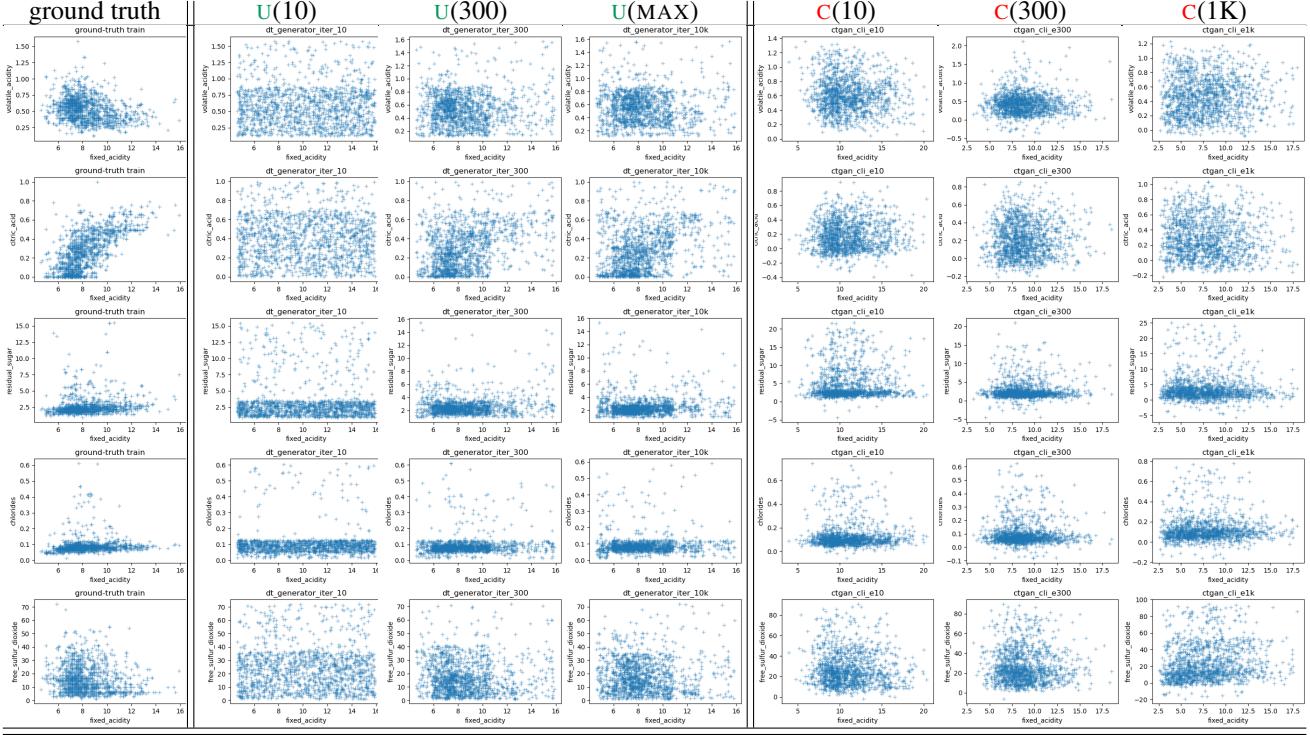


Figure 17. Experiment TRAIN-GEN: 2D distribution plots for domain winered (warning: scales vary). Notice that GTs manage to capture complex distribution shapes (such as the 'wavy' 2nd row) that neural nets do not necessarily capture, and neural networks can generate data clearly outside admissible bounds (negative values for some variables)

sample. This training sample is then used *in lieu of* the original training sample to train a model following the original data's task. The model is then used to predict the class of the testing fold's examples. To alleviate the bias on the choice of this classifier and its training algorithm, we pick two types of classifiers / training models: random forests (RFs) and gradient boosted decision trees (GBDTs), that we thus run on each dataset. We compute the accuracy for label prediction and root mean square error for regression problems. For each domain and each generator type, we thus obtain 10 statistics (5 for RFs, 5 for GBDTs), to which we add the 10 running times and use them for comparisons between methods. Speaking methods, we consider for generative trees (GTs) copycat training against a discriminator minimizing Matusita's loss (Kearns & Mansour, 1996). We grow generative trees with three different sizes: very small (10 splits, *i.e.* 21 total nodes), medium (300 splits, *i.e.* 601 total nodes) and 'maximal'. In this last case, we provide a limit size of 10 000 splits (*i.e.* 20 001 total nodes). Notice that this maximal size is usually not reached, in particular when a training fold contains less than 10 000 training examples. We compare our three GT training flavours to three training neural network based methods relying on the state of the art CTGAN (Xu et al., 2019). We use CTGAN code⁶ with default parameters and varying number of epochs, choosing small (10), medium (300) and large (1K) training epochs. We also use two more contenders: the first is the COPY contender, which uses the original training data as training data. The second is the UNIFORM contender, which just generates uniform data in the features domains. While we expect COPY to lead the pack of algorithms, contender UNIF is used to point algorithms failing at learning anything 'substantial' about the domain when it significantly beats them.

Results Table A8 provides the *ranking* results among all eight contenders for each domain, where the average of the metric (accuracy or RMSE) is used to rank from best to worst. Table A9 provides a more synthetic view by computing the average rank for each contender. There are several conclusions to draw: (i) as perhaps expected, COPY is always the best approach; what is perhaps less expected is that (ii) UNIF is far from always being the worst, CTGAN being the most frequently beaten contenders (albeit not necessarily statistically significantly, see below) on four out of ten domains. More importantly, (iii) our approach largely performs the best in all non-COPY approaches *unless* small GTs are used (U(10)). In eight out of ten domains, U(300) is in the top-three contenders, and top-two if we exclude COPY. An interesting observation is that U(MAX) is *also* in the top-three contenders in eight out of ten domains, showing that there is reduced overfitting effect due to the large tree size (which we attribute in part to the constraint that $p \notin \{0, 1\}$ for GT splits). Obviously, in a context where explainability would be key, the smaller size option (U(300)) would be a preferred choice. To dig further in comparing our method to CTGAN, we have computed the number of domains one *significantly* ($p = 0.01$) beats the other, adding to those statistics the number of times UNIF does *significantly* ($p = 0.05$) beat some contender(s). All results are summarized in Table A10. The results display the superiority of our GT-based approach (unless, again small trees are used), but they also display that CTGAN are, in few cases, significantly beaten by UNIF – and this can happen for all three epoch numbers. This aligns with the observation that dealing with tabular data with neural nets forces sophisticated choices for the design (Xu et al., 2019) and probably has as consequence that not optimizing sufficiently hyperparameters can result in worse performances than uniform generation. We clearly do not have this problem and believe that this is due to the fact that the tree (graphs) used in GTs bring the same convenience for data generation as the tree (graphs) used in decision trees have for discrimination. Figures 16 and 17 provide two examples of sets of 2D plots showing the distribution of generated examples according to different sets of couples of real-values variables. The absence of overfitting as well as the capturing of sophisticated features of the data's distribution is quite apparent, also in comparison with neural nets. Last, we have computed the average computation time to get the generated datasets for us and CTGAN – thus, inclusive of the training time for the generator. Results are provided in Table A11. One must be cautious in comparing numbers as our implementation of our algorithms is in Java while CTGAN's is in Python, but at least one conclusion seems fair to draw: we are in general – and especially for bigger models / longer training – achieving much better results than CTGAN. The dna domain, for which the imputation experiments were already displaying our superiority in terms of training time (Section 6), is a clear example of reduction in training time that can be of order $10\times$ — $100\times$ with GTs compared to neural networks.

To put our results in perspective, we have also computed the relative size of the GT learned with respect to each domain size, for each experiment: we use as GT size $|GT|$ the total number of vertices and arcs, which equals $(1 + 5 \cdot \text{split number})$ and as domain size, $|\mathcal{X}|$, the total size of the dataset used (number of examples times number of variables). Table A12 presents the results obtained for U(300), from which it appears that the GT learned can represent a tiny proportion of a domain's size — at most a few percents in most cases.

⁶<https://github.com/sdv-dev/CTGAN>

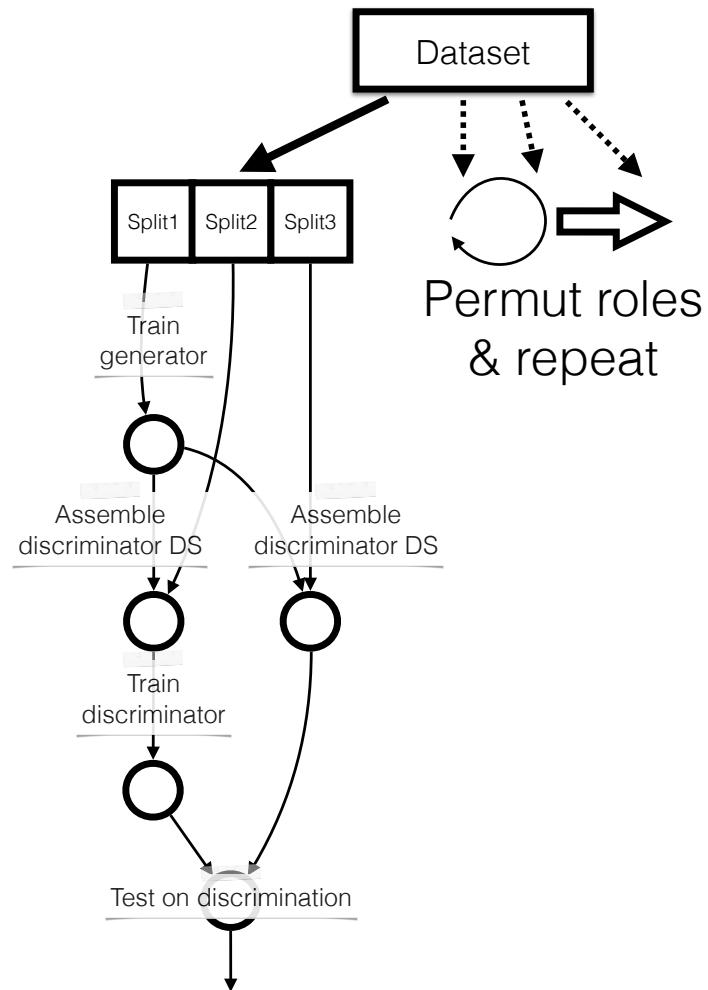


Figure 18. Experiment GEN-DISCRIM: General overview of the pipeline, designed to avoid rewarding generators that would just copy their training sample.

II.7. 'Fake-real discrimination' experiment (GEN-DISCRIM)

Domain	wins vs COPY	loses vs UNIF
circgauss	U(300) (0.01), U(MAX) (0.02)	None
randgauss	U(300) (0.00004), U(MAX) (0.0002)	C(10) (0.01)
ringgauss	U(300) (0.002), U(MAX) (0.001)	None
abalone	None	C(10) (0.0006)
dna	None	None
house-votes	None	C(10) (0.0002)
iris	None	C(10) (0.007), C(300) (0.0002)
led24	None	None
led	None	C(1K) (0.03)
winered	None	U(300) (0.003), U(MAX) (0.001)
winewhite	None	U(300) (0.006), U(MAX) (0.001)
sigma cabs	None	U(10) (0.001), U(300) (0.001), U(MAX) (0.001), C(10) (0.001), C(300) (0.003)
open-policing	None	None

Table A13. Experiment GEN-DISCRIM: for each domain, we compute the list of contenders in our method and CT-GAN statistically winning against COPY and statistically losing against UNIF (for $p \leq 0.05$, indicated in parenthesis).

		10	300	1K
		10	300	1K
	C(.)	7 / 4 / 2	6 / 5 / 2	6 / 5 / 2
	U(.)	8 / 4 / 1	8 / 3 / 2	8 / 3 / 2
		8 / 4 / 1	8 / 3 / 2	8 / 3 / 2
10		7 / 4 / 2	6 / 5 / 2	6 / 5 / 2
300		8 / 4 / 1	8 / 3 / 2	8 / 3 / 2
MAX		8 / 4 / 1	8 / 3 / 2	8 / 3 / 2

Table A14. Experiment GEN-DISCRIM: statistical wins / ties / statistical losses for us (U()) vs CT-GAN (C()). Statistical = significant for $p \leq 0.01$. For example, $a / b / c$ means we statistically win a times, lose c times and there is no statistical difference b times. (See Table A13 to spot those domains).

$\text{C}(.)$	10	300	1K
$\text{U}(.)$	6 / 1 / 3	6 / 2 / 2	4 / 1 / 5
10	9 / 0 / 1	9 / 0 / 1	6 / 2 / 2
300	9 / 0 / 1	9 / 0 / 1	7 / 1 / 2
MAX	9 / 0 / 1	9 / 0 / 1	7 / 1 / 2

Table A15. Experiment GEN-AUG: statistical wins / ties / statistical losses for us ($\text{U}(.)$) vs CT-GAN ($\text{C}(.)$). Statistical = significant for $p \leq 0.01$. For example, $a / b / c$ means we statistically win a times, lose c times and there is no statistical difference b times.

Objective The objective fits in a simple question: *can generated examples look like real ones ?*

Experimental setting The question is simple but its treatment non trivial: we need in particular to avoid ‘rewarding’ generators that would just copy their training examples. Figure 18 provides the training pipeline we have designed, that we ran for each domain considered. In short, we split each domain in three equal sized parts, say \mathcal{S}_1 , \mathcal{S}_2 and \mathcal{S}_3 . One of these parts, say \mathcal{S}_1 , is used to train the generator, which then generates a sample $\tilde{\mathcal{S}}_1$ having the same size as \mathcal{S}_1 . We then train a discriminator for the 2-classes supervised learning problem consisting in distinguishing real from fake, using $\tilde{\mathcal{S}}_1$ and another original part, say \mathcal{S}_2 as training samples. The discriminator is then *tested* on the problem consisting in distinguishing $\tilde{\mathcal{S}}_1$ from the last original part (not yet used), \mathcal{S}_3 in this case. The smaller the final accuracy, the more ‘realistic’ is considered $\tilde{\mathcal{S}}_1$. We then permute the roles of the three samples and run the experiment again, ending up in $3! = 6$ accuracies for each domain. Considering that the discriminator is trained and tested on two different subsets of the original data as real data, there is an incentive to not just ‘copy’ the original data but capture features about the domain that generalise well for data generation. We have considered the same generators as in experiment ‘TRAIN-GEN’: CTGAN with small (10), medium (300) and large (1K) number of training epochs; our method with 10, 300 and MAX splits for GTs (recall that MAX = training up to 10 000 splits); we also consider the COPY and UNIFORM baselines, the former giving an idea of the accuracy for the original data and the latter giving the most ‘blunt’ baseline. We consider the same discriminators as in experiment ‘TRAIN-GEN’ (random forests and gradient boosted decision trees).

Results We first have a look at the *extreme* results, *i.e.* how our method and CTGAN compare to COPY and UNIF. Table A13 presents the detailed results obtained for each domain. In this table, we only look at statistically significant results — for example, when the accuracies on testing were statistically significantly *larger* than UNIF (which means that UNIF performed better), or when the accuracies on testing were statistically significantly *smaller* than COPY (which means that COPY performed worse). The picture with respect to UNIF displays that some CTGAN (disregarding the number of epochs) get worse results on almost half of the domains (6) while some of our methods gets worse results on 3 of them. When looking at COPY, we see that on our simulated domains, our method actually gets *systematically* significantly better results than COPY when the number of splits is at least 300. This, we believe, signals the potential for GTs to be used as efficient data generators, eventually as parts of more complex generators for more complex domains than our generated domains. We have also drilled down in the comparison between our approach and CTGAN in the same way as we did for experiment TRAIN-GEN. Table A14 presents the aggregated results, whose formatting follows the same rules as for Table A10. The conclusion from this Table is that our approach does better at producing ‘realistically looking’ datasets than CTGAN does. When the GTs are big enough (at least 300 splits), the picture displays that our approach wins against all CTGAN alternatives on a large majority of domains.

II.8. ‘Generated augmentation’ experiment (GEN-AUG)

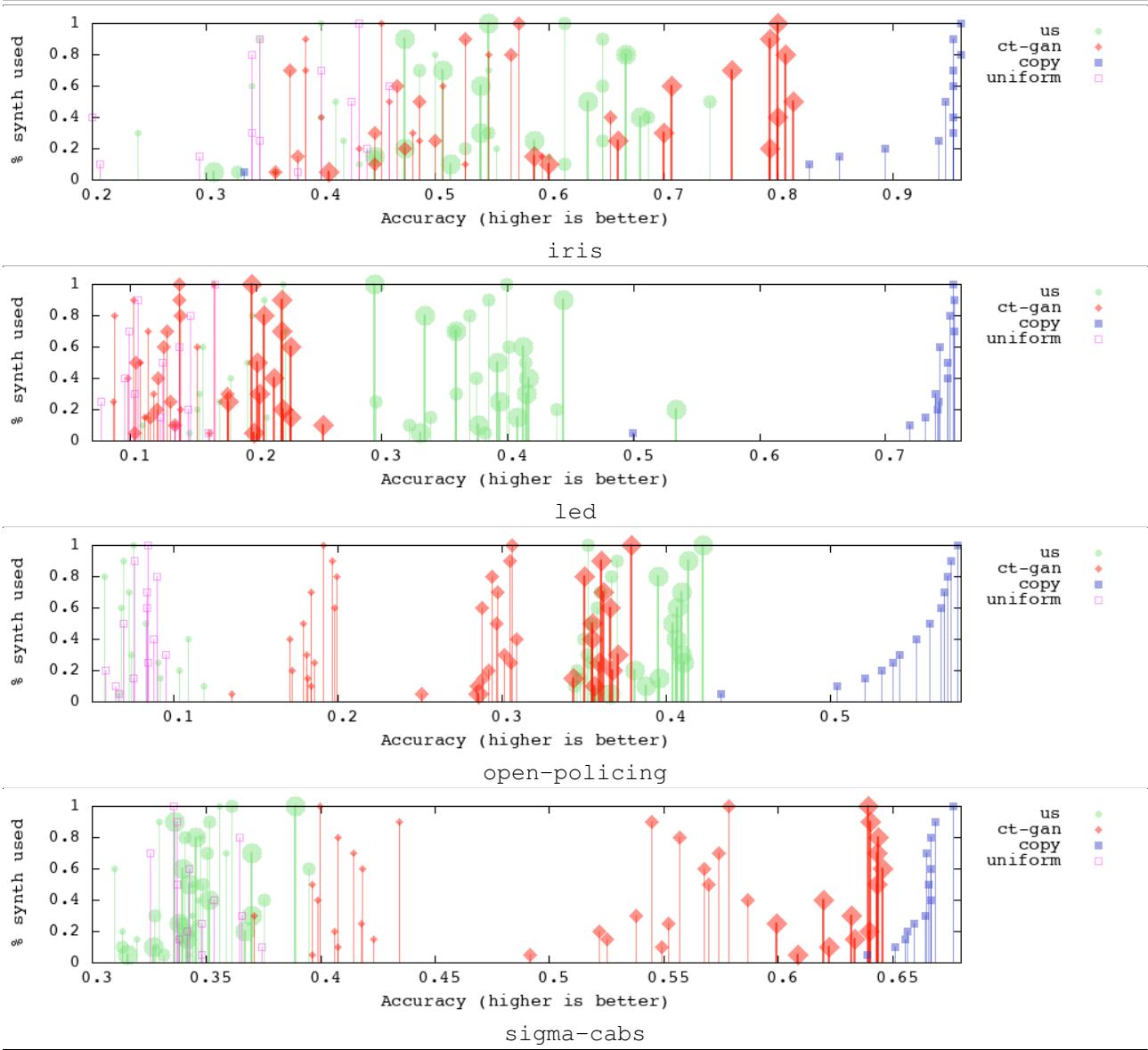


Table A16. Experiment GEN-AUG: detailed results on four additional domains for which the metric is the accuracy. Conventions follow Table 4 in the main file.

Objective Supplementing real data with additional ‘faithful’ generated data could be envisioned as a way to improve the performances of models trained from the whole data, a problem of substantial practical impact. Our objective was to test how generative models can perform in such a scenario.

Experimental setting The setting can be summarized as the equivalent of the TRAIN-GEN experiment (Section II.6), with the sole modification that we train (supervised) models using generated data mixed with the training data of the generators instead of just the generated data alone. Equivalently, we train with COPY + a generated sample. This generated sample could be generated by CTGAN or our GTs, but we also try the COPY case (we add real examples) and the UNIFORM case (we add uniformly generated examples). We first tried the simplest experimental setting in which the size of the generated data was the same as the size of the training data, but the experiments were largely inconclusive in terms of who wins or loses. Hence, we have dug in this scenario, allowing a varying % of generated data to be added to the training data, for a % of generated data that would represent 5%, 10%, 15%, ..., 100% of the training (real) data. This represents a lot more experiments but also allows us to understand, for each generative technique, what is the effect of putting more generative examples in the training sample. Regardless of the comparison with the COPY approach, a good generative approach should bring improved results as the number of generated examples increases. All other parameters, generative models and supervised classifiers considered are the same as in the TRAIN-GEN experiment.

Results Table A15 summarises the results obtained, much in the same way as Tables A10 and A14. The results display that GTs tend to be a better fit for data augmentation than CTGAN. Training more the neural nets does not yield an advantage over GTs, as even when comparing with GTs having a few dozen nodes, the picture is still quite balanced. Table A16 completes Table 4 for the remains domains used. We observe that while dna is clearly the domain where CTGAN obtained the worst results, with *all* accuracies (substantially) lower than any of UNIF, such bad performances also occur in domains winewhite and winered, which could signal that the issue is not linked to the domain being categorical (winewhite and winered are both real-valued). On iris, CTGAN have a slight edge over GTs, an edge which much more significant on sigma-cabs where GT results basically cannot be distinguished from UNIF. We observe that in this domain, CTGAN manage to get an improvement over +5% real data, in the same way as GTs manage to get an improvement over +5% real data on abalone. open-policing, which is our biggest domain, displays that training for a longer time is beneficial to both CTGAN and GTs, but ultimately GTs get the best results. Two conclusions can be drawn: first, there is still work to do to beat adding real data, for both neural nets and generative trees, even when those latter models seem to overall get the best and most stable results. Second, we do not observe for generative trees the apparent overfitting pattern that follows CTGAN on dna, winewhite and winered. Apart from the fact that we prevent discarding support in the induction of the GTs, we do not see currently any other reason for this observation to happen.