Google Research

Generative Trees: Adversarial and Copycat

Richard Nock Mathieu Guillame-Bert

Why this work

- Tabular data = prevalent+ for value potential — Chui *et al.* 2018
- Generative SOTA for tabular data = DL based ***unlike*** SOTA for classification (tree based)
- Dissatisfaction, new directions needed for generative approaches — Camino *et al.* 2020

Starting point

- Map key frameworks for tabular data classification onto generative world
  - ↳ **Loss functions**: properness — Savage, 1971
  - ↳ **Models**: tree-based — Breiman *et al.*, 1984
  - ↳ **Algorithms**: boosting — Kearns & Mansour, 1996

What we get (summary)

- GAN-game losses *from discriminator*, *tight* if discriminator *calibrated*, chi square "*universal*" to train generator
- New *tree-based* generative models: XAI+, density in O(depth), likelihood *w/ missing features* in O(size)
- *Boosting* algorithm for adversarial training +geometric conv. of chi square under *weak generative assumption*
- *New copycat training* (unknown for DL), "boosting for free" convergence if boosted discriminator (C4.5, etc.)

Loss functions

No discussion on generalisation

Discriminative (for class probability estimation)

Objective: learn posterior η to estimate $\mathbb{P}[Y = 1|X]$

Ideal: $\eta^\star = \pi \cdot (\mathrm{dP}/\mathrm{dM})$ — Bayes posterior

Binary task

$\mathbb{P}[X|Y = 1]$ $\mathbb{P}[X|Y = -1]$

$\mathbb{P}[Y = 1]$

$\mathrm{B} \doteq (\pi, \mathrm{P}, \mathrm{N})$

prior "real" "fake"

$\mathrm{M} \doteq \pi \cdot \mathrm{P} + (1-\pi) \cdot \mathrm{N}$

Generative (for measure comparison)

Objective: learn sampler for N that looks like P

Ideal: P

Loss: $\ell : \{-1, 1\} \times [0, 1] \to \overline{\mathbb{R}}$ (true class, posterior estimate)

$\underline{L}(p) \doteq \inf_u \mathbb{E}_{Y \sim \mathrm{B}(p)}[\ell(Y, u)]$ Bayes risk

$p = \arg\inf \Rightarrow \ell$ ***strictly proper***

Information of Binary Task B

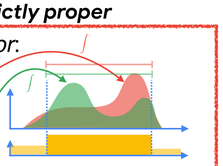
$$\mathbb{I}_f(\mathrm{P}, \mathrm{N}) \doteq \int f\left(\frac{\mathrm{dP}}{\mathrm{dN}}\right) \mathrm{dN}$$

(cvx)

The better N, the **smaller** $\mathbb{I}_f(\mathrm{P}, \mathrm{N})$

Calibrated posterior:

$$\tilde{\eta} = \pi \cdot \frac{\mathrm{dP}_{\tilde{\eta}}}{\mathrm{dM}_{\tilde{\eta}}}$$



agreement on the level sets of the posterior

↳ Best calibrated posterior: Bayes' $\eta^\star$, worst: prior π

↳ Important: **any** decision tree with local posterior predictions at the leaves *is* calibrated

Statistical information of calibrated $\tilde{\eta}$

$$\Delta\underline{\mathbb{L}}(\tilde{\eta}, \mathrm{M}_{\tilde{\eta}}) = \underline{L}(\pi) - \underline{\mathbb{E}_{X \sim \mathrm{M}_{\tilde{\eta}}}[\underline{L}(\tilde{\eta}(X))]}$$

CART, C4.5, etc. (splitting criterion)

The better $\tilde{\eta}$, the **higher** $\Delta\underline{\mathbb{L}}(\tilde{\eta}, \mathrm{M}_{\tilde{\eta}})$

Theorem: for any *calibrated* $\tilde{\eta}$ and any *strictly proper symmetric differentiable* loss ℓ,

Generator's loss (see paper)

$$\Delta\underline{\mathbb{L}}(\tilde{\eta}, \mathrm{M}_{\tilde{\eta}}) = \mathbf{H} + \mathbf{G} = \mathbb{I}_{f^\pi}(\mathrm{P}_{\tilde{\eta}}, \mathrm{N}_{\tilde{\eta}})$$

Tight, unlike *f*-GAN formulation:

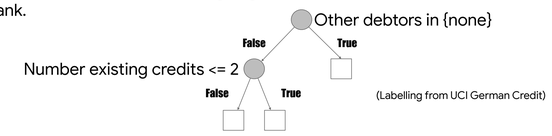
$$\mathbb{I}_f(\mathrm{P}, \mathrm{N}) \geq \sup_{\tilde{h}} \left\{ \underbrace{\mathbb{E}_{\mathrm{P}}[\tilde{h}(X)]}_{\mathbf{H}} - \underbrace{\mathbb{E}_{\mathrm{N}}[f^\star \circ \tilde{h}(X)]}_{\mathbf{G}} \right\}$$

+Lemma: if additional property on ℓ, to minimise **G**, sufficient to minimise the *chi square* $\chi^2(\mathrm{N}_{\tilde{\eta}}||\mathrm{P}_{\tilde{\eta}})$ holds for many popular losses (log-, square-, etc.)

↳ gives "one loss to train N against any calibrated discriminator" — valid outside decision trees

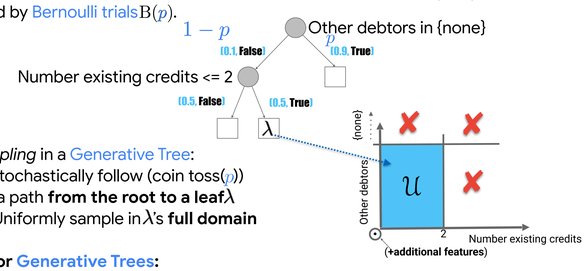
Models

Tree: a tree is a binary directed tree whose internal nodes are labeled with a test on an observation variable and outgoing arcs are labeled with truth values. Leaves are blank.



Decision Tree: A *decision tree h* is a **tree** in which leaves are labeled by values in [0,1]

Generative Tree: A *generative tree* (GT) *G* is a **tree** in which outgoing arcs are labeled by Bernoulli trials $\mathrm{B}(p)$.



↳*Sampling* in a Generative Tree:
(1) Stochastically follow (coin toss(p)) a path **from the root to a leaf** λ
(2) Uniformly sample in λ's **full domain**

Pros for Generative Trees:
↳Trainable from data w/ *missing values* (see algorithms)
↳XAI+: as "easy" to *interpret* as a decision tree (see experiments)
↳*Density* computable in O(depth) at any point,
↳*Likelihood* of missing values given partial observation in O(size), P[A|B] equivalently easy
↳Many metrics involving a GT computable *exactly* (no sampling required !): chi square, etc.

Cons for Generative Trees:
↳*Axis // partition* of the support (restrictive)
↳Support / domain *closed* (workarounds exist, *e.g.* Box-Muller transform for Gaussians)

Algorithms

The prior is under control, so we pick π=0.5

Adversarial training

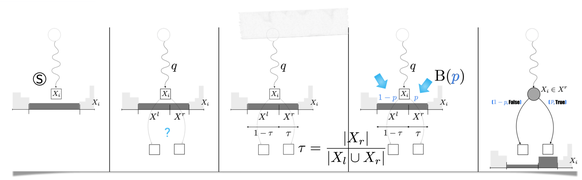
Algorithm TD-GEN(G, h)

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

$$p \leftarrow \textsc{Clamp}\left(\frac{\mu_{\mathrm{LL}} - \mu_{\mathrm{LR}}}{\mu_{\mathrm{LL}} + \mu_{\mathrm{RR}} - 2\mu_{\mathrm{LR}}}\right);$$

(see paper)

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



Top-down: pick a leaf, variable, split, compute Bernoulli's p, *repeat*

Convergence if p in (0,1)
↳If $\chi^2(.||\mathrm{P}_{\tilde{\eta}}) \geq \delta$ and $|\tau - p| \geq \varepsilon$

$$\chi^2(\mathrm{N}'_{\tilde{\eta}}||\mathrm{P}_{\tilde{\eta}}) \leq \underbrace{\frac{1}{1+\delta\varepsilon^2}}_{\text{new}} \cdot \underbrace{\chi^2(\mathrm{N}_{\tilde{\eta}}||\mathrm{P}_{\tilde{\eta}})}_{\text{old}}$$

Convergence if p in {0,1} — *discard support*
↳If weak generative assumption($\delta > 0$)
(δ=0⇒complete independence between generation and discrimination)

$$\chi^2(\mathrm{N}'_{\tilde{\eta}}||\mathrm{P}_{\tilde{\eta}}) \leq \frac{1}{1+\delta q^2(\tau + (1-2\tau)p)^2} \cdot \chi^2(\mathrm{N}_{\tilde{\eta}}||\mathrm{P}_{\tilde{\eta}})$$

Copycat training new

Generator and Discriminator contain a **tree**
In copycat training, the Discriminator h is learnt by a boosting algorithm (C4.5, CART, ...)
↳After each iteration for h, the Generator G copies the update of the **tree** of h *in its* **tree** and computes its Bernoulli probabilities to keep $\chi^2(\mathrm{N}_{\tilde{\eta}}||\mathrm{P}_{\tilde{\eta}}) = 0$
↳After the update on G, G updates the training sample of h by resampling what has been affected by its update ⇒ the error of h on the new sample is 50%
↳Repeat with a new update to the **tree** of h

G implements the WHA's distribution !

Pros for Copycat training
↳Straightforward to implement to build the Generator
↳Almost zero algorithmic overhead over the Discriminator's boosting algorithm
Cons for Copycat training
↳No privacy for the Discriminator

Boosting-compliant convergence "*for free*"

↳boosting rates in density ratio loss for the Generator follow directly from boosting rates for the Discriminator
↳Let ℓ= *Matusita*'s loss ($\underline{L}(u) = 2\sqrt{u(1-u)}$), which means we run Kearns & Mansour's top-down DT induction algorithm (Kearns & Mansour, STOC'96)
↳Suppose boosting's Weak Hypothesis Assumption WHA($\delta > 0$) holds to train h
↳For any $\varepsilon > 0$, if #iters T for h satisfies

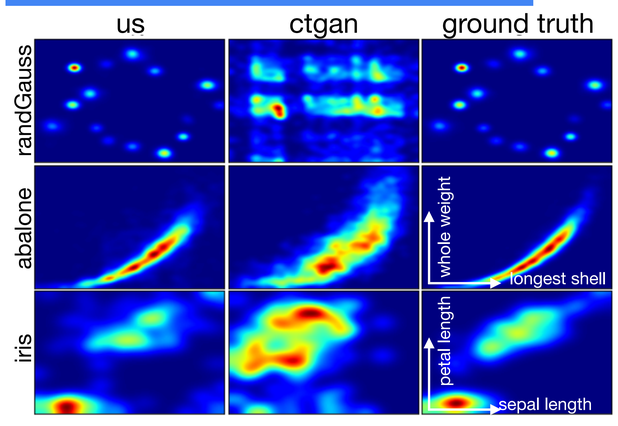
$$T \geq \left(\frac{1}{f(\varepsilon)}\right)^{\frac{32}{\delta^2}} \in (0, 1)$$

then the Generator G satisfies

$$\mathbb{B}_\ell(\mathrm{P}, \mathrm{G}) \leq \varepsilon \cdot \mathbb{I}_{f^\pi}(\mathrm{P}, \mathrm{U})$$

density ratio loss

Experiments



us = copycat training a GT G for max 10K splits
SOTA: CTGAN (Xu *et al.*, NeurIPS'19), trained for 1K epochs

Missing data imputation

Setup:
↳Remove q% values in dataset (*Missing Completely At Random*), 5-fold CV
↳Impute missing values w/ algorithm, compare with domain using OT (W_2^2) metric

Some results (more in paper)

U = we win; M = mice wins (**bold** = p-val < .1, indicated)

circGauss	Us vs Mice | #trees p. fold	NORM N/A	CART 10	RF 1 000
	(q=)5%	U(0.003)	U	U(0.07)
	10%	U(0.03)	U(0.002)	U(0.007)
	20%	U(0.0005)	U(0.001)	U(0.001)
	50%	U(0.04)	U	U

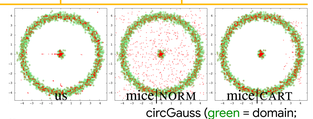
letter	Us vs Mice | #trees	CART 40	CART 4 000	RF 4 000
	5%	U(0.04)	U	
	10%	U(0.04)	M	
	20%	U	U	
	50%	M	U	

tictactoe	Us vs Mice | #trees	CART 45	RF 4 500
	5%	U	M(0.07)
	10%	U	M
	20%	M	M(0.05)
	50%	U	M

dna	Us vs Mice | #trees	CART 905	RF 90 500
	5%	M(0.03)	M(0.02)
	10%	M(0.02)	M(0.004)
	20%	M(0.01)	M(0.002)
	50%	M(0.002)	M(0.001)

domain dim	2	8	9	181
#trees p.fold (us)	1	1	1	1
#trees p.fold (mice | RF)	1 000	4 000	4 500	90 500

Observations:
↳GTs can beat mice if small dim (unexpected) & always with 1 tree (vs many for mice)
↳training time: GTs << mice (especially w/ RFs)
↳+GTs provide generator (mice: no byproduct)



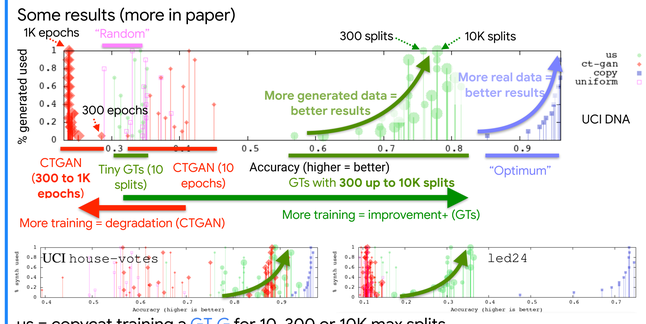
circGauss (green = domain; red = imputed data, q=20%)

us = copycat training a GT G for max 10K splits
SOTA: mice (van Buuren, 2018)
↳After initial guess, round-robin updates one column's missing values given the others using regression/classification method, iterates 5 times
↳We used method in {NORM, CART, Random Forests = RF with 100 trees each}

Generated data augmentation

Setup:
↳Take a **supervised** domain (e.g. UCI iris), 5-fold CV
↳Train generator, generate additional q in {5, 10, ...100}% (of training) examples, train supervised model (GBDT or RFs) from all data, get accuracy/RMSE on test
↳2 baselines: random = +uniform data ("worst"), copy = +real data ("optimum")

Some results (more in paper)

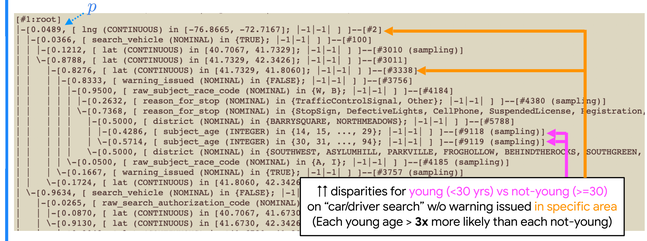


us = copycat training a GT G for 10, 300 or 10K max splits
SOTA: CTGAN (Xu *et al.*, NeurIPS'19), trained for 10, 300 or 1K epochs

... and more

Additional experiments in paper:
↳Training from just generated data (vs CTGAN)
↳Distinguishing real from fake data (vs CTGAN)

XAI by the example: crop of a GT learned on Stanford Open Policing (Hartford)



↑↑ disparities for young (<30 yrs) vs not-young (>=30) on "car/driver search" w/o warning issued in specific area (Each young age > **3x** more likely than each not-young)

Conclusion and future work:
↳Contribution on losses applies beyond our framework to any calibrated disc.
↳Interesting avenues on alleviating the Cons for Generative Trees
↳Experiments display edge vs CTGAN + potential use for missing data imputation