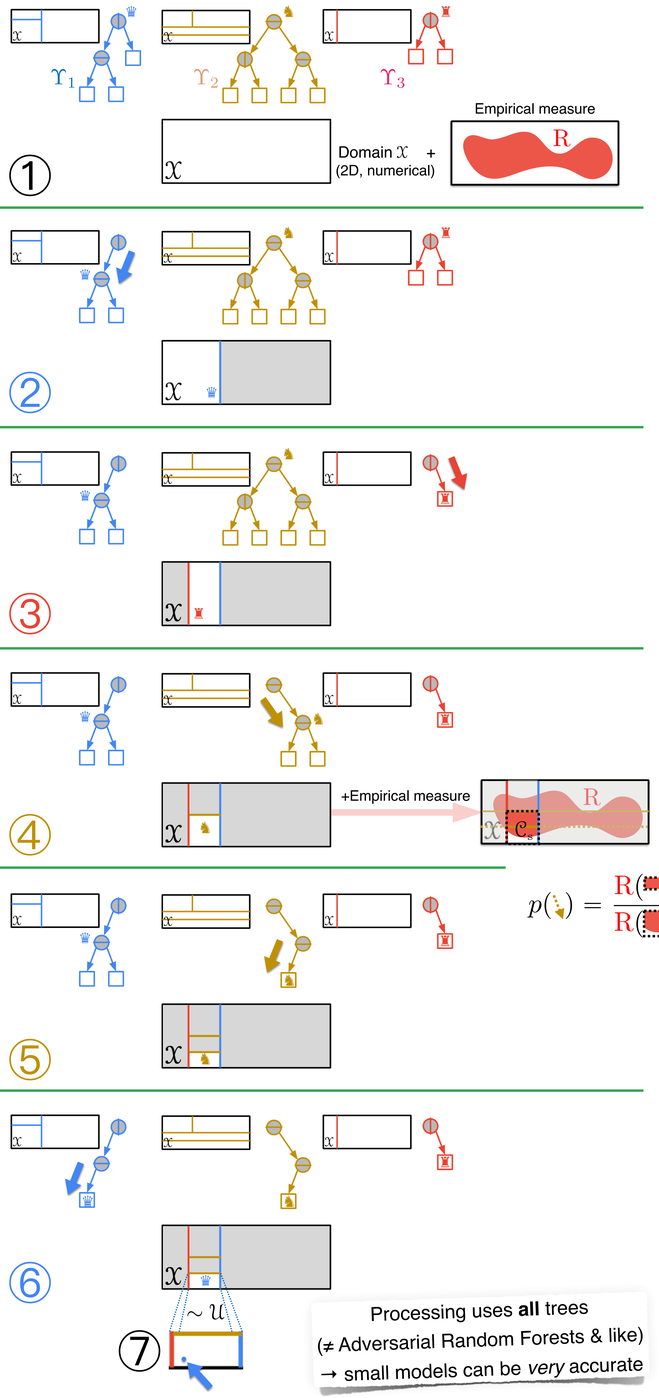
Generative Forests

Richard Nock Mathieu Guillame-Bert

Google Research

Summary

- ***Interpretable*** *forest based generative models*; also enable *efficient* missing data imputation, density estimation
- **Training** *boosting compliant;* reduction trick to *binary supervised training decision trees; natively any kind* of tabular data, *handles missing values*
- **Implementation**: can easily leverage code for top-down DT induction (e.g. CART, C4.5). **Compute**: *cheap*, runs on low-end CPU (**purposefully**)



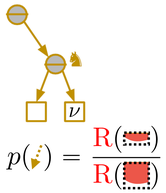
Models, generation, etc.

- Generative Forest G = set of T trees $\{\Upsilon_t\}_{t=1}^T$ + empirical measure R ①
- Generation (1 observation):
  - initialize "tags" = star nodes, ♜ ♛ ♞..., to each root
  - initialize the sampling subdomain $\mathcal{C}_s = \mathcal{X}$
  - until all star nodes are leaves in their respective tree, repeat:

    ②③④⑤⑥ 1. Pick a tree whose star node ♞ ≠ leaf
    2. Compute transition probabilities to a child node ν using empirical measure in $\mathcal{C}_s$

$$p(\nu) = \frac{R(\mathcal{C}_s \cap \mathrm{supp}(\nu))}{R(\mathcal{C}_s)}$$

    3. For selected ν do:
       - ♞ ← ν
       - $\mathcal{C}_s \leftarrow \mathcal{C}_s \cap \mathrm{supp}(\nu)$



$$p(\cdot) = \frac{R(\cdot)}{R(\cdot)} \qquad p(\cdot) = \frac{R(\cdot)}{R(\cdot)}$$

  - when all star nodes are leaves, sample uniformly an observation in the sampling support obtained $\mathcal{C}_s$ ⑦
- Among trees, star node updates can be sequential, concurrent, randomized, etc.

- **+Density estimation**: same procedure but (i) probabilistic branching → deterministic and (ii) stop before $R(\mathcal{C}_s) = 0$. Dens.$\propto R(\mathcal{C}_s)/\mathrm{Vol}(\mathcal{C}_s)$
- **+Missing data imputation**: same as density estimation but get **all** potential $\mathcal{C}_s$ to which partial observation can belong (deterministic branching → *multiple* branching), sample all unknowns in $\mathcal{C}_s$ with *highest* $R(\mathcal{C}_s)/\mathrm{Vol}(\mathcal{C}_s)$

Training

- 2-classes supervised training of Decision Tree *h*: distinguish between "positives" P vs "negatives" Q
- Mixture with prior $\pi = p[Y = 1]$ for positives $M = \pi \cdot P + (1-\pi) \cdot Q$
- Minimize through top-down splitting Bayes risk

$$\mathbb{L}(h) \doteq \sum_{\text{leaf } \lambda} p_M[\lambda] \cdot \underline{L}\left(\frac{\pi p_P[\lambda]}{p_M[\lambda]}\right)$$

$\underline{L}(u) = u \cdot \ell_1(u) + (1-u) \cdot \ell_{-1}(u)$ partial losses for class 1, -1 (log-loss: $\ell_1(u) = -\log(u) = \ell_{-1}(1-u)$)

- Training a generative forest = same proc. with the following adjustments:
  - support of leaves → support intersection of tuples of leaves (1 per tree)
  - positives → observed data (R)
  - negatives → uniform distribution U
  - prior π user-chosen (e.g. 1/2)

Boosting !

For *any* strictly proper differentiable loss ℓ, let T = #trees in GF, all initialized to a root ($G_0 = U$). Suppose weak learning assumption (γ, κ). Then after J splits, current GF G_J has ***likelihood ratio risk***:

$$\mathbb{D}_\ell(R, G_J) \leqslant \mathbb{D}_\ell(R, G_0) - \frac{\kappa\gamma^2\kappa^2}{8} \cdot T \log\left(1 + \frac{J}{T}\right)$$

$$\mathbb{D}_\ell(A, B) \doteq \pi \cdot \mathbb{E}_U\left[D_\varphi\left(\frac{dA}{dU} \middle\| \frac{dB}{dU}\right)\right] \qquad \varphi(z) \doteq \left(z + \frac{1-\pi}{\pi}\right) \cdot (-\underline{L})\left(\frac{z}{z + \frac{1-\pi}{\pi}}\right)$$

"density ratio" loss using Bregman divergence with generator φ

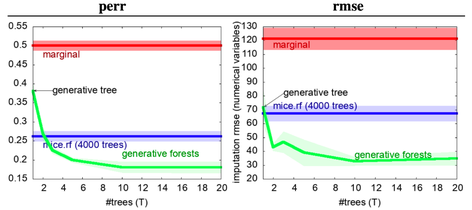
Experiments

Realistic data generation

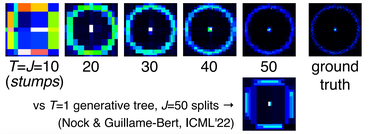
- Metrics:
  - Regularized OT distance (Sinkhorn) (Cuturi, NeurIPS'13) ↘
  - Coverage $\propto \sum_i 1_{\exists \text{ fake} \in B(\text{real}_i,\ r_i)}$ (Naeem et al., ICML'20) ↗
  - Density $\propto \sum_j \sum_i 1_{\text{fake}_j \in B(\text{real}_i,\ r_i)}$ (van Buuren & Groothuis-Oudshoorn, J. of Stat. Soft.'11) ↗
  - F1 measure (using NN classifier) ↘
- Contenders
  - Adversarial Random Forests (stochastic trees) (Watson et al., AISTATS'23)
  - CT-GAN (GAN + NN) (Xu et al., NeurIPS'19)
  - Forest Flow (diffusion trees) (Jolicoeur-Martineau et al., AISTATS'24)
  - Vine Copulas Auto-Encoders (Graphical models + NN) (Tagasovska et al., NeurIPS'19)
- Parameters:
  - ARFs: up to 200 trees (each tree has to be a good generative model → **big** trees)
  - CT-GAN: up to 1000 epochs training
  - FF: need special encoding for categorical variables (!= ours & ARF) → test on 100 % numerical domains; default params: 50 trees, depth ≤ 7 → up to J = 6 750 splits/model
  - VCAE: tested all options (Gaussian, center, direct, regular)
  - us: T=200 trees, J=500 *total* splits (small models) or T=500, J=2 000 (medium sized models)
  - us + ARF + FF run on low-end CPU; CT-GAN + VCAE ran on higher end desktop
- **Results summary** (all tables in paper): we consistently beat NN-based approaches (**CT-GAN, VCAE**); **ARF**: our med-sized models substantially better on all metrics; small-sized better on Sinkhorn, coverage & density; **FF**: our med-sized perform better; small-sized on par for all metrics except density

Missing data imputation

- Missing completely at random (MCAR): randomly remove 5% of data, train from data *with* missing values, predict missing
- Metric: average per-feature error (categorical), rmse (numeric)
- Contender: MICE with CART or **big** random forests (round robin prediction of missing values w/ classification / regression methods) (van Buuren & Groothuis-Oudshoorn, J. of Stat. Soft.'11)
- Example result on OpenML analcatdata_supreme (more in paper):



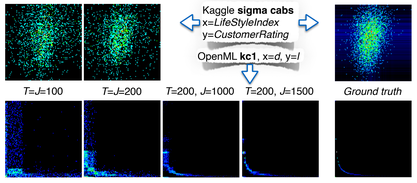
Density learned: example



vs T=1 generative tree, J=50 splits → (Nock & Guillame-Bert, ICML'22)

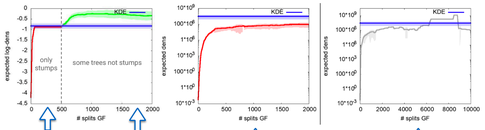
Data generation: examples

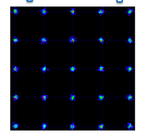
T = #trees
J = *total* #splits



Density estimation

- 5-fold stratified cross validation; model learned is used to compute density on test fold (higher = better)
- Metric: likelihood or log-likelihood
- Contender: kernel density estimation (default kernel+params)





- UCI abalone, T=500 trees (left); T=15 (right)
- Some couples (T,J) clearly better than others (not just for density estimation) → need to explore early stopping or pruning algorithms (future work)

Code & more: