NRGBoost:

Energy-Based Generative Boosted Trees

João Bravo

Feedzai

ICLR 2025

Overview

Introduction

Generative Models for Tabular Data

- Deep Learning has received the most attention
- Focus on sampling and not density estimation

Introduction

Generative Models for Tabular Data

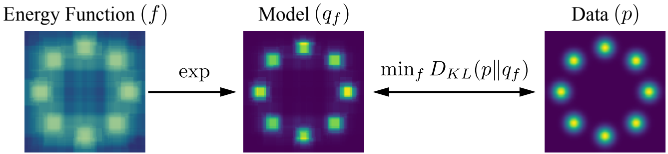
- Deep Learning has received the most attention
- Focus on sampling and not density estimation

Our Contribution: extend **Gradient-Boosted Trees** to generative modeling

- Tree-based generative model capable of (unnormalized) density estimation
- Outperforms other generative models at inference tasks
- Competitive with Deep Learning approaches for sampling

Energy-Based Generative Boosting

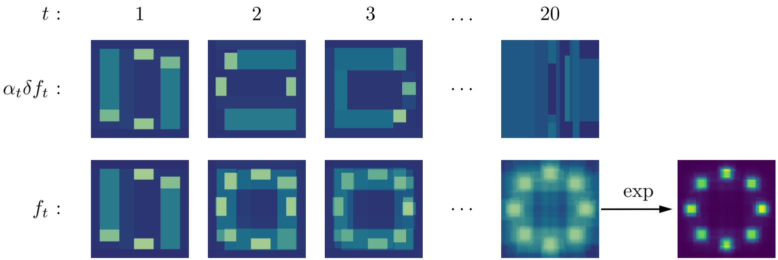
Goal: Approximate a data distribution, p, with an EBM: $q_f(\mathbf{x}) = \frac{\exp(f(\mathbf{x}))}{Z[f]}$



Energy-Based Generative Boosting

Goal: Approximate a data distribution, p, with an EBM: $q_f(\mathbf{x}) = \frac{\exp(f(\mathbf{x}))}{Z[f]}$

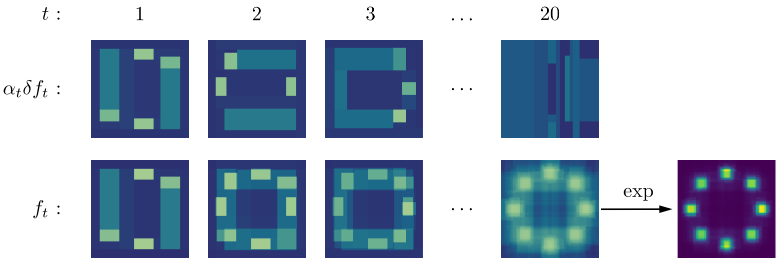
Boosting: Each round add a new δf_t to the energy function: $f_t = f_{t-1} + \alpha_t \delta f_t$



Energy-Based Generative Boosting

Goal: Approximate a data distribution, p, with an EBM: $q_f(\mathbf{x}) = \frac{\exp(f(\mathbf{x}))}{Z[f]}$

Boosting: Each round add a new δf_t to the energy function: $f_t = f_{t-1} + \alpha_t \delta f_t$



δf_t chosen to maximize a local quadratic approximation to the log-likelihood at f_{t-1}

Newton's method in the space of energy functions

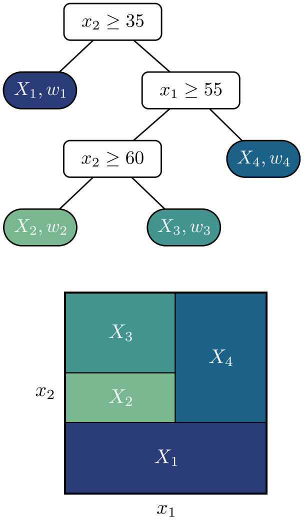
Weak Learners

Each δf is a piecewise constant function given by a binary tree

$$\delta f(\mathbf{x}) = \sum_{j=1}^{J} w_j \mathbf{1}_{X_j}(\mathbf{x})$$

Leaf Value

Leaf Support



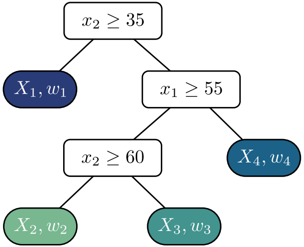
Weak Learners

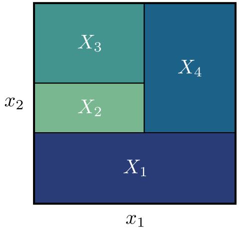
Each δf is a piecewise constant function given by a binary tree

$$\delta f(\mathbf{x}) = \sum_{j=1}^{J} w_j \mathbf{1}_{X_j}(\mathbf{x})$$

Choose X_j and w_j that maximize a quadratic approximation to the **log-likelihood** at current iterate f:

$$X_1^*, \ldots, X_J^* = \underset{X_1,\ldots,X_J}{\arg\max} \underbrace{\sum_{j=1}^{J} \frac{P^2(X_j)}{Q_f(X_j)}}_{\text{Splitting Criterion}}, \quad w_j^* = \frac{P(X_j^*)}{Q_f(X_j^*)} - 1$$

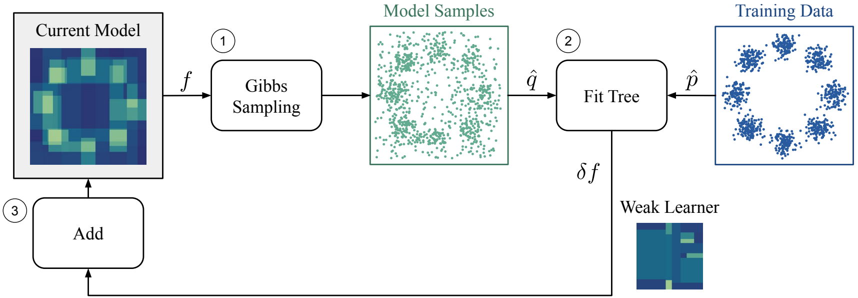




Sampling

Need to estimate two types of quantities:

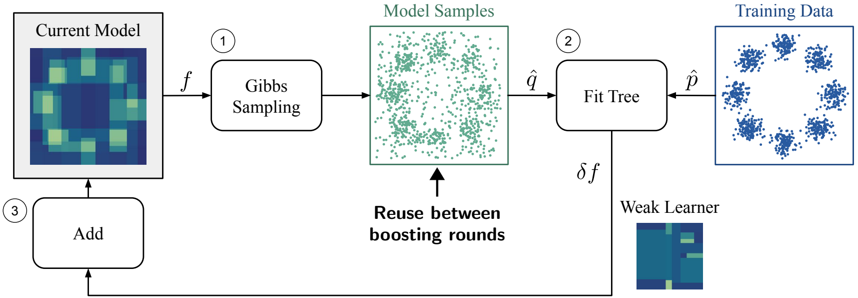
- $P(X)$: Using empirical training data
- $Q_f(X)$: Using samples drawn from the model



Sampling

Need to estimate two types of quantities:

- $P(X)$: Using empirical training data
- $Q_f(X)$: Using samples drawn from the model

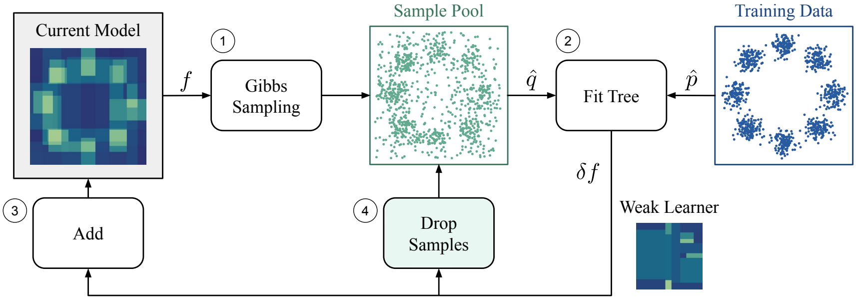


Model may not change significantly between consecutive rounds of boosting

Sampling

Need to estimate two types of quantities:

- $P(X)$: Using empirical training data
- $Q_f(X)$: Using samples drawn from the model



Model may not change significantly between consecutive rounds of boosting

Use rejection sampling to retain samples from previous round that conform to new model

Inference Tasks

An EBM can be used directly for inference over **any** input variable:

$$q_f(y|\mathbf{x}) = \frac{\exp\left(f(y, \mathbf{x})\right)}{\sum_{y'} \exp\left(f(y', \mathbf{x})\right)}$$

Inference Tasks

An EBM can be used directly for inference over **any** input variable:

$$q_f(y|\mathbf{x}) = \frac{\exp\left(f(y, \mathbf{x})\right)}{\sum_{y'} \exp\left(f(y', \mathbf{x})\right)}$$

	$R^2 \uparrow$			AUC $\uparrow$		Accuracy $\uparrow$	
	AB	CH	PR	AD	MBNE	MNIST	CT
RFDE	0.071 $\pm$0.096	0.340 $\pm$0.004	0.059 $\pm$0.007	0.862 $\pm$0.002	0.668 $\pm$0.008	0.302 $\pm$0.010	0.679 $\pm$0.002
ARF	0.531 $\pm$0.032	0.758 $\pm$0.009	0.591 $\pm$0.007	0.893 $\pm$0.002	0.968 $\pm$0.001	-	0.938 $\pm$0.005
DEF (ISE)	0.467 $\pm$0.037	0.737 $\pm$0.008	0.566 $\pm$0.002	0.854 $\pm$0.003	0.653 $\pm$0.011	0.206 $\pm$0.011	0.790 $\pm$0.003
DEF (KL)	0.482 $\pm$0.027	0.801 $\pm$0.008	0.639 $\pm$0.004	0.892 $\pm$0.001	0.939 $\pm$0.001	0.487 $\pm$0.007	0.852 $\pm$0.002
NRGBoost	**0.547 $\pm$0.036**	**0.850 $\pm$0.011**	**0.676 $\pm$0.009**	**0.920 $\pm$0.001**	**0.974 $\pm$0.001**	**0.966 $\pm$0.001**	**0.948 $\pm$0.001**

Table: Discriminative performance of different methods at inferring the value of a single target variable

Inference Tasks

An EBM can be used directly for inference over **any** input variable:

$$q_f(y|\mathbf{x}) = \frac{\exp\left(f(y, \mathbf{x})\right)}{\sum_{y'} \exp\left(f(y', \mathbf{x})\right)}$$

	$R^2 \uparrow$			AUC $\uparrow$		Accuracy $\uparrow$	
	AB	CH	PR	AD	MBNE	MNIST	CT
RFDE	0.071 ±0.096	0.340 ±0.004	0.059 ±0.007	0.862 ±0.002	0.668 ±0.008	0.302 ±0.010	0.679 ±0.002
ARF	0.531 ±0.032	0.758 ±0.009	0.591 ±0.007	0.893 ±0.002	0.968 ±0.001	-	0.938 ±0.005
DEF (ISE)	0.467 ±0.037	0.737 ±0.008	0.566 ±0.002	0.854 ±0.003	0.653 ±0.011	0.206 ±0.011	0.790 ±0.003
DEF (KL)	0.482 ±0.027	0.801 ±0.008	0.639 ±0.004	0.892 ±0.001	0.939 ±0.001	0.487 ±0.007	0.852 ±0.002
NRGBoost	**0.547 ±0.036**	**0.850 ±0.011**	**0.676 ±0.009**	**0.920 ±0.001**	**0.974 ±0.001**	**0.966 ±0.001**	**0.948 ±0.001**
NGBoost	0.546 ±0.040	0.829 ±0.009	0.621 ±0.005	-	-	-	-
XGBoost	0.552 ±0.035	0.849 ±0.009	0.678 ±0.004	0.927 ±0.000	0.987 ±0.000	0.976 ±0.002	0.971 ±0.001

Table: Discriminative performance of different methods at inferring the value of a single target variable

Inference with a Missing Feature

An EBM can also be used for inference with a missing input variable, z:

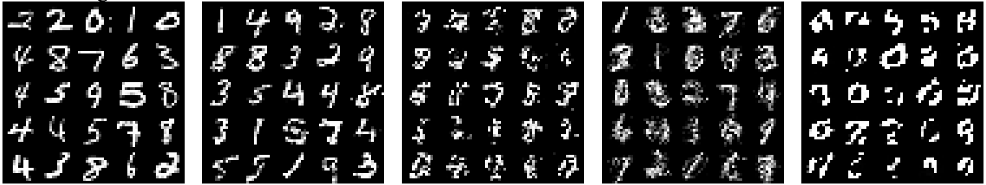
$$q_f(y|\mathbf{x}) = \frac{\sum_z \exp\left(f(y, z, \mathbf{x})\right)}{\sum_{y', z} \exp\left(f(y', z, \mathbf{x})\right)}$$

Model	**Imputation**	**CH** (R^2 ↑)	**AD** (AUC ↑)	**CT** (Accuracy ↑)
XGBoost	Full Data	0.849 ±0.009	0.927 ±0.000	0.971 ±0.001
	Mean	-0.283 ±0.107	N/A	0.610 ±0.004
	Median/Mode	-0.117 ±0.107	0.914 ±0.003	0.621 ±0.002
	KNN (K=5)	0.150 ±0.107	0.910 ±0.003	0.883 ±0.001
NRGBoost	Full Data	0.850 ±0.011	0.920 ±0.001	0.948 ±0.001
	Marginalization	**0.773 ±0.010**	**0.920 ±0.001**	**0.923 ±0.001**

Table: Discriminative performance for inference with a missing covariate

Sample Quality

Training Data | NRGBoost | Forest-Flow | ARF | TVAE



	AB	CH	PR	AD	MBNE	MNIST	CT
TVAE	0.971 ±0.004	0.834 ±0.006	0.940 ±0.002	0.898 ±0.001	1.000 ±0.000	1.000 ±0.000	0.999 ±0.000
TabDDPM	0.818 ±0.015	0.667 ±0.005	**0.628 ±0.004**	0.604 ±0.002	**0.789 ±0.002**	-	0.915 ±0.007
Forest-Flow	0.987 ±0.002	0.926 ±0.002	0.885 ±0.002	0.932 ±0.002	1.000 ±0.000	1.000 ±0.000	0.985 ±0.001
ARF	0.975 ±0.005	0.973 ±0.004	0.795 ±0.008	0.992 ±0.000	0.998 ±0.000	1.000 ±0.000	0.989 ±0.001
DEF (KL)	0.823 ±0.013	0.751 ±0.008	0.877 ±0.002	0.956 ±0.002	1.000 ±0.000	1.000 ±0.000	0.999 ±0.000
NRGBoost	**0.625 ±0.017**	**0.574 ±0.012**	0.631 ±0.006	**0.559 ±0.003**	0.993 ±0.001	**0.943 ±0.003**	**0.724 ±0.006**

Table: AUC of an XGBoost model trained to distinguish real from generated data (lower is better)

Thank You

- **Paper:** `https://arxiv.org/abs/2410.03535`
- **Github:** `https://github.com/ajoo/nrgboost`
- **PyPI:** `pip install nrgboost`