Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

FXAM: A unified and fast interpretable model for predictive analytics

Yuanyuan Jiang [a], Rui Ding [b,*], Tianchi Qiao [c], Yunan Zhu [d], Shi Han [b], Dongmei Zhang [b]

[a] School of Statistics, Renmin University of China, Haidian District, Beijing, 100872, China
[b] Microsoft Research Asia, Haidian District, Beijing, 100080, China
[c] School of Computer Science and Engineering, Southeast University, Nanjing, Jiangsu Province, 211189, China
[d] School of Information Science and Technology, University of Science and Technology of China, He Fei, Anhui Province, 230031, China

ARTICLE INFO

Keywords:
Generalized additive model
Interpretable machine learning
Predictive analytics
Training efficiency

ABSTRACT

Predictive analytics aims to build machine learning models to predict behavior patterns and use predictions to guide decision-making. Predictive analytics is human involved, thus the machine learning model is preferred to be interpretable. In literature, Generalized Additive Model (GAM) is a standard for interpretability. However, due to the one-to-many and many-to-one phenomena which appear commonly in real-world scenarios, existing GAMs have limitations to serve predictive analytics in terms of both accuracy and training efficiency. In this paper, we propose FXAM (Fast and eXplainable Additive Model), a unified and fast interpretable model for predictive analytics. FXAM extends GAM's modeling capability with a unified additive model for numerical, categorical, and temporal features. FXAM conducts a novel training procedure called Three-Stage Iteration (TSI). TSI corresponds to learning over numerical, categorical, and temporal features respectively. Each stage learns a local optimum by fixing the parameters of other stages. We design joint learning over categorical features and partial learning over temporal features to achieve high accuracy and training efficiency. We prove that TSI is guaranteed to converge to the global optimum. We further propose a set of optimization techniques to speed up FXAM's training algorithm to meet the needs of interactive analysis. Thorough evaluations conducted on diverse data sets verify that FXAM significantly outperforms existing GAMs in terms of training speed, and modeling categorical and temporal features. In terms of interpretability, we compare FXAM with the typical post-hoc approach XGBoost+SHAP on two real-world scenarios, which shows the superiority of FXAM's inherent interpretability for predictive analytics.

1. Introduction

Expert systems are often used in decision-making scenarios (Zimmermann, 1987), especially in the high-stakes domains (Meske, Bunde, Schneider, & Gersch, 2022; Simkute, Luger, Jones, Evans, & Jones, 2021) (such as healthcare, criminal justice, or finance) where they can provide valuable insights and recommendations to help with complex decision-making processes. Predictive analytics is an essential topic in expert systems (Changqing, 2018) and aims to predict behavior patterns from multi-dimensional data and use predictions to guide decision-making (Finlay, 2014; Kumar & Ram, 2021). Multi-dimensional data is conceptually organized in a tabular format that consists of a set of records, where each record is represented by a set of attributes, with one attribute called response (i.e., the target to be predicted) and the others called features (or predictors), which are used to predict the response. A multi-dimensional data set typically consists of three types of features: numerical, categorical, and temporal. Fig. 1 shows an example of a house sale data set with several features, such as $Income$ (numerical), $County$ (categorical), $Selldate$ (temporal), etc., and the response is $Price$. By building an ML model from multi-dimensional data, follow-up analysis is performed, such as understanding existing records or predicting response on a newly unseen record.

Predictive analytics is human-involved and is frequently conducted for high-stakes prediction applications thus the ML model is preferred to be interpretable (Rudin, 2019). In the literature, the Generalized Additive Model (GAM) is a standard for interpretability (Hastie & Tibshirani, 1990). GAM untangles the overall prediction by summing up contributions from each feature (before applying the link function), thus retaining interpretability. Moreover, GAM's training procedure (a.k.a. backfitting) works by iterative smoothing of partial residuals over each feature, which guarantees convergence to an optimal solution (when suitable smoothers are chosen). GAMs are continuously being developed, such as GA2M (Lou, Caruana, Gehrke, & Hooker, 2013),

* Corresponding author.
E-mail addresses: jyy_amy@ruc.edu.cn (Y. Jiang), juding@microsoft.com (R. Ding), tianchi-qiao@seu.edu.cn (T. Qiao), zhuyn@mail.ustc.edu.cn (Y. Zhu), shihan@microsoft.com (S. Han), dongmeiz@microsoft.com (D. Zhang).

https://doi.org/10.1016/j.eswa.2024.123890
Received 12 April 2023; Received in revised form 19 November 2023; Accepted 28 December 2023

County	Heating	Income	#rooms	Sell date	Price
San Diego	Gas	195229	6	1/1/2017	909600
San Diego	Gas	105877	5	1/1/2017	748700
Los Angeles	Gas	106248	5	1/1/2017	773600
Los Angeles	Wall	74604	5	1/1/2017	579200
Alameda	Wall	73933	5	1/2/2017	480800
Alameda	Wall	56705	4	1/2/2017	460800
Los Angeles	Gravity	66822	4	1/2/2017	473500
Los Angeles	Gravity	59690	4	1/2/2017	439300
Los Angeles	Gravity	50056	5	1/2/2017	369800
Los Angeles	Gravity	56674	4	1/2/2017	467200
Kern	Gas	81476	4	1/4/2017	591200
Kern	Wall	45789	3	1/4/2017	352400
Kern	Wall	38234	4	1/4/2017	322700
Kern	Gravity	47209	4	1/4/2017	353700
Los Angeles	Gravity	47593	4	1/5/2017	321600
Alameda	Gravity	79060	4	1/5/2017	377800
Alameda	Gravity	38382	3	1/5/2017	334100
...	...	...	...	...	...

Categorical features | Numerical features | Temporal feature | Response

Fig. 1. An example of the multi-dimensional data set.

GAMut (Hohman, Head, Caruana, DeLine, & Drucker, 2019), multi-class GAM (Zhang et al., 2019), ReluctantGAM (Tay & Tibshirani, 2020), COGAM (Abdul, von der Weth, Kankanhalli, & Lim, 2020), etc. However, due to the *one-to-many* and *many-to-one* phenomena that appear commonly in multi-dimensional data, existing GAMs have limitations in serving predictive analytics.

One-to-many: Learning multiple components from each temporal feature. A numerical feature typically introduces a locally smoothing constraint on its contribution to response, but a temporal feature (e.g., 'Sell date') introduces multiple global constraints from a time-series perspective: it is desirable to identify multiple components from a temporal feature, such as monthly repeating (i.e., seasonality) component, long-term progression pattern (i.e., trend), or aperiodic cycles (Zarnowitz & Ozyildirim, 2006), etc. However, existing GAMs treat a temporal feature as an ordinary numerical feature and thus only learn a single smoothing component. As a result, their model capacity is limited w.r.t. dealing with temporal features.

Many-to-one: Since there is no local smoothing constraint across categorical values, users focus on identifying the contribution of each distinct value (e.g., the extra cost of buying a house when it is located in 'County = LA'). Existing GAMs generally conduct histogram-type smoothing per categorical feature, which converges slowly since only the weights of values of a specific categorical feature are updated in each iteration, while all the other weights (w.r.t. distinct values from other categorical features) are fixed. If the weights of values across all categorical features could be updated simultaneously, we could speed up model training.

Moreover, predictive analytics is often conducted iteratively. Fast training makes the analysis more interactive and continuous, which cannot be easily facilitated by existing GAMs due to their unsatisfactory training speed. To address these challenges, we propose FXAM: a unified, fast, and interpretable model for predictive analytics. FXAM has significant advantages in the following areas:

Modeling. FXAM extends GAM's modeling capability with a unified additive model for numerical, categorical, and temporal features. For each temporal feature, FXAM identifies multiple components in terms of trend and seasonality; FXAM proposes a homogeneous set to model categorical values across all categorical features and represents each value via one-hot encoding.

Training. FXAM conducts a novel training procedure called Three-Stage Iteration (TSI). The three stages correspond to learning over numerical, categorical, and temporal features, respectively. Each stage learns a local optimum by fixing the parameters of other stages. Specifically, we design joint learning over categorical features and partial learning over temporal features to achieve high training efficiency and high accuracy. We also provide theoretical analysis in Theorem 1 to show that TSI converges to a global optimum.

Efficiency. We further propose two optimization techniques (i.e., intelligent sampling and dynamic feature iteration) with theoretical guidance to speed up FXAM's training algorithm to meet the needs of interactive analysis.

In summary, we make the following contributions:

- FXAM extends GAMs modeling capability with a unified model for numerical, categorical, and temporal features.
- We propose FXAM's training procedure: Three Stage Iteration, and prove its convergence and optimality.
- We propose two optimization techniques to speed up FXAM's training algorithm.
- We conduct evaluations and verify that FXAM significantly outperforms existing GAMs in terms of training speed and modeling categorical and temporal features.

2. Related work

Predictive analytics & iML (interactive Machine Learning). Predictive analytics is often conducted for high-stakes prediction applications, such as healthcare, finance, or phishing detection thus the ML model is preferred to be interpretable (Rudin, 2019). Operationally, predictive analytics is often conducted iteratively and interactively, thus iML (interactive Machine Learning) is becoming a cornerstone for predictive analytics (Abdul, Vermeulen, Wang, Lim, & Kankanhalli, 2018; Fails & Olsen Jr, 2003), which requires ML model to respond in an interactive fashion. Therefore, ML model's training efficiency becomes primarily important.

XAI (Explainable artificial intelligence). XAI is becoming a hot topic (Kaur et al., 2020; Lombrozo, 2006; Miller, 2019) and current XAI techniques can generally be grouped into two categories (Arrieta et al., 2020; Du, Liu, & Hu, 2019). **Interpretable**: designing inherently explainable ML models (Caruana et al., 2015; Jung, Concannon, Shroff, Goel, & Goldstein, 2017; Lou et al., 2013) or **Explainable**: providing post-hoc explanations to opaque models (Lundberg & Lee, 2017; Ribeiro, Singh, & Guestrin, 2016; Tan, Caruana, Hooker, Koch, & Gordo, 2018), depending on the time when explainability is obtained (Molnar, 2020). In the domain of predictive analytics, interpretable ML models tend to be more useful since explainability is needed throughout the analysis process, such as probing different subsets of data, incorporating domain constraints, or understanding model mechanisms locally or globally. FXAM is an extension of GAM, thus retaining interpretability.

Generalized Additive Models (GAMs). GAMs are gaining great attention in the literature of interpretable machine learning (Arrieta et al., 2020; Chang, Tan, Lengerich, Goldenberg, & Caruana, 2021; Linardatos, Papastefanopoulos, & Kotsiantis, 2021; Rudin, 2019), mainly due to its standard for interpretability (Wang et al., 2021) and its broad adoptions in the real world (Calabrese et al., 2012; Pierrot & Goude, 2011; Tomić & Božić, 2014; Wang et al., 2021). GAM-based approaches are continuously being developed: pureGAM (Sun, Wang, Ding, Han, & Zhang, 2022) and GA2M (Lou et al., 2013) model pairwise feature interaction; multi-class GAM (Zhang et al., 2019) generalizes GAM to the multi-class setting; COGAM (Abdul et al., 2020) and

ReluctantGAM (Tay & Tibshirani, 2020) impose linear constraints on certain features to achieve a tradeoff between cognitive load and model accuracy. There also exists work on modeling GAM's shape functions by neural nets such as NAM (Agarwal, Frosst, Zhang, Caruana, & Hinton, 2020) and GAMI-Net (Yang, Zhang, & Sudjianto, 2021).

FXAM is complementary to these works by modeling numerical, categorical, and temporal features in a unified way and by proposing an efficient and accurate training procedure. In FXAM, joint learning is conducted over all categorical features instead of per-feature learning (e.g., histogram-type smoothing in pyGAM) to improve training efficiency; partial learning is adopted to accurately learn trend and seasonality components from each temporal feature. Such an approach can be naturally extended to learn arbitrary components. Although there exists work on identifying seasonality components by adopting cyclic cubic spline, they require additional efforts on data preprocessing (Simpson, 2014), and the learned seasonal component is restricted to be identical in each period thus progressive changes of seasonal component (e.g., amplifying or damping) cannot be captured. Lastly, such a preprocessing approach is difficult to extend to learn other components, such as aperiodic cyclic components (Hyndman, 2011; Hyndman & Athanasopoulos, 2018).

3. Terms and notations

Except for special instructions, we use uppercase italics for variables, uppercase bold letters for matrices, lowercase bold letters for vectors, lowercase letters for scalars, subscripts for the variable index, and superscripts with parentheses for the instance index. Our discussion will center on a response random variable Y, and p numerical features $X_1, \dots, X_p$; q categorical features $Z_1, \dots, Z_q$; u temporal features $T_1, \dots, T_u$. Given a multi-dimensional data set D consists of N instances, the realizations of these random variables can be denoted by $(y^{(1)}, x_1^{(1)}, \dots, x_p^{(1)}, z_1^{(1)}, \dots, z_q^{(1)}, t_1^{(1)}, \dots, t_u^{(1)}), \dots, (y^{(N)}, x_1^{(N)}, \dots, x_p^{(N)}, z_1^{(N)}, \dots, z_q^{(N)}, t_1^{(N)}, \dots, t_u^{(N)})$. The summary of terms and notations is shown in Table 1.

Categorical features. For each $m \in 1, \dots, q$, denote the domain of Z_m as $dom(Z_m)$, which indicates the distinct values for categorical feature Z_m. For instance, each element in $dom(Z_m)$ can be a string value that is composed of the specific value in Z_m with the corresponding feature name as the suffix. Hence, the domains of different categorical features are disjoint.

Denote $H_{cat} = \bigcup_{m=1}^{q} dom\left(Z_m\right)$ as the homogeneous set, and $c = |H_{cat}|$ as total cardinality (i.e., number of distinct values) over all categorical features. Denote $O_j \in \{0,1\}$, $j = 1, 2, \dots, c$, thus any instantiation of $Z_1, \dots, Z_q$ can be represented by a unique q-hot vector $\left(O_1, \dots, O_c\right)$ provided that pre-specified indices are assigned to elements in H_{cat}. Continuing with the example in Fig. 1, the categorical variables can be processed as shown in Fig. 2, where each row represents a q-hot vector, and the jth column represents the variable O_j.

Numerical features. Following standard convention, for each $i \in 1, \dots, p$, let $\mathcal{H}_{num}^{i}$ denote the Hilbert space of measurable functions $f_i(X_i)$ such that $E\left[f_i\right] = 0$, $E\left[f_i^2\right] < \infty$ and inner product $\left\langle f_i, f_i' \right\rangle = E\left[f_i f_i'\right]$. Here the expectation is defined over the probability density distribution corresponding to the training data. For our purpose, we would like to learn (or estimate) a shape function $f_i(X_i) \in H_{num}^{i}$ for each numerical feature.

Temporal features. To identify trend component, for each $k \in 1, \dots, u$, let $\mathcal{H}_{tem}^{k}$ denote the Hilbert space of measurable function $f_{T_k}(T_k)$ for trend and $f_{S_k}(T_k)$ for seasonality over temporal feature T_k. $\mathcal{H}_{tem}^{k}$ is with the same property as $\mathcal{H}_{num}^{i}$. To identify the seasonal component, denote the period of the seasonal component as a positive integer $d_k > 1$. Note that d_k is an input parameter based on domain knowledge, which is common practice in the business data analytics domain Cleveland, Cleveland, McRae, and Terpenning (1990), Wen et al. (2019).

Table 1
A summary of terms and notations.

Type	Symbol	Explanation
Data set	D	Data set
	N	Data set size
Response	Y	Random variable
	$\boldsymbol{y}^T = \left(y^{(1)}, y^{(2)} \dots, y^{(N)}\right)$	Instances
Predictors	$X_1, X_2, \dots, X_p$	p numerical features
	$Z_1, Z_2, \dots, Z_q$	q categorical features
	$T_1, T_2, \dots, T_u$	u temporal features
Categorical features	$dom(Z_m),\ m = 1, \dots, q$	The set including the distinct values for the categorical feature Z_m
	$H_{cat} = \bigcup_{m=1}^{q} dom\left(Z_m\right)$	The homogeneous set including the distinct values for all categorical features
	$c = \lvert H_{cat} \rvert$	Total cardinality over all categorical features
	$(O_1, O_2, \dots, O_c)$ where $O_j \in \{0,1\}$, $j = 1, \dots, c$	The q-hot vector representing the encoding of $Z_1, \dots, Z_q$
	$f_Z\left(O_j\right) = \beta_j O_j$	The parameterized form by representing categorical values $Z_1, \dots, Z_q$ in the c-dimensional vector
Numerical features	$H_{num}^{i},\ i = 1, \dots, p$	The Hilbert space of the measurable function $f_i(X_i)$ over numerical feature X_i
	$f_i(X_i)$	The univariate smooth function modeling the contributions of X_i, $f_i(X_i) \in H_{num}^{i}$
Temporal features	$H_{tem}^{k},\ k = 1, \dots, u$	The Hilbert space of the measurable functions $f_{S_k}(T_k)$ for seasonality and $f_{T_k}(T_k)$ for trend over temporal feature T_k
	$f_{S_k}(T_k)$	The function modeling the seasonality of T_k, $f_{S_k}(T_k) \in H_{tem}^{k}$
	$f_{T_k}(T_k)$	The function modeling the trend of T_k, $f_{T_k}(T_k) \in H_{tem}^{k}$
	d_k	The period of seasonal component, $d_k > 1$ and d_k is a hyperparameter;
	$dom(T_k) = \{t_k^{(1)}, \dots, t_k^{(N)}\}$	The set including all the ordered values of T_k in D, where $t_k^{(1)} \le \dots \le t_k^{(N)}$
	$t_k^{(l)} - t_k^{(l-1)} = 0$ or τ	The corresponding gap between two consecutive time points, τ is a constant, $l = 2, \dots, N$
	$\mathcal{T}_{k,\varphi} := \left\{ t_k^{(l)} \mid t_k^{(l)} / \tau \bmod d_k = \varphi \right\}$	The set of time points with phase-φ, where $\varphi \in \{0, \dots, d_k - 1\}$, $l = 1, \dots, N$
	$dom(T_k) = \bigcup_{\varphi=0}^{d_k-1} \mathcal{T}_{k,\varphi}$	$dom(T_k)$ can be written as the sum of $\mathcal{T}_{k,\varphi}$

We order the values of T_k in D as: $t_k^{(1)} \le \dots \le t_k^{(N)}$, and assume $\left\{ t_k^{(l)} - t_k^{(l-1)} | l = 2, \dots, N \right\} = \{0, \tau\}$ for the sake of simplicity: continuing with the example in Fig. 1, it is common that there would be multiple instances with the same value on $t_k^{(*)}$ as shown in Fig. 3(a). Instances with the same value at time $t_k^{(*)}$ are compressed into one instance with weight w_* as shown in Fig. 3(b). Thus the corresponding gap between two consecutive time points $t_k^{(l)} - t_k^{(l-1)} = 0$ or $t_k^{(l)} - t_k^{(l-1)} = \tau$ allows us to treat $t_k^{(*)}$ as discrete time points. Now it is easy to

$dom(Z_1)$ $dom(Z_2)$

County_0	County_1	County_2	County_3	Heating_0	Heating_1	Heating_2
1	0	0	0	1	0	0
1	0	0	0	1	0	0
0	1	0	0	1	0	0
0	1	0	0	0	1	0
0	0	1	0	0	1	0
0	0	1	0	0	1	0
0	1	0	0	0	0	1
0	1	0	0	0	0	1
0	1	0	0	0	0	1
0	1	0	0	0	0	1
0	0	0	1	1	0	0
0	0	0	1	0	1	0
0	0	0	1	0	1	0
0	0	0	1	0	0	1
0	1	0	0	0	0	1
0	0	1	0	0	0	1
0	0	1	0	0	0	1
…	…	…	…	…	…	…

Fig. 2. An example of the processed categorical variables.

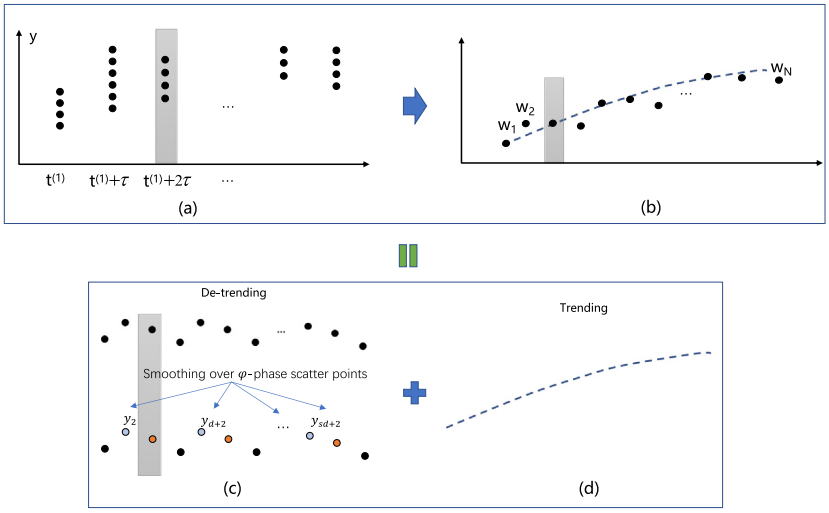

Fig. 3. An example of the processed temporal variables.

decompose the time series of temporal feature T_k into a trend f_{T_k} and a seasonality f_{S_k} as shown in Fig. 3(c) and Fig. 3(d). Denote $\mathcal{T}_{k,\varphi} := \left\{ t_k^{(l)} \mid t_k^{(l)}/\tau \bmod d_k = \varphi, \forall l \right\}$ as phase-φ set since all the elements in $\mathcal{T}_{k,\varphi}$ share same phase $\varphi \in \{0, 1, \ldots, d_k - 1\}$. As shown in Fig. 3(c), instances of the same color belong to the same set $\mathcal{T}_{k,\varphi}$. It is easy to see that $\mathcal{T}_{k,i}$, $\mathcal{T}_{k,j}$ $(i, j \in \{0, 1, \ldots, d_k - 1\}$ and $i \neq j)$ are mutually disjoint and thus $\left\{ t_k^{(1)}, \ldots, t_k^{(N)} \right\} = \mathcal{T}_{k,0} + \cdots + \mathcal{T}_{k,d_k-1}$. Our approach can also easily deal with missing data as the shaded part shown in Fig. 3 (i.e., the gap between two consecutive time points could be larger than τ in a data set due to insufficient sample), which is discussed in Section 4.6.

4. Approach

Initially, we elucidate the principles of the classical Generalized Additive Model (GAM), followed by a comprehensive discourse on the utilization of the Backfitting algorithm, a prominent method employed for the resolution of the GAM model. Then we illustrate the FXAM's modeling over numerical, categorical, and temporal features, and propose FXAM's training procedure called Three-Stage Iteration. At last, two optimization techniques are presented to further improve FXAM's training efficiency.

4.1. GAMs' modeling

Linear models, while straightforward to interpret and apply, frequently struggle to adequately capture the intricate and evolving relationships found in real-world scenarios. Machine learning models such as deep neural networks are proficient in fitting complex non-linear relationships, however, they often act as black boxes, making their outcomes challenging to interpret.

Backfitting Algorithm

Initialize:

$\boldsymbol{f}_i = \mathbf{0}, i = 1, \ldots, m$

1: **Cycle** $i = 1,2, \ldots m, 1,2, \ldots, m, \ldots,$

2: $\boldsymbol{y}_i = \boldsymbol{y} - \sum_{k=1,k\neq i}^{m} \boldsymbol{f}_k$

3: $\boldsymbol{f}_i = smoothing\ over \left\{(x_i^{(l)}, y_i^{(l)}) | l = 1, \ldots, N\right\}$

4: **Until** $\boldsymbol{f}_i$ does not change, $\forall i$

Fig. 4. Backfitting algorithm procedure.

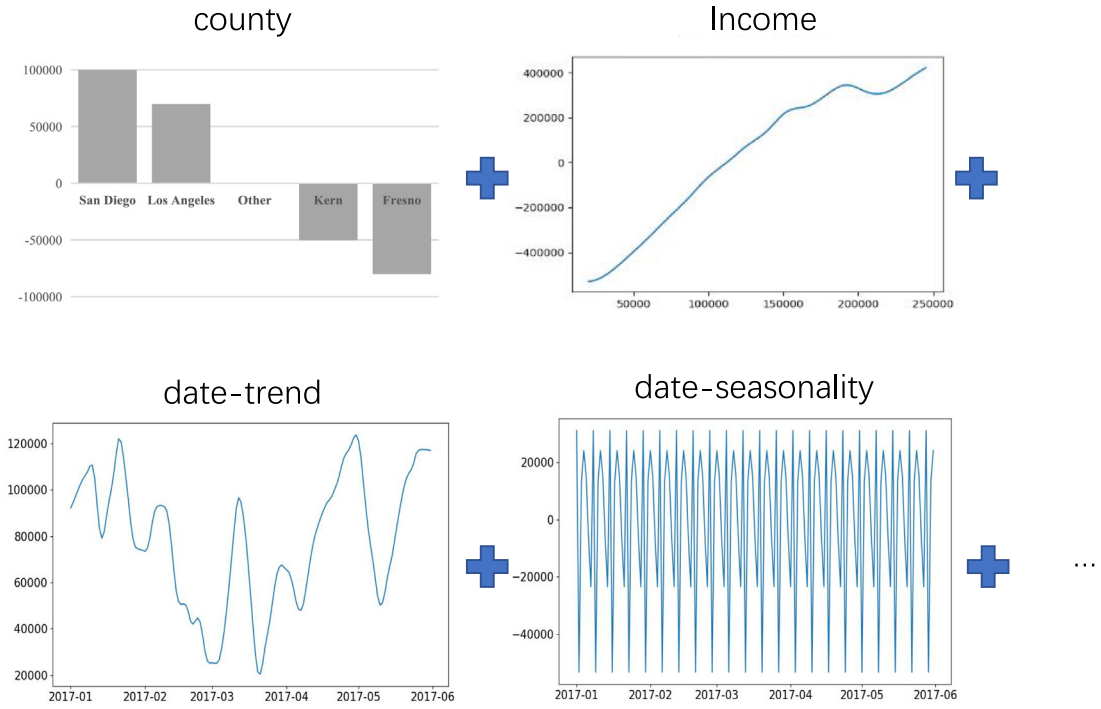

Fig. 5. Illustrative example of FXAM.

GAMs (Hastie & Tibshirani, 1990) presents an effective intermediate solution. It has the capacity to represent non-linear relationships in a flexible manner while maintaining interpretability. This characteristic aids in understanding the model's structure and its outcomes, facilitating statistical inferences. Therefore, GAMs provide an advantageous balance between model complexity and interpretability, making them a crucial tool in data-driven research. A prototypical formulation of a GAM can be outlined as follows:

$$g(E(Y)) = \sum_{i=1}^{m} f_i\left(X_i\right) \tag{1}$$

Here $f_i(\cdot)$ in the equation can take forms that are parametric, non-parametric, or semi-parametric, providing flexibility in model specification. The function $g(\cdot)$ acts as the link function, serving as the bridge that connects the predictors and the mean of the response variable.

Without loss of generality, we focus on the case where link function $g(\cdot)$ is an identity function, thus we are focusing on the regression problem subsequently. Once the model is specified, the smooth functions $f_i(X_i)$ can be estimated from the data. This is typically done using a technique known as backfitting. Its specific procedural flow is graphically delineated in Fig. 4.

The backfitting algorithm is an iterative procedure that estimates each shape function $f_i(X_i)$ in turn, holding the others fixed. The algorithm starts with initial estimates for the functions, and then repeatedly cycles through the predictors, updating the estimate for each predictor's function while keeping the others fixed, until the estimates converge. At each step, the algorithm fits a model to the residuals from the previous step and then adds the fitted values to the current estimate of the function. This process is repeated until the changes in the functions are negligible, which indicates that the algorithm has converged to a solution.

Backfitting is a simple and computationally efficient method for fitting GAMs, especially when the number of predictors is large. However, it may converge slowly. To enhance the convergence rate, three principal enhancements have been incorporated into the FXAM model, compared to the traditional GAM model. Firstly, an intelligent sampling algorithm has been employed during the initialization phase (as depicted in Fig. 5) to achieve a superior initialization function. Secondly, targeting each variable in line 1 of Fig. 4, the Dynamic Feature Iteration algorithm is proposed to prioritize iterating those variables possessing potent predictive abilities. Lastly, during the line 3 smoothing phase in Fig. 4, Fast Kernel Smoothing is utilized at each smoothing instance to perform data fitting. More details are at Section 4.4. Furthermore, the FXAM model extends the GAM model's ability to model categorical features and temporal features, as detailed in Section 4.5.

4.2. FXAM's modeling

We model FXAM as follows:

$$E\left(Y|X_1,\dots,X_p;Z_1,\dots,Z_q;T_1,\dots,T_u\right)=\sum_{i=1}^{p}f_i\left(X_i\right)+\sum_{j=1}^{c}f_Z\left(O_j\right)+\sum_{k=1}^{u}\left[f_{T_k}\left(T_k\right)+f_{S_k}\left(T_k\right)+other\_component\left(T_k\right)\right] \tag{2}$$

Here $f_Z(O_j)=\beta_j O_j$, $j=1,\dots,c$. $\beta_j\in\mathbb{R}$ is the parameter and $f_i\in\mathcal{H}^i_{num}$, $f_{T_k}\in\mathcal{H}^k_{tem}$, $f_{S_k}\in\mathcal{H}^k_{tem}$ are the functions we want to learn. $O_j\in\{0,1\}$ is obtained by one-hot encoding over homogeneous set H_{cat}. The overall model is composed of three parts additively w.r.t. modeling numerical, categorical, and temporal features respectively.

Modeling numerical features. Following standard convention, f_i is the shape function that models the contribution of X_i (w.r.t. response Y) by a univariate smooth function.

Modeling categorical features. We conduct one-hot encoding for each element in the homogeneous set H_{cat}. Specifically, $\sum_{j=1}^{c}f_Z\left(O_j\right)=\sum_{j=1}^{c}\beta_j O_j$ is a parameterized form by representing categorical values $Z_1,\dots,Z_q$ in a c-dimensional q-hot vector and assigns a weight β_j to each entry O_j.

Modeling temporal features. $\left[f_{T_k}\left(T_k\right)+f_{S_k}\left(T_k\right)+\cdots\right]$ explicitly decomposes the time series of temporal feature T_k into a trend f_{T_k}, a seasonality f_{S_k} and some other signals. Such decomposition expresses multiple components from a single feature to address the one-to-many phenomenon.

Continuing with the example in Fig. 1, the final housing prices can be understood as the summation of these shape functions, as depicted in Fig. 5.

For the sake of simplicity, we focus on seasonal and trend decomposition, and we assume $u=1$ (i.e., only one temporal feature) in subsequent illustrations. We thus drop the subscript k and use T to denote the temporal feature. Note that the theorem of FXAM's convergence is valid for arbitrary u and FXAM's training procedure can be easily extended to support multiple temporal features (details are elaborated in Section 4.6).

4.3. FXAM's optimization

The objective function we want to minimize is:

$$\mathcal{L}\left(f_1,\dots,f_p,\beta_1,\dots,\beta_c,f_T,f_S\right)=\sum_{l=1}^{N}\left(y^{(l)}-\sum_{i=1}^{p}f_i\left(x_i^{(l)}\right)-\sum_{j=1}^{c}f_Z\left(o_j^{(l)}\right)-f_T\left(t^{(l)}\right)-f_S\left(t^{(l)}\right)\right)^2+\lambda\sum_{i=1}^{p}J\left(f_i\right)+\lambda_Z\boldsymbol{\beta}^T\boldsymbol{\beta}+\lambda_T J(f_T)+\lambda_S\sum_{\varphi=0}^{d-1}J\left(f_{S_\varphi}\right) \tag{3}$$

Here $\boldsymbol{\beta}^T=(\beta_1,\dots,\beta_c)$. Eq. (3) consists of the total square error and the other regularization items. Functional $J(f):=\int\left[f''(v)\right]^2dv$ thus $\lambda J(f)$ trades off the smoothness of f with its goodness-to-fit. In addition to standard regularization for numerical features $\lambda J\left(f_i\right)$ and trend $\lambda_T J(f_T)$, we divide seasonal component f_S into d sub-components f_{S_φ} ($f_S:=f_{S_0}\oplus\cdots\oplus f_{S_{d-1}}$ indicates that overall seasonal component f_S which domain-merges all the sub-components f_{S_φ}) and apply regularization per f_{S_φ}. By doing so, we impose smoothness for each phase-equivalent sub-component f_{S_φ}, which is helpful to convey the overall repeating pattern. We propose standard L_2 regularization $\lambda_Z\boldsymbol{\beta}^T\boldsymbol{\beta}$ correspondingly.

Quadratic Form of Objective Function. By standard calculus (Reinsch, 1967), the optimal solution for minimizing a square error with regularization $\lambda J\left(f_i\right)$ is natural cubic spline smoothing with knots at $x_i^{(1)},\dots,x_i^{(N)}$, thus the vector version of objective function $\mathcal{L}$ can be expressed as a quadratic form:

$$\mathcal{L}\left(\boldsymbol{f}_1,\dots,\boldsymbol{f}_p,\boldsymbol{f}_Z,\boldsymbol{f}_T,\boldsymbol{f}_S\right)=\left\|\boldsymbol{y}-\sum_{i=1}^{p}\boldsymbol{f}_i-\boldsymbol{f}_Z-\boldsymbol{f}_T-\boldsymbol{f}_S\right\|^2+\lambda\sum_{i=1}^{p}\boldsymbol{f}_i^T\boldsymbol{K}_i\boldsymbol{f}_i+\lambda_Z\boldsymbol{\beta}^T\boldsymbol{\beta}+\lambda_T\boldsymbol{f}_T^T\boldsymbol{K}_T\boldsymbol{f}_T+\lambda_S\sum_{\varphi=0}^{d-1}\boldsymbol{f}_{S_\varphi}^T\boldsymbol{K}_{S_\varphi}\boldsymbol{f}_{S_\varphi} \tag{4}$$

In Eq. (4), $\|*\|^2$ denotes the total square error and we use $\boldsymbol{f}_i,\boldsymbol{f}_Z,\boldsymbol{f}_T$ and $\boldsymbol{f}_S\in\mathbb{R}^N$ as the vector version realizations of f_i,f_Z,f_T,f_S in Eq. (3) respectively. Here $\boldsymbol{f}_Z=\boldsymbol{Z}\boldsymbol{\beta}$, $\boldsymbol{Z}$ is a $N\times c$ design matrix corresponding to N q-hot encoded vectors from categorical features as shown in Fig. 2. $\boldsymbol{y}^T=(y^{(1)},\dots,y^{(N)})$, $\boldsymbol{f}_i^T=\left(f_i(x_i^{(1)}),\dots,f_i(x_i^{(N)})\right)$, $\boldsymbol{f}_T^T=\left(f_T(t^{(1)}),\ \dots,f_T(t^{(N)})\right)$, $\boldsymbol{f}_S^T=\left(f_S(t^{(1)}),\dots,f_S(t^{(N)})\right)$. $\boldsymbol{K}_i$ is a $N\times N$ matrix pre-calculated by values $x_i^{(1)},\dots,x_i^{(N)}$ (Buja, Hastie, & Tibshirani, 1989). $\boldsymbol{K}_T$ is calculated the same way. $\boldsymbol{K}_{S_\varphi}$ is an $N\times N$ matrix obtained by applying cubic spline smoother over $\mathcal{T}_\varphi$ and then re-ordering the indices of records with a permutation matrix $\boldsymbol{P}_\varphi$. Specifically, $\boldsymbol{K}_{S_\varphi}=\boldsymbol{P}_\varphi^T\begin{bmatrix}\widetilde{\boldsymbol{K}_{S_\varphi}}&0\\0&0\end{bmatrix}\boldsymbol{P}_\varphi$, where $\widetilde{\boldsymbol{K}_{S_\varphi}}$ is a $\left|\mathcal{T}_\varphi\right|\times\left|\mathcal{T}_\varphi\right|$ matrix w.r.t. cubic spline smoothing over knots $\left\{t_\varphi^{(1)},t_\varphi^{(2)},\dots,t_\varphi^{\left(\left|\mathcal{T}_\varphi\right|\right)}\right\}$ (i.e. $\mathcal{T}_\varphi$), and $\boldsymbol{P}_\varphi^T$ is a $N\times N$ permutation matrix mapping the indices of these knots into the original indices of elements in T.

Analysis of Optimality. To minimize $\mathcal{L}$ in Eq. (4), we derive $\mathcal{L}$'s stationary solution via FXAM's normal equations:

$$\begin{array}{l}\nabla_{f_i}\mathcal{L}=0|_{i:1,\dots,p}\\\nabla_{\boldsymbol{\beta}}\mathcal{L}=0\\\nabla_{f_T}\mathcal{L}=0\\\nabla_{f_{S_\varphi}}\mathcal{L}=0|_{\varphi:0,\dots,d-1}\end{array}\Rightarrow$$

$$\begin{bmatrix}\boldsymbol{I}&\boldsymbol{M}_Z&\boldsymbol{M}_Z&\boldsymbol{M}_Z&\dots&\boldsymbol{M}_Z\\\boldsymbol{M}_1&\boldsymbol{I}&\boldsymbol{M}_1&\boldsymbol{M}_1&\dots&\boldsymbol{M}_1\\\boldsymbol{M}_2&\boldsymbol{M}_2&\boldsymbol{I}&\boldsymbol{M}_2&\dots&\boldsymbol{M}_2\\\vdots&\vdots&\vdots&\ddots&\vdots&\vdots\\\boldsymbol{M}_T&\boldsymbol{M}_T&\boldsymbol{M}_T&\dots&\boldsymbol{I}&\boldsymbol{M}_T\\\boldsymbol{M}_S&\boldsymbol{M}_S&\boldsymbol{M}_S&\dots&\boldsymbol{M}_S&\boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{f}_Z\\\boldsymbol{f}_1\\\boldsymbol{f}_2\\\vdots\\\boldsymbol{f}_T\\\boldsymbol{f}_S\end{bmatrix}=\begin{bmatrix}\boldsymbol{M}_Z\boldsymbol{y}\\\boldsymbol{M}_1\boldsymbol{y}\\\boldsymbol{M}_2\boldsymbol{y}\\\vdots\\\boldsymbol{M}_T\boldsymbol{y}\\\boldsymbol{M}_S\boldsymbol{y}\end{bmatrix}$$

where

$$\begin{cases}\boldsymbol{M}_Z=\boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{Z}+\lambda_Z\boldsymbol{I}\right)^{-1}\boldsymbol{Z}^T\\\boldsymbol{M}_i=\left(\boldsymbol{I}+\lambda\boldsymbol{K}_i\right)^{-1},\forall i\in\{1,\dots,p\}\\\boldsymbol{M}_T=\left(\boldsymbol{I}+\lambda_T\boldsymbol{K}_T\right)^{-1}\\\boldsymbol{M}_S=\boldsymbol{P}^T\begin{bmatrix}\left(\boldsymbol{I}+\lambda_S\widetilde{\boldsymbol{K}_{S_0}}\right)^{-1}&\dots&0\\\vdots&\ddots&\vdots\\0&\dots&\left(\boldsymbol{I}+\lambda_S\widetilde{\boldsymbol{K}_{S_{d-1}}}\right)^{-1}\end{bmatrix}\boldsymbol{P}\end{cases} \tag{5}$$

In Eq. (5), $\boldsymbol{P}=\boldsymbol{P}_0\dots\boldsymbol{P}_{d-1}$ is the overall permutation matrix, by mapping indices of elements from $\left\{t^{(1)},\dots,t^{(N)}\right\}$ to the indices of elements in $\left\{\mathcal{T}_0,\dots,\mathcal{T}_{d-1}\right\}$. Then estimate $\boldsymbol{f}_1,\dots,\boldsymbol{f}_p,\boldsymbol{f}_Z,\boldsymbol{f}_T,\boldsymbol{f}_S$ by given $\boldsymbol{M}_1,\dots,\boldsymbol{M}_p,\boldsymbol{M}_Z,\boldsymbol{M}_T,\boldsymbol{M}_S$ and $\boldsymbol{y}$. By definition, a solution of FXAM's normal equations is a local optimum of $\mathcal{L}$, below we further prove that the solution also achieves the global optimum of $\mathcal{L}$.

Theorem 1. *Solutions of FXAM's normal equations exist and are the global optimum.*

Proof. Here is the sketch proof, more details are at Appendix A. According to theorem 2 in Buja et al. (1989), the solutions of normal equations exist and are globally optimal if each smoothing matrix

FXAM's training: TSI (Three Stage Iterations)

Initialize:
$\boldsymbol{\beta} = \mathbf{0}, \boldsymbol{f}_Z = \mathbf{0}$ /* parameters for categorical features */
$\boldsymbol{f}_i = \mathbf{0}, i = 1, \dots, p$ /* parameters for numerical features */
$\boldsymbol{f}_T = \mathbf{0}, \boldsymbol{f}_{S_\varphi} = \mathbf{0}, \boldsymbol{f}_S = \mathbf{0}, \varphi = 0, \dots, d-1$ /* parameters for temporal feature */
1: **Cycle**
/* Stage1: learning on numerical features */
2: **Cycle** $i = 1,2, \dots p, 1,2, \dots, p, \dots,$
3: $\boldsymbol{y}_i = \boldsymbol{y} - \boldsymbol{f}_Z - \boldsymbol{f}_T - \boldsymbol{f}_S - \sum_{k=1,k\neq i}^{p} \boldsymbol{f}_k$
4: $\boldsymbol{f}_i = smoothing\ over \left\{(x_i^{(l)}, y_i^{(l)}) | l = 1, \dots, N\right\}$
5: **Until** $\boldsymbol{f}_i$ does not change, $\forall i$
/* Stage2: learning on categorical features */
6: $\boldsymbol{y}_Z = \boldsymbol{y} - \sum_{i=1}^{p} \boldsymbol{f}_i - \boldsymbol{f}_T - \boldsymbol{f}_S$
7: Solve $\boldsymbol{\beta} = (\boldsymbol{Z}^T\boldsymbol{Z} + \lambda_Z \boldsymbol{I})^{-1}\boldsymbol{Z}^T\boldsymbol{y}_Z$
Using Nesterov acceleration + Power method
8: $\boldsymbol{f}_Z = \boldsymbol{Z}\boldsymbol{\beta}$
/* Stage3: learning trend and seasonality on temporal feature */
9: $\boldsymbol{y}_{TS} = \boldsymbol{y} - \sum_{i=1}^{p} \boldsymbol{f}_i - \boldsymbol{f}_Z$
10: $\left\{\left(f_T(t^{(l)}), f_{S_\varphi}(t^{(l)})\right) | l = 1, \dots, N\right\} = seasonal\ trend\ decomposition\ of\ \left\{(t^{(l)}, y_{TS}{}^{(l)}) | l = 1, \dots, N\right\}$
11: $\boldsymbol{f}_S = \sum_{\varphi=0}^{d-1} \boldsymbol{f}_{S_\varphi}$
12: **Until** $\boldsymbol{f}_Z, \boldsymbol{f}_i, \boldsymbol{f}_T, \boldsymbol{f}_S$ does not change

Fig. 6. Three stage iteration procedure.

$\boldsymbol{M}_i, \boldsymbol{M}_Z, \boldsymbol{M}_T$, or $\boldsymbol{M}_S$ is symmetric and shrinking (i.e., with eigenvalues in $[0, 1]$). Thus we check $\boldsymbol{M}_i, \boldsymbol{M}_Z, \boldsymbol{M}_T$, and $\boldsymbol{M}_S$ one by one and prove that they possess these properties. ■

4.4. FXAM's training

To solve FXAM's normal equations, we extend backfitting and develop a novel training procedure: Three-Stage Iteration (TSI). TSI consists of three stages: learning over numerical, categorical, and temporal features, respectively. As shown in Fig. 6: standard backfitting is applied over numerical features (line $2-5$), we additionally design *joint learning* over categorical features (line $6-8$) to improve training efficiency, and *partial learning* over temporal features (line $9-11$) to learn trend and seasonal components to improve accuracy. TSI is more appealing in that it maintains convergence to the solution of FXAM's normal equations, thus the output of TSI is the global optimum of $\mathcal{L}$.

Joint learning on categorical features. To deal with categorical features, existing GAMs conduct per-feature smoothing (e.g., histogram-type smoothing in pyGAM), which converges slowly since only the weights of a specific categorical feature (e.g., Z_1) are updated (e.g., $\beta_1, \dots, \beta_{|Z_1|}$) but all the other weights (e.g., $\beta_{|Z_1|+1}, \dots, \beta_c$) are fixed in each iteration (depicted in Fig. 7). In contrast, we pull all categorical values into a homogeneous set H_{cat} and learn all the parameters $\beta_1, \dots, \beta_c$ jointly. Joint learning enables accelerating gradient descent via adopting improved momentum: we adopt Nesterov acceleration and power method (Nesterov, 1983; Sutskever, Martens, Dahl, & Hinton, 2013) to improve training efficiency (line 7 in Fig. 6). In our experiment, joint learning achieves $3-10$ times faster than per-feature learning.

Nesterov acceleration can be viewed as an improved momentum, thus making convergence significantly faster than gradient descent especially when the cardinality of the homogeneous set is large (Nesterov, 1983; Sutskever et al., 2013). As depicted in line 7 of Fig. 6, the task of learning the contributions of categorical values boils down to calculating $\boldsymbol{\beta} = \left(\boldsymbol{Z}^T\boldsymbol{Z} + \lambda_Z \boldsymbol{I}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{y}_Z$. We adopt Nesterov's Gradient Acceleration (NGA) to estimate $\boldsymbol{\beta}$ together with power iteration to identify optimal learning rate μ. The adoption of NGA with power iteration in our problem is depicted on the left side of Fig. 9. The optimal learning rate μ equals the greatest eigenvalue of $\boldsymbol{Z}^T\boldsymbol{Z} + \lambda_Z \boldsymbol{I}$, which can be efficiently identified by power iteration. The complexity of our algorithm is $O\left(kc^2\right)$ where k is #iterations of NGA that $k \ll c$. Note that directly calculating matrix inversion has complexity $O\left(c^3\right)$ which is unaffordable when c is large.

Partial learning on temporal features. We adopt partial learning to accurately learn multiple components from each temporal feature T. Specifically, we first duplicate T into two virtual features T_{tr} and T_{se}, and then apply smoother $\boldsymbol{M}_T$ (de-trend operation) on T_{tr} and $\boldsymbol{M}_S$ (de-seasonal operation) on T_{se} iteratively until a partial convergence, and then move out to other features (Fig. 8(a)). In contrast, total learning (i.e., standard approach) puts T_{tr} and T_{se} with numerical features together and conducts backfitting (Fig. 8(b)) without partial convergence, which could lead to undesirable entanglement: the learned trend component exhibits small periodicity (i.e., carries partial seasonal component), and the learned seasonal component exhibits slow drift (i.e., carries partial trend component). In comparison, partial learning learns more accurate trends and seasonal components.

The details about learning multiple components (i.e., seasonal trend decomposition) have been shown in Fig. 9: we conduct local iteration to identify trend $\boldsymbol{f}_T$ and seasonality $\boldsymbol{f}_S$ from temporal feature T as depicted on the right side of Fig. 9. $\boldsymbol{f}_T$ is obtained by applying smoothing matrix $\boldsymbol{M}_T$ (line 3) and cycle-subseries smoothing (line 6) is applied to smoothing matrix $\boldsymbol{M}_{S_\varphi}$ to obtain $\boldsymbol{f}_{S_\varphi}$. Such local iteration ensures effective decomposition of $\boldsymbol{f}_T$ and $\boldsymbol{f}_S$, leading to more stable and accurate results. According to Theorem 2, such local iteration still preserves overall convergence.

Theorem 2. *TSI converges to a solution of FXAM's normal equations.*

Proof. Here is the sketch proof, more details are at Appendix A. A full round of TSI can be denoted as a linear map $\boldsymbol{\mathcal{K}} = \left(\boldsymbol{\Phi}_S\boldsymbol{\Phi}_T\right)^\infty \boldsymbol{\Phi}_Z\boldsymbol{K}^\infty$, where $\boldsymbol{K}^\infty = \left(\boldsymbol{\Phi}_p\boldsymbol{\Phi}_{p-1}\dots\boldsymbol{\Phi}_1\right)^\infty$ and $\boldsymbol{K}^\infty$ is one round of partial learning over numerical features. Here $\boldsymbol{K}^\infty$ represents partial convergence of

	Z_1	Z_2	...	Z_q
Homogeneous set	$dom(Z_1)$	$dom(Z_2)$	...	$dom(Z_q)$
Parameters	$\beta_1, \ldots, \beta_{\|Z_1\|}$	$\beta_{\|Z_1\|+1}, \ldots, \beta_{\|Z_1\|+\|Z_2\|}$	...	$\beta_{c-\|Z_q\|+1}, \ldots, \beta_c$

Fig. 7. Joint learning vs. per-feature learning (Joint learning: all parameters $\beta_1, \ldots, \beta_c$ are jointly updated in each iteration. Per-feature learning: only parameters of Z_i are updated in each iteration).

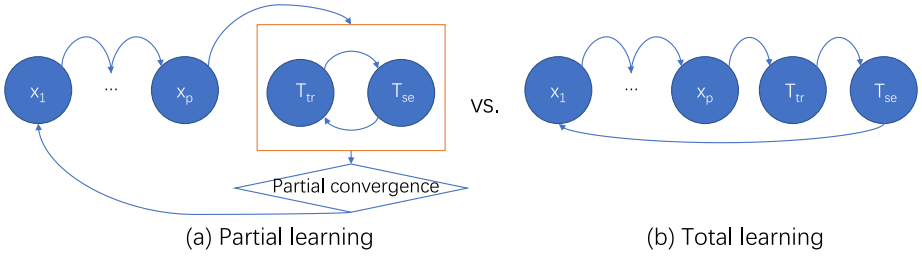

Fig. 8. Partial learning vs. total learning.

Nesterov's gradient acceleration with Power iteration

Solving $\boldsymbol{\beta} = (\boldsymbol{Z}^T\boldsymbol{Z} + \lambda_Z \boldsymbol{I})^{-1}\boldsymbol{Z}^T\boldsymbol{y}_Z$

Initialize:

$\lambda_0 = 0$

$\lambda_S = \left(1 + \sqrt{1 + 4\lambda_{S-1}^2}\right)/2$

$\gamma_S = (1 - \lambda_S)/\lambda_{S+1}$

$\boldsymbol{\beta}^0 = \boldsymbol{0}$

1: $\boldsymbol{A} = \boldsymbol{Z}^T\boldsymbol{Z} + \lambda_Z \boldsymbol{I}$

/* find the greatest eigenvalue of A by using power iteration */

2: μ = greatest eigenvalue of $\boldsymbol{A}$

3: **Cycle**

/* Nesterov's gradient acceleration */

4: $\boldsymbol{b}^{S+1} = \boldsymbol{\beta}^S - (\boldsymbol{A}\boldsymbol{\beta}^S - \boldsymbol{Z}^T\boldsymbol{y}_Z)/\mu$

5: $\boldsymbol{\beta}^{S+1} = (1 - \gamma_S)\boldsymbol{b}^{S+1} + \gamma_S \boldsymbol{b}^S$

6: **Until** $\boldsymbol{\beta}^{S+1}$ does not change

Seasonal-Trend Decomposition Procedure

Input $\{(t^{(l)}, {y_{TS}}^{(l)}) | l = 1, \ldots, N\}$

Initialize: $\boldsymbol{f}_T \leftarrow \boldsymbol{0}, \boldsymbol{f}_S \leftarrow \boldsymbol{0}$

1: **Cycle**

/*de-seasonalizing*/

2: $\boldsymbol{y}_T \leftarrow \boldsymbol{y}_{TS} - \boldsymbol{f}_S$

/*trending*/

3: $\boldsymbol{f}_T \leftarrow smoothing\ over\ \{(t^{(l)}, {y_{TS}}^{(l)}) | l = 1, \ldots, N\}$

/*de-trending*/

4: $\boldsymbol{y}_S \leftarrow \boldsymbol{y}_{TS} - \boldsymbol{f}_T$

5: **for** $\varphi \leftarrow 0$ to $d - 1$

/*cycle-subseries smoothing*/

6: $\boldsymbol{f}_{S_\varphi} \leftarrow smoothing\ over\ \{(t_\varphi, y_{S_\varphi}) | t_\varphi \in \mathcal{T}_\varphi\}$

7: **end for**

8: $\boldsymbol{f}_S \leftarrow \boldsymbol{f}_{S_0} \oplus \ldots \oplus \boldsymbol{f}_{S_{d-1}}$

9: **Until** $\boldsymbol{f}_T, \boldsymbol{f}_S$ do not change

Fig. 9. Details of training in TSI (Left part: Nesterov's acceleration. Right part: seasonal-trend decomposition).

numerical features (Stage1), and $(\boldsymbol{\Phi}_S\boldsymbol{\Phi}_T)^\infty$ represents partial convergence of one temporal feature t (Stage3). It is easy to verify that $\boldsymbol{K}^\infty$ is no longer a symmetric matrix thus we cannot directly utilize the GAM's theorem of convergence. We have to take a step back and try to analyze the basic properties of $\boldsymbol{\mathcal{K}}$. To show $\boldsymbol{\mathcal{K}}^\infty$ exits, we need to check the seminorm descent principle (Buja et al., 1989). Denote the loss function of homogeneous equations as $\mathcal{L}_0(\boldsymbol{f}) := \left\|\sum_{j\in I} \boldsymbol{f}_j\right\|^2 + \sum_{j\in I} \boldsymbol{f}_j^T \left(\boldsymbol{M}_j^- - \boldsymbol{I}\right) \boldsymbol{f}_j$. We prove that we have a linear mapping $\boldsymbol{\mathcal{K}}$ satisfying $\mathcal{L}_0(\boldsymbol{\mathcal{K}}\boldsymbol{f}) < \mathcal{L}_0(\boldsymbol{f})$ when $\mathcal{L}_0(\boldsymbol{f}) > 0$ and $\boldsymbol{\mathcal{K}}\boldsymbol{f} = \boldsymbol{f}$ when $\mathcal{L}_0(\boldsymbol{f}) = 0$. According to theorem 8 of Buja et al. (1989), $\boldsymbol{\mathcal{K}}^m$ converges to $\boldsymbol{\mathcal{K}}^\infty$. ■

4.5. Improving training efficiency

To further improve training speed, we propose two techniques: intelligent sampling and dynamic feature iteration to improve backfitting efficiency of Stage 1 in TSI. Last but not least, we adopt a recent fast version of kernel smoothing instead of standard cubic spline smoothing.

Intelligent Sampling. For large-scale data sets (e.g., N > 100,000), we estimate the shape function over a sample set as better initialization of $\boldsymbol{f}_i$ (Initialize part in Fig. 6). By doing so, the number of iterations towards convergence could be reduced while the time cost of sampling-based initialization could be negligible if the sample size is chosen appropriately. Next, we illustrate how to determine an appropriate sample size n.

Considering the task of smoothing over $\{(x^{(j)}, y^{(j)}) | j = 1, \ldots, N\}$. Assume the records are drawn from ground-truth function $F(X)$: $y^{(i)} = F(x^{(i)}) + \epsilon^{(i)}$ where $\epsilon^{(i)}$ are i.i.d. random errors with $E\left(\epsilon^{(i)}\right) = 0, \mathrm{Var}\left(\epsilon^{(i)}\right) \leq \sigma^2$. Denote U as maximum slope of F, i.e., $\left|F\left(x^{(i)}\right) - F\left(x^{(j)}\right)\right| \leq U\left\|x^{(i)} - x^{(j)}\right\|, \forall x^{(i)}, x^{(j)}$ (Lipschitz condition). Denote f_N and f_n as shape functions obtained by smoothing over $\{(x^{(j)}, y^{(j)}) | j = 1, \ldots, data\ set\ size\}$ on full set and sample set respectively, we verify that the sample variation $E\left\|f_N - f_n\right\|^2$ has an upper bound according to Lemma 1. Therefore the sampling error is controllable and the estimates are accurate.

Lemma 1. $E\left\|f_N - f_n\right\|^2 \leq 4c\left[\left(\sigma^2 + \sup F^2\right) U/n\right]^{2/3}$

Proof. Here is the sketch proof, more details are at Appendix A. Sample variation is the difference between a smoothing function $f_N(X)$ obtained from all records and another smoothing function $f_n(X)$ obtained from sampled records with sample size n. We can easily see that as n approaching N, the difference vanishes. We thus prove that the difference is bounded according to the characterstics of the smooth function and sample size n. ■

According to Lemma 1, sample variation depends on sample size n, noise level σ, maximum slope U and square of maximum absolute value

$\sup F^2$ of F. To bound sample variation for all features, sample size n_i for feature X_i should be $n_i \propto \left(\sigma_i^2 + \sup F_i^2\right) U_i$. We use a pre-specified small sample size n_0 (e.g., 10,000) to approximately estimate $F_i \approx f_{n_0}$ for feature X_i, and then use it to further obtain estimation of σ_i and U_i.

We define $n^* = \max_i \gamma \left(\sigma_i^2 + \sup F_i^2\right) U_i$ as sample size applied to initialization for all numerical features, which is conservative since under sample size n^*, sample variations for all features are bounded. γ is a hyperparameter.

Dynamic Feature Iteration (DFI). DFI is a heuristic algorithm. We propose DFI to dynamically adjust the order of features for smoothing. Smoothing over a feature with higher predictive power will reduce more loss locally (i.e., within the current cycle) thus achieving faster convergence. We propose a lightweight estimator to calculate and update the predictive power of each feature and use it to dynamically order features.

Definition 1. The predictive power of X_i is defined as $Power_i = 2TSS \cdot r_i^2/(N-2) - \left(2\hat{U}_i Bh\right)^2$

Here $TSS = \sum_{l=1}^{N} \left(\tilde{y}^{(l)} - \bar{\tilde{y}}\right)^2$, $\tilde{y}^{(l)}$ is current partial residual and $\bar{\tilde{y}}$ is its average. Assume $\tilde{y}^{(l)}$ is the lth instance of variable $\tilde{Y}$, thus r_i is the Pearson correlation coefficient of X_i and $\tilde{Y}$. B is the bounded support of kernel K_h, and h is corresponding smoothing bandwidth. $\hat{U}_i$ is the estimated maximum slope. In each full cycle over features $X_1, \ldots, X_p$, we estimate $Power_i$ of X_i and use it to sort features by descending order.

Fast Kernel Smoothing Approximation. Kernel smoothing is a popular alternative but suffers from low efficiency due to $O\left(N^2\right)$ complexity in general. However, a fast kernel smoothing method is proposed (Langrené & Warin, 2019), which achieves $O(N)$ complexity and with much smaller coefficient (i.e., $\ll 35$). The key idea is called fast-sum-updating: given a polynomial kernel, this method pre-computes the cumulative sum of each item in the polynomial form on evaluation points and uses these cumulative sums to perform one-shot scanning over evaluation points to complete the task. Our smoothing task takes N input samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(N)}, y^{(N)}\right)$ where $x^{(1)}, \ldots, x^{(N)}$ are also evaluation points (here we strip the feature index i for simplicity). Natural cubic spline smoothing has $O(N)$ time complexity which is still expensive since it takes about 35N operations (Silverman, 1985). We choose Epanechnikov kernel: $K(X) = 3\left(1 - X^2\right)/4$, which is a degree-2 polynomial kernel with good theoretical property. We adopt the fast-sum-updating algorithm to approximate original cubic spline smoothing to reduce operations to $\approx$ 4N almost without loss of accuracy.

4.6. Flexibility

TSI's modularity. As depicted in Fig. 6, each stage in TSI takes partial residuals as input and estimates separate parameters (i.e., corresponds to numerical, categorical, and temporal features respectively) until partial convergence (stage-level). Therefore, each stage can be performed as a standalone module, and TSI can be viewed as a framework to learn these modules iteratively. Such modularization allows us to adopt optimization techniques within each module. As aforementioned, we propose intelligence sampling and DFI to improve training efficiency in Stage1.

Learning more time series components. TSI's modularity allows us to view Stage3 (learning on temporal features) as a classical seasonal-trend decomposition task (as depicted on the right side of Fig. 9), thus more sophisticated approaches can be adopted from literature such as STL (Cleveland et al., 1990) or RobustSTL (Wen et al., 2019). Moreover, Stage3 can be extended to learn additional components such as multiple seasonal components (Cleveland et al., 1990) or aperiodic cyclic components (Hyndman, 2011).

Tolerance to missing data. FXAM is tolerant w.r.t. missing data in temporal features. We partition the instances into phase-φ sets, conduct smoothing within each phase-φ set, and learn sub-component $\boldsymbol{f}_{S_\varphi}$. These sub-components $\boldsymbol{f}_{S_\varphi}$ are further domain-merged to obtain the seasonal component $\boldsymbol{f}_S$. It is known that smoothing is good at interpolation thus it is tolerant of missing data issues.

Extension to multiple temporal features. The previous illustration presupposes a single temporal feature T (for simplicity). When there are multiple temporal features $T_1, \ldots, T_u$ where $u > 1$ as shown in Fig. 10, TSI can be extended naturally by applying Stage3 (line 9–11) in Fig. 6 for each temporal feature provided that the period of its seasonal component is given.

5. Evaluation

We evaluate FXAM on both synthetic and real data sets. We generate synthetic data sets to comprehensively evaluate FXAM's performance against varied data scales and data characteristics, and we use 13 representative real data sets and a case study to demonstrate the effectiveness of FXAM.

Comparison algorithms. We choose 3 representative algorithms for comparison: pyGAM (Servén & Brummitt, 2018), EBM (Nori, Jenkins, Koch, & Caruana, 2019), and XGBoost (Chen et al., 2015). pyGAM is a standard implementation of GAM in python and EBM is the implementation of GA2M. We choose the opaque model XGBoost as a reference for accuracy. The detailed API calls are shown in Appendix F.

Hardware. All experiments are conducted on a Windows Server 2012 machine with 2.8 GHz Intel Xeon CPU E5-2680 v2 and 256 GB RAM. FXAM is implemented by C#. We use the latest version of pyGAM, EBM, and XGBoost in python.

Design and metric. We design experiments to evaluate:

- *Modeling*: FXAM's effectiveness in addressing one-to-many and many-to-one phenomena by varying scales of categorical or temporal features.
- *Training*: The performance of TSI procedure by comparing with pyGAM.
- *Efficiency*: FXAM vs. all competitors on training speed.

To measure the training time of ML model used in predictive analytics, we fix the hyperparameters of each competitor algorithm beforehand. These hyperparameters are carefully tuned to achieve the best performance (details are shown in Appendix F). For each data set, we conduct 5-fold cross-validation and use the average root-mean-square error (RMSE) to measure accuracy and we record average training time.

5.1. Evaluation on synthetic data

Synthetic data generation. To thoroughly evaluate FXAM's performance against varied data scales/characteristics, we synthetically generate data sets by specifying a configuration that is composed of seven factors as shown in Table 2. Factors $1-3$ specify the data scale, factors $4-6$ specify data characteristics and factor 7 specifies the difficulty level of ground-truth generation functions. The generation functions in easy mode follow standard additive models, i.e., response is the sum of the contribution of each feature and then with a small random noise. The hard mode considers a significant portion of feature interactions with higher noise level (details are in Appendix B).

Results. We have conducted evaluations by varying #records, #features, and so on. In each setup, we fix the other factors and only vary a specific one (details are in Appendix C). Fig. 11 depicts results on hard data sets. The results of training time are represented by logarithmic scale to facilitate clear comparison on the same graph, as the FXAM model's training time is orders of magnitude smaller than that of other models. As the number of features gradually increases, the proportion of categorical variables increases, or the seasonal ratio in temporal variables increases, the accuracy of XGBoost begins to decline, and

Extension to Multiple Temporal Features

/* Stage3: learning trend and seasonality on multiple temporal features */

1: $k = 1,2, \dots u$

2: $\boldsymbol{y}_{T_k S_k} = \boldsymbol{y} - \sum \boldsymbol{f}_i - \boldsymbol{f}_Z - \sum_{j \neq k} \boldsymbol{f}_{T_j} - \sum_{j \neq k} \boldsymbol{f}_{S_j}$

3: $\left\{\left(f_{T_k}(t_k^{(l)}), f_{S_{k,\varphi}}(t_k^{(l)})\right) | l = 1, \dots, N\right\} = seasonal\ trend\ decomposition\ of\ \left\{\left(t_k^{(l)}, y_{T_k S_k}^{(l)}\right) | l = 1, \dots, N\right\}$

4: $\boldsymbol{f}_{S_k} = \sum_{\varphi} \boldsymbol{f}_{S_{k,\varphi}}$

5: **Until** $\boldsymbol{f}_Z, \boldsymbol{f}_i, \boldsymbol{f}_{T_k}, \boldsymbol{f}_{S_k}$ does not change

Fig. 10. Extension to multiple temporal features in Stage3.

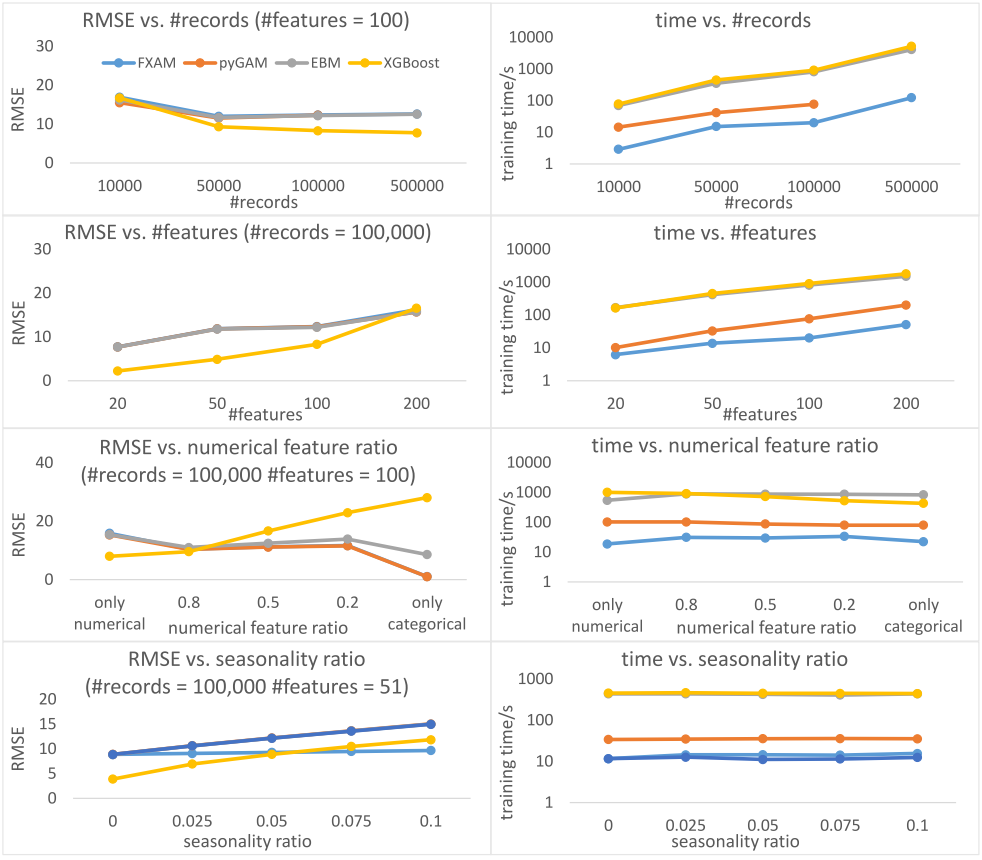

Fig. 11. Evaluation on synthetic data sets. Four rows reflect different data scales/characteristics by varying: Row1: #records | Row2: #features | Row3: numerical feature ratio | Row4: seasonality ratio of the temporal feature. The 51 features indicated in the left figure on the third Row are composed of 50 numerical features and 1 temporal feature.

Table 2
Seven factors for generating synthetic data sets.

ID	Factor	Value range
1	#records	[10, 000, 500, 000]
2	#features	[20, 200]
3	#total cardinalities	[0, 2000]
4	numerical feature ratio	[0, 1]
5	has temporal feature	{yes, no}
6	seasonality ratio	[0.0, 0.1]
7	difficulty	{easy, hard}

FXAM gradually outperforms XGBoost in accuracy. Here only shows the results on 'hard' data sets. For 'easy' data sets, since the ground-truth generating mechanism is with feature contributions fully untangled, GAM-related approaches could achieve optimal accuracy, this is why XGBoost does not perform well on 'easy' data sets. The complete results are shown in Appendix E.

Addressing One-To-Many over temporal features. The 4th row of Fig. 11 illustrates the effectiveness of FXAM on learning trends and seasonal components over temporal features. The seasonality ratio (defined as Fraction-of-Variance-Explained: (Achen, 1990)) is varied from 0% to 10%. We also compare with a simplified version of FXAM called "FXAM_no_TAS", i.e., treating the temporal feature as a normal numerical feature. As the seasonality ratio increases from 0% to 10%, the RMSEs of all the algorithms increase except FXAM's RMSE which remains stable and achieves the highest accuracy.

Addressing Many-To-One over categorical features. In the first chart of 3rd row of Fig. 11, the right-most data points show RMSEs when all features are categorical with total cardinality = 2000. Both FXAM and pyGAM achieve the smallest RMSE since they have the same regularization on categorical features. FXAM's training speed is 3 times faster than pyGAM and 10 times faster than EBM/XGBoost due to its joint learning strategy.

Efficiency and Convergence of TSI. Results in Fig. 11 generally show the performance of FXAM's training procedure TSI. In the 1st column, XGBoost achieves the best accuracy. Meanwhile, FXAM achieves close or even better accuracy vs. pyGAM or EBM. ALL results in the 2nd column show that FXAM achieves magnitude-order speed-up.

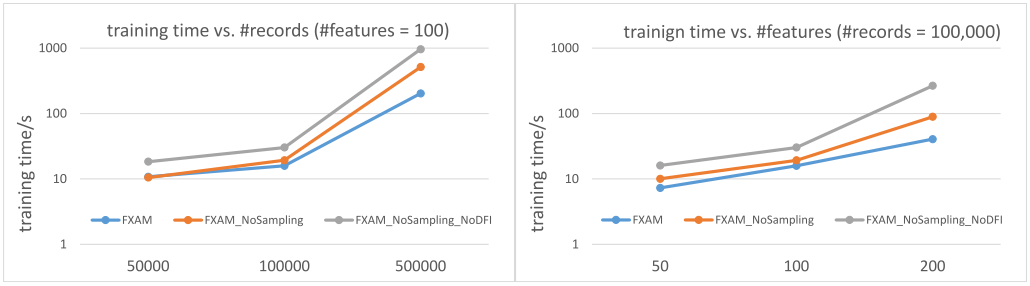

Fig. 12. Ablation study of FXAM.

Fig. 13. Training time (logarithmic scale in the *y*-axis of the top figure) and RMSE (bottom figure) on 13 real data sets.

Feasibility for interactive ML (iML). The typical scale of multi-dimensional data is with #records = 100,000, and #features = 100. FXAM uses less than 20 s for training on such scale data set whereas the other algorithms cost more than 100 s (pyGAM) or even 1000 s (EBM or XGBoost). pyGAM throws Out-Of-Memory exception when training on data sets with 500,000 records. To facilitate smooth iML experience, the machine is asked to respond within 10 s. Therefore, FXAM is feasible to facilitate iML in an interactive and iterative manner.

Ablation Study. We also conduct an ablation study to evaluate the efficiency improvement by using two novel techniques proposed in FXAM (hyperparameters are shown in Appendix D): intelligent sampling and DFI (Dynamic Feature Iteration). As shown in Fig. 12, we compare the efficiency of FXAM among disabling sampling (FXAM_NoSampling, orange curve), both sampling and dynamic feature iteration disabled (FXAM_NoSampling_NoFDI, grey curve) and original model (FXAM, blue curve). All data sets are with difficulty level = 'hard'. The results show that sampling and DFI improve efficiency significantly, which confirms that sampling indeed identifies better initialization of smoothing functions, and dynamic feature iteration increases convergent speed. All three algorithms have the same RMSE because we observed that the two techniques involved, intelligent sampling and the DFI algorithm, primarily accelerate the convergence speed without affecting the predictive accuracy. More specifically, intelligent sampling enhances the initialization phase by estimating an initial function based on sampling. This significantly accelerates the rate of convergence to the global optimum. Similarly, the DFI algorithm identifies features with higher predictive power. FXAM first performs smoothing operations on these features, which also enables a faster convergence to the global optimum. Theorems 1 and 2 further support this by demonstrating that FXAM can converge to the global optimum. Thus, while these techniques improve the speed of convergence, they do not alter the ultimate accuracy of the prediction as the algorithms are still converging to the same global optimum.

5.2. Evaluation on real data

We have collected 13 representative real data sets from diverse domains as shown in Table 3.

Results. As depicted in Fig. 13, compared with pyGAM and EBM, FXAM achieves the best accuracy on 6 of 9 data sets that have temporal features compared with pyGAM and EBM. For the left 3 data sets 'BikeShare', 'Kickstarter', and 'CSAT', FXAM's accuracy is still very competitive. Such result reflects the effectiveness of FXAM in decomposing trend and seasonal components (FXAM exhibits even better accuracy than XGBoost on three data sets 'Clif', 'SH', and 'GZ'). Regarding training speed, FXAM is significantly faster than the other algorithms. FXAM uses < 10 seconds to finish training on 12 out of 13 data sets except for the ultra-large data set 'Birth', where FXAM uses ≈ 200 seconds but XGBoost and EBM use nearly one hour and pyGAM runs into Out-Of-Memory exception. Therefore, FXAM is suitable to facilitate iML.

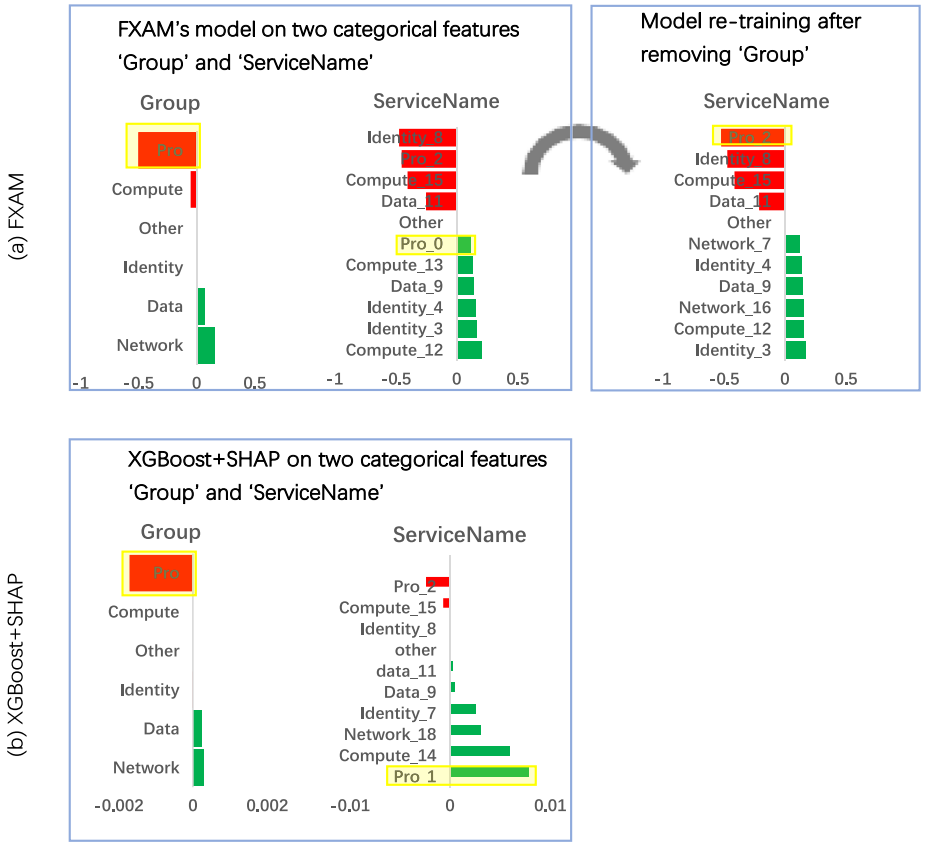

Fig. 14. Model interpretability comparison on categorical features (Combined with domain knowledge, the results show that FXAM accurately detects the positive contribution (green) of *Pro*, while XGBoost+SHAP mistakenly identifies the contribution of *Pro* as negative (red).

Table 3
Details of real data sets.

Name	#Records	#Features	Has temporal	Domain
BikeShare[a]	17,414	8	Y	Social
SY[a]	20,451	10	Y	Environment
Clif[a]	7,484	25	Y	Sales
CD[a]	27,366	10	Y	Environment
Kickstarter[a]	55,427	6	Y	Financial
SH[a]	34,040	8	Y	Environment
CSAT[b]	31,779	11	Y	Computer
Energy[c]	19,735	28	Y	Energy
GZ[a]	32,353	10	Y	Environment
IT[b]	341,721	6	N	Computer
Asteroid[a]	137,680	26	N	Astronomy
Autos[a]	304,133	11	N	Sales
Birth[a]	1,000,000	46	N	Healthcare

[a] This data set is from Kaggle https://www.kaggle.com/.
[b] This data set is from Microsoft Research https://www.microsoft.com/en-us/research/.
[c] This data set is from Archive https://archive.org/.

Real-world case: predicting customer satisfaction compared with XGBoost+SHAP. 'CSAT' is about customer satisfaction of using an online service system. Each record corresponds to specific customer feedback, with a satisfaction score ranging from 1 to 5. Engineers initially collect 20 features to predict the score. We build an Excel add-in to facilitate user interaction with FXAM. Engineers want to know: *(1) whether these features are effective to support expert decision-making; (2) how to support expert decision-making? For example, whether it is necessary to add more resources for customer service on weekends.* We show how FXAM facilitates answering these questions.

In contrast, we use SHAP (Lundberg et al., 2020) to explain the prediction results of XGBoost, and the average of SHAP values of the feature is calculated as the total contribution of the feature. It should be noted that in general, each feature has a positive or negative SHAP value on each instance, and in this experiment, the SHAP values of the feature remain positive or negative on the vast majority of instances, so there is no offset between the positive and negative contributions when calculating the average.

FXAM is trustworthy enough to support expert decisions. The contributions of features output by a trustworthy model should be consistent with the ground truth. The following takes *Pro* as an example to illustrate the trustworthiness of FXAM. The horizontal axis in Fig. 14 represents the contribution of each feature to the results. Red represents negative contribution, while green represents positive contribution. Fig. 14(a) depicts the FXAM model on categorical features *Group* and *ServiceName*. *ServiceName* is the subdivision of *Group* to indicate specific services within each *Group* (each service name takes its corresponding group name as a prefix). FXAM reports a positive contribution from *Pro*_0 (box selected) on *SeriviceName*. However, due to domain knowledge, services within *Pro* are more likely to have negative contributions. A feasible explanation is that *Pro* from feature *Group* absorbs the most negative contribution (1 st column in Fig. 14(a)). Considering the dependency between *Group* and *ServiceName*, engineers delete *Group* and then re-train FXAM. In seconds, FXAM returns an updated model which is verified to be meaningful (3rd column of Fig. 14(a)). Following such interactive analysis flow, engineers eventually select 11 features to predict the score, and the corresponding model is verified to be trustworthy.

As shown in Fig. 14(b), when explaining XGBoost with SHAP, *Pro* shows the same contradictory results on *Group* and *Servicename*. But it also shows that *Pro* tends to have a positive impact, which contradicts our domain knowledge.

More resources of customer service should be added on weekends, based on FXAM's interpretability on Date. In Fig. 15(a), FXAM identifies a

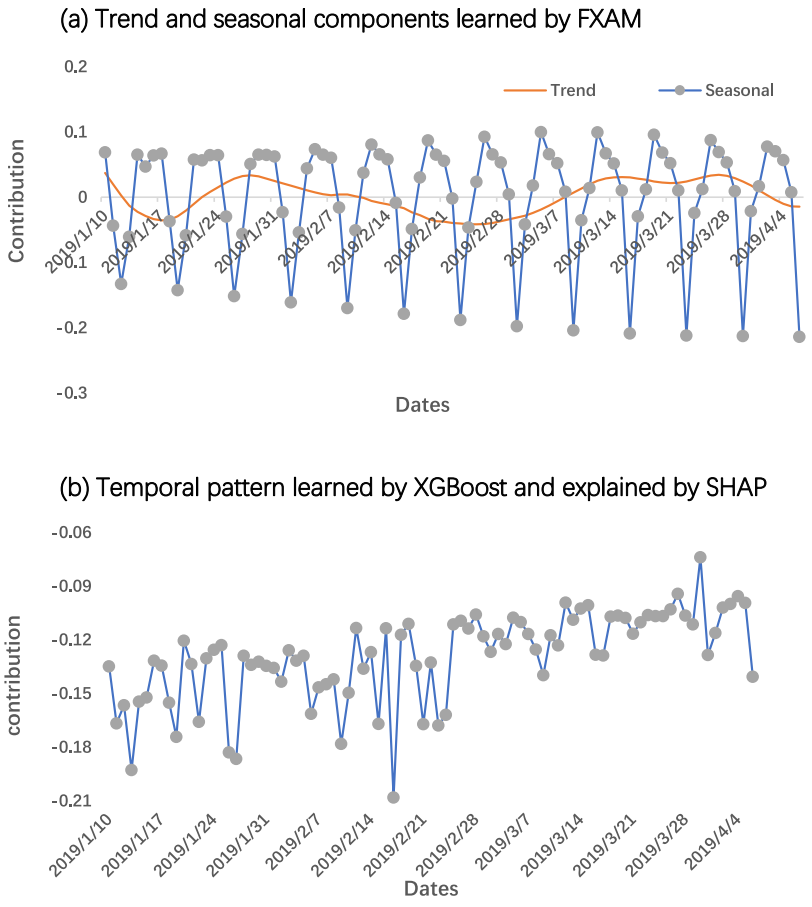

Fig. 15. Model interpretability comparison on feature *Date* (FXAM can disentangle the seasonality and trend contributions of temporal features, while they are entangled in XGBoost+SHAP 's results).

significant seasonal component f_S (blue curve) and a trend component f_T (orange curve) on feature *Date*. In f_S, a weekly repeating pattern is exhibited and each Saturday (i.e., valley points) has the most negative contribution within a period. There is no clear upward or downward trend in f_T. Thus, stable user behavior is exhibited such that customer satisfaction becomes worse during the weekend. Moreover, the amplitude of f_S is growing, which suggests the urgency to allocate more customer service on the weekend.

The results of XGBoost+SHAP are shown in Fig. 15(b). The trend and seasonality are mixed together, which cannot be effectively analyzed and cannot help answer the second question.

6. Conclusion

We have proposed FXAM, which extends GAM's modeling capability with a unified additive model for numerical, categorical, and temporal features. FXAM addresses the challenges introduced by one-to-many and many-to-one phenomena, which are commonly appeared in predictive analytics. FXAM conducts a novel training procedure called TSI (Three-Stage Iteration). We prove that TSI is guaranteed to converge and the solution is globally optimal. We further propose two novel techniques to speed up FXAM's training algorithm to meet the needs of interactive ML. Evaluations have verified that FXAM remarkably outperforms the existing GAMs regarding training speed and modeling categorical or temporal features.

The success of FXAM's real-world adoption has demonstrated the importance of interactive and interpretable ML, which are also the main design goals of FXAM. One future direction is to extend FXAM to support not only main effects (i.e., the univariate shape functions in this paper) but also pairwise or higher-order interactions such as pureGAM (Sun et al., 2022) and GA2M (Lou et al., 2013). It is known that learning interactions will introduce significantly higher computational costs, and the high-efficiency training speed of FXAM is clearly a benefit.

CRediT authorship contribution statement

Yuanyuan Jiang: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Rui Ding:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Tianchi Qiao:** Conceptualization, Methodology, Software, Validation, Investigation. **Yunan Zhu:** Conceptualization, Methodology, Software, Validation, Investigation. **Shi Han:** Conceptualization, Writing – review & editing. **Dongmei Zhang:** Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A. Proof details

A.1. Proof of Theorem 1

Proof. According to theorem 2 in Buja et al. (1989), the solutions of normal equations exist and are globally optimal if each smoothing matrix $\boldsymbol{M}_i, \boldsymbol{M}_Z, \boldsymbol{M}_T$, or $\boldsymbol{M}_S$ is symmetric and shrinking (i.e., with eigenvalues in $[0,1]$). Thus we check $\boldsymbol{M}_i, \boldsymbol{M}_Z, \boldsymbol{M}_T$, and $\boldsymbol{M}_S$ one by one:

$\boldsymbol{M}_i, \boldsymbol{M}_T$ are indeed symmetric and shrinking according to standard analysis of cubic spline smoothing matrix.

Re-write $\boldsymbol{M}_Z = \boldsymbol{Z}\boldsymbol{\Lambda}\boldsymbol{Z}^T$ where $\boldsymbol{\Lambda} = \left(\boldsymbol{Z}^T\boldsymbol{Z} + \lambda_Z \boldsymbol{I}\right)^{-1}$. It is easy to see that $\boldsymbol{\Lambda}$ is a symmetric matrix thus $\boldsymbol{M}_Z^T = \left(\boldsymbol{Z}^T\right)^T \boldsymbol{\Lambda}^T \boldsymbol{Z}^T = \boldsymbol{Z}\boldsymbol{\Lambda}\boldsymbol{Z}^T = \boldsymbol{M}_Z$.

Denote singular value decomposition of $\boldsymbol{Z}$ is $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, $\boldsymbol{\Lambda}$ is a $c \times c$ diagonal matrix, with diagonal entries $\Lambda_{11} \geq \cdots \geq \Lambda_{cc} \geq 0$. Thus we have $\boldsymbol{M}_Z\boldsymbol{y} = \boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{Z} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{y} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T\left(\boldsymbol{V}\boldsymbol{\Lambda}^2\boldsymbol{V}^T + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{U}^T\boldsymbol{y} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T\left(\boldsymbol{V}\boldsymbol{\Lambda}^2\boldsymbol{V}^T + \lambda\boldsymbol{V}\boldsymbol{I}\boldsymbol{V}^T\right)^{-1}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{U}^T\boldsymbol{y} = \boldsymbol{U}\boldsymbol{\Lambda}\left(\boldsymbol{\Lambda}^2 + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{\Lambda}\boldsymbol{U}^T\boldsymbol{y} = \sum_{j=1}^{c} \boldsymbol{u}_j \frac{\Lambda_{jj}^2}{\Lambda_{jj}^2+\lambda}\boldsymbol{u}_j^T\boldsymbol{y}$, thus the eigenvalues $\frac{\Lambda_{jj}^2}{\Lambda_{jj}^2+\lambda}$ are in $[0,1]$ considering $\lambda > 0$.

Re-write $\boldsymbol{M}_S = \boldsymbol{P}^T\boldsymbol{\Theta}\boldsymbol{P}$. Since each $\widetilde{\boldsymbol{K}_{S_\varphi}}$ is a symmetric matrix, thus $\left(\boldsymbol{I} + \lambda_S\widetilde{\boldsymbol{K}_{S_\varphi}}\right)^{-1}$ is symmetric and $\boldsymbol{\Theta}$ is symmetric, thus $\boldsymbol{M}$ is symmetric. Due to the shrinking property of $\left(\boldsymbol{I} + \lambda_S\widetilde{\boldsymbol{K}_{S_\varphi}}\right)^{-1}$, and considering $\boldsymbol{\Theta}$ is a block-diagonal matrix with $\left(\boldsymbol{I} + \lambda_S\widetilde{\boldsymbol{K}_{S_\varphi}}\right)^{-1}$ as its blocks, thus $\boldsymbol{\Theta}$ is also shrinking: $\|\boldsymbol{\Theta}\boldsymbol{y}\|^2 \leq \|\boldsymbol{y}\|^2, \forall \boldsymbol{y}$. So $\left\|\boldsymbol{M}_S\boldsymbol{y}\right\|^2 = \boldsymbol{y}^T\boldsymbol{M}_S^T\boldsymbol{M}_S\boldsymbol{y} = \boldsymbol{y}^T\boldsymbol{P}^T\boldsymbol{\Theta}^T\boldsymbol{\Theta}\boldsymbol{P}\boldsymbol{y} = \|\boldsymbol{\Theta}\boldsymbol{P}\boldsymbol{y}\|^2 \leq \|\boldsymbol{P}\boldsymbol{y}\|^2 = \|\boldsymbol{y}\|^2$, thus $\boldsymbol{M}_S$ is shrinking. ■

A.2. Proof of Theorem 2

Proof. A full cycle of TSI can be written as a linear map $\boldsymbol{\mathcal{K}} = \left(\boldsymbol{\Phi}_S\boldsymbol{\Phi}_T\right)^\infty \boldsymbol{\Phi}_Z\boldsymbol{K}^\infty$, where $\boldsymbol{K}^\infty = \left(\boldsymbol{\Phi}_p\boldsymbol{\Phi}_{p-1}\ldots\boldsymbol{\Phi}_1\right)^\infty$. Here $\boldsymbol{K}^\infty$ represents partial convergence of numerical features (Stage1), and $\left(\boldsymbol{\Phi}_S\boldsymbol{\Phi}_T\right)^\infty$ represents partial convergence of one temporal feature t (Stage3). We need to prove $\boldsymbol{\mathcal{K}}^m$ converges to $\boldsymbol{\mathcal{K}}^\infty$, and $\boldsymbol{\mathcal{K}}^\infty$ is a solution of FXAM's normal equations.

Denote the index set $\mathcal{I} := \{1, 2, \ldots, p, Z, T, S\}$. We only need to prove that TSI converges to a solution of FXAM's homogeneous equations (i.e., FXAM's normal equations with $\boldsymbol{y} = \boldsymbol{0}$) because a general solution is a solution of homogeneous equations plus an arbitrary solution of FXAM's normal equations. Denote the loss function of homogeneous equations as $\mathcal{L}_0(\boldsymbol{f}) := \left\|\sum_{j\in\mathcal{I}} \boldsymbol{f}_j\right\|^2 + \sum_{j\in\mathcal{I}} \boldsymbol{f}_j^T\left(\boldsymbol{M}_j^- - \boldsymbol{I}\right)\boldsymbol{f}_j$.

We define a linear map $\boldsymbol{\Phi}_j$ to describe the updating of jth component in TSI when $\boldsymbol{y} = \boldsymbol{0}$:

$$\boldsymbol{\Phi}_j : \begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_p \\ \boldsymbol{f}_Z \\ \boldsymbol{f}_T \\ \boldsymbol{f}_S \end{bmatrix} \equiv \boldsymbol{f} \to \begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ -\boldsymbol{M}_j \sum_{i\in\mathcal{I}, i\neq j} \boldsymbol{f}_i \\ \boldsymbol{f}_Z \\ \boldsymbol{f}_T \\ \boldsymbol{f}_S \end{bmatrix}, \forall j \in \mathcal{I}$$

A full cycle of backfitting over numerical features is then described by $\boldsymbol{K} = \boldsymbol{\Phi}_p\boldsymbol{\Phi}_{p-1}\ldots\boldsymbol{\Phi}_1$. Denote the m full cycles as $\boldsymbol{K}^m$. It is obvious that $\boldsymbol{K}^m$ converges to a limit $\boldsymbol{K}^\infty$ (we can view this as a standard task of backfitting over pure numerical features) therefore with property $\boldsymbol{K}\boldsymbol{K}^\infty = \boldsymbol{K}^\infty$. Note that $\boldsymbol{K}^\infty$ describes the procedure of stage 1, thus the full cycle of the entire TSI is $\boldsymbol{\mathcal{K}} = \boldsymbol{\Phi}_S\boldsymbol{\Phi}_T\boldsymbol{\Phi}_Z\boldsymbol{K}^\infty$. Since each component of $\boldsymbol{\mathcal{K}}$ is minimizer of $\mathcal{L}_0(\boldsymbol{f})$ and since $\mathcal{L}_0$ is a quadratic form, hence $\mathcal{L}_0(\boldsymbol{\mathcal{K}}\boldsymbol{f}) \leq \mathcal{L}_0(\boldsymbol{f})$. When $\mathcal{L}_0(\boldsymbol{\mathcal{K}}\boldsymbol{f}) = \mathcal{L}_0(\boldsymbol{f})$, no strict descent is possible on any component, thus $\boldsymbol{\Phi}_S\boldsymbol{f} = \boldsymbol{f}, \boldsymbol{\Phi}_T\boldsymbol{f} = \boldsymbol{f}, \boldsymbol{\Phi}_Z\boldsymbol{f} = \boldsymbol{f}, \boldsymbol{K}^\infty\boldsymbol{f} = \boldsymbol{f}$. Considering $\boldsymbol{K}\boldsymbol{K}^\infty = \boldsymbol{K}^\infty$, thus $\boldsymbol{K}\boldsymbol{K}^\infty\boldsymbol{f} = \boldsymbol{K}^\infty\boldsymbol{f} \Leftrightarrow \boldsymbol{K}\boldsymbol{f} = \boldsymbol{f}$ when descent vanishes. Since each component $\boldsymbol{\Phi}_j$ of $\boldsymbol{K}$ only updates separate $\boldsymbol{f}_j$, thus $\boldsymbol{K}\boldsymbol{f} = \boldsymbol{f} \Leftrightarrow \boldsymbol{\Phi}_j\boldsymbol{f} = \boldsymbol{f}, \forall j \in \{1, \ldots, p\}$. So descent vanishes on any $\boldsymbol{f}$ equivalent to $\boldsymbol{\Phi}_j\boldsymbol{f} = \boldsymbol{f}, \forall j \in \mathcal{I}$. Meanwhile, such $\boldsymbol{f}$ satisfies homogeneous equations, which indicates $\mathcal{L}_0(\boldsymbol{f}) = 0$ according to theorem 5 in Buja et al. (1989). In summary, we have a linear mapping $\boldsymbol{\mathcal{K}}$ satisfying $\mathcal{L}_0(\boldsymbol{\mathcal{K}}\boldsymbol{f}) < \mathcal{L}_0(\boldsymbol{f})$ when $\mathcal{L}_0(\boldsymbol{f}) > 0$ and $\boldsymbol{\mathcal{K}}\boldsymbol{f} = \boldsymbol{f}$ when $\mathcal{L}_0(\boldsymbol{f}) = 0$. According to theorem 8 of Buja et al. (1989), $\boldsymbol{\mathcal{K}}^m$ converges to $\boldsymbol{\mathcal{K}}^\infty$. ■

A.3. Proof of Lemma 1

Proof. Sample variation is the difference between a smoothing function $f_N(X)$ obtained from all records and another smoothing function $f_n(X)$ obtained from sampled records with sample size n.

According to theorem 5.2 in Györfi, Kohler, Krzyżak, and Walk (2002), for any kernel smoother f_N, $E\left\|f_N - F\right\|^2 \leq c\left(\frac{(\sigma^2+\sup F^2)U}{N}\right)^{2/3}$, $\forall N$. This provides a way to estimate the upper bound of sample variation by

$$\begin{aligned}
E\left\|f_N - f_n\right\|^2 &= \int \left(f_N(X) - f_n(X)\right)^2 \mu(dX) \\
&= \int \left(f_N(X) - F(X) + F(X) - f_n(X)\right)^2 \mu(dX) \\
&\leq \int \left(f_N(X) - F(x) + F(X) - f_n(X)\right)^2 \mu(dX) \\
&+ \int \left(f_N(X) - F(X) - F(X) + f_n(X)\right)^2 \mu(dX) \\
&= 2\left(E\left\|f_N - F\right\|^2 + E\left\|f_n - F\right\|^2\right) \\
&\leq 4E\left\|f_n - F\right\|^2 \\
&= 4c\left(\frac{\left(\sigma^2 + \sup|F|^2\right)U}{n}\right)^{\frac{2}{3}}
\end{aligned}$$ ■

Appendix B. Generation details of synthetic data

B.1. Generation for numerical features

Easy Mode. The numerical features are generated based on three univariate functions as follows:

1. $f(X) = A_1 X$
2. $g(X) = A_2 X^2 + A_3 X$
3. $h(X) = A_4 \sin\left(A_5 X + A_6\right)$

For a specific numerical feature, we first randomly choose one of the three functions by probabilities: 0.3 : 0.3 : 0.4 (w.r.t. f, g and h respectively). A_1, A_3, A_4 are random variables that are uniformly and independently drawn from $[-2, 2]$. A_2 is drawn uniformly from $[-1, 1]$. A_5 is drawn uniformly from $[0, 6\pi]$, and A_6 is drawn uniformly from $[-0.5, 0.5]$.

Once the coefficients $A_1, \ldots, A_6$ are set, our generator generates each record with variable X drawn uniformly from $[0, 10]$.

The final response is the total sum of each function and additionally with a random noise ϵ : ϵ follows normal distribution with $E(\epsilon) = 0$, and its variance is adjusted based on generated data so that the $\mathrm{Var}(\epsilon)/TSS = 0.1\%$.

Hard Mode. Besides the three univariate functions, we include two additional two-variable functions I_1 and I_2 to indicate feature interactions:

- $I_1(X_1, X_2) = B_1 X_1 X_2 + B_2 X_1 + B_3 X_2$
- $I_2(X_1, X_2) = B_4 \cos\left(B_5 X_1 X_2 + B_6 X_1 + B_7 X_2 + B_8\right)$

In hard mode, for a specific numerical feature, we first randomly choose one of the five functions $\{f, g, h, I_1, I_2\}$ by probabilities: 0.1 : 0.1 : 0.2 : 0.2 : 0.4 accordingly. If the function is drawn to be either I_1 or I_2, we will use two numerical features to generate their contributions to the response.

Coefficients B_1, B_2, B_3, B_4 are random variables that are uniformly and independently drawn from $[-2, 2]$. B_5, B_6, B_7 are drawn uniformly from $[0, 4\pi]$, and B_8 is drawn uniformly from $[-0.5, 0.5]$

Once the coefficients are set, if the function is either I_1 or I_2, the two variables are drawn uniformly and independently $X_1 \sim [0, 10], X_2 \sim [0, 10]$ to generate each record.

The final response is the total sum of each function and additionally with a random noise ϵ. We adjust the interaction items to assure they contribute 60%–70% to final response (w.r.t. Fraction of Variance Explained by interaction items). ϵ follows normal distribution with $E(\epsilon) = 0$, and its variance is adjusted based on generated data so that the $\mathrm{Var}(\epsilon)/TSS = 0.5\%$. Thus, the noise level for hard mode is five times larger than it for easy mode

B.2. Generation for categorical features

For each categorical feature, its cardinality is set uniformly from integers in [2, $MaxCardinality$]. $MaxCardinality$ is a configuration parameter with value ranging from 10 to 38 (so the average cardinality is from 6 to 20).

The contribution of each specific categorical value Z_i is $\beta(Z_i)$, called weight, and $\beta(Z_i) \sim [0, 15]$ which is drawn independently and uniformly.

B.3. Generation for temporal feature

We inject seasonality components into data by considering a temporal feature with the form:

$$f_{TS}(T) = V_1 \sin\left(\frac{2\pi T}{10} + V_2\right)$$

Here $V_2 \sim [-5, 5]$ and V_1 is used to control the ratio of seasonality components (w.r.t. its influence on final response).

T is the discrete time, so it is an integer randomly drawn from $[1, 200]$.

Appendix C. Configurations for evaluation on synthetic data

C.1. Varying # records

Here we set $MaxCardinality = 10$ per each categorical feature, so the expectation of total cardinality is 120 as shown in Table C.1.

Table C.1
Varying # Records.

ID	Factor	Value range
1	#records	[10,000, 500,000]
2	#features	100
3	#total cardinalities	120
4	Numerical feature ratio	0.8
5	Has temporal feature	No
6	Difficulty	{easy, hard}

Table C.2
Varying # Features.

ID	Factor	Value range
1	#records	100,000
2	#features	[20, 200]
3	#total cardinalities	[24, 240]
4	Numerical feature ratio	0.8
5	Has temporal feature	No
6	Difficulty	easy, hard

Table C.3
Varying numerical feature ratio.

ID	Factor	Value range
1	#records	100,000
2	#features	100
3	#total cardinalities	[0, 2000]
4	Numerical feature ratio	[0, 1]
5	Has temporal feature	No
6	Difficulty	{easy, hard}

Table C.4
Varying seasonality ratio from temporal feature.

ID	Factor	Value range
1	#records	100,000
2	#features	51
3	#total cardinalities	60
4	Numerical feature ratio	40/51
5	Has temporal feature	yes
6	Difficulty	{easy, hard}

C.2. Varying # features

Here we set $MaxCardinality = 10$ per each categorical feature as shown in Table C.2.

C.3. Varying numerical feature ratio

Here we set $MaxCardinality = 38$ per each categorical feature as shown in Table C.3 so that the expectation of total cardinality is

0: when numerical feature ratio = 1;

2000 : when numerical feature ratio = 0

C.4. Varying seasonality ratio from temporal feature

Here we set $MaxCardinality = 10$ per each categorical feature as shown in Table C.4.

Fig. D.1. Performance comparison from different perspectives by varying. Row1: #records | Row2: #features | Row3: numerical feature ratio | Row4: seasonality ratio of temporal feature.

Table D.1
Hyperparameters in study 1.

ID	Factor	Value range
1	#records	{50, 000, 100, 000, 500, 000}
2	#features	100
3	#total cardinalities	0
4	Numerical feature ratio	1
5	Has temporal feature	No
6	Difficulty	Hard

Table D.2
Hyperparameters in study 2.

ID	Factor	Value range
1	#records	100,000
2	#features	{50, 100, 200}
3	#total cardinalities	0
4	Numerical feature ratio	1
5	Has temporal feature	No
6	Difficulty	Hard

Appendix D. Ablation study

Ablation study mainly evaluates two novel techniques of FXAM: intelligent sampling and dynamic feature iteration, and the hyperparameters are shown in Table D.1 and Table D.2 respectively. Since these two techniques are applied for numerical features, thus in this study, we only generate data sets with pure numerical features.

Appendix E. Complete results on synthetic data

In this paper, we only present results on 'hard' data sets. Here the first two columns are additional results on 'easy' data sets as shown in Fig. D.1. FXAM and other related approaches perform much better than XGBoost on accuracy and efficiency for 'easy' data set.

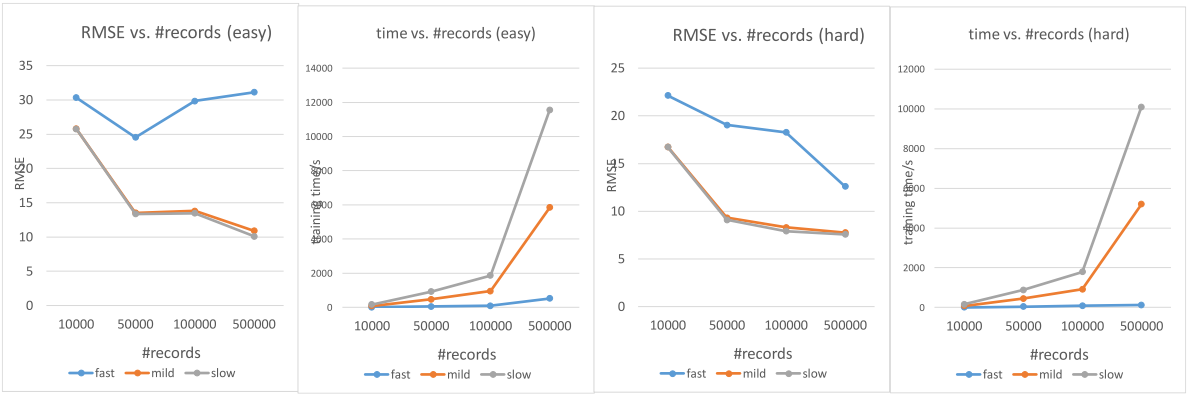

Fig. E.1. Performance results of XGBoost for three different versions.

Appendix F. API calls for XGBoost, EBM, mgcv and pyGAM

To make fair comparison with FXAM, for categorical features, we conduct the same one-hot encoding and apply it to all the competing algorithms.

Below are the detailed API calls that we choose for comparison, the hyperparameters are carefully tuned to make the result accurate and as fast as possible.

F.1. EBM

We use the python code from Nori et al. (2019) for evaluation. The detailed API calling is: ExplainableBoostingRegressor (n_estimators = 16, learning_rate = 0.01, max_tree_splits = 2, (default parameters) n_jobs = 1)

F.2. pyGAM

We use the python code from Servén and Brummitt (2018) for evaluation. The detailed API calling is

/* For pyGAM, we choose to fit categorical feature with smoothing function type:

"f()", i.e. factor term; we choose to fit numerical feature with smoothing function type:

"s()", i.e. spline term.

Therefore, terms is a list pre-generated based on feature type, which is used to indicate which type of smoothing function is selected for the corresponding feature. */

LinearGAM(terms, max_iter = 100, tol = 1e − 4)

F.3. XGBoost

We choose three typical versions of parameters to run XGBoost, which are (1) fast, (2) mild, and (3) slow. The "fast" version is with fast training speed but accuracy is low, and the "slow" version is with good accuracy but training speed is low. "mild" is a set of parameters which we carefully tuned; thus it is a good balance, which is the version used in our evaluation. The results of three typical versions are shown in Fig. E.1.

For instance, below we show our experiments for three versions of parameters to call XGBoost. You can see that "mild" achieves very close accuracy with "slow" but its time is much faster.

Fast. n_estimators = 100, learning rate = 0.3, max_depth = 6, min_child_weight = 1

Mild. n_estimators = 500, learning rate = 0.3, max_depth = 7, min_child weight = 5

Slow. n_estimators = 1000, learning rate = 0.1, max_depth = 7, min_child_weight = 5

References

Abdul, A., Vermeulen, J., Wang, D., Lim, B. Y., & Kankanhalli, M. (2018). Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI conference on human factors in computing systems* (pp. 1–18).

Abdul, A., von der Weth, C., Kankanhalli, M., & Lim, B. Y. (2020). COGAM: Measuring and moderating cognitive load in machine learning model explanations. In *Proceedings of the 2020 CHI conference on human factors in computing systems* (pp. 1–14).

Achen, C. H. (1990). What does "explained variance "explain?: Reply. *Political Analysis*, *2*, 173–184.

Agarwal, R., Frosst, N., Zhang, X., Caruana, R., & Hinton, G. E. (2020). Neural additive models: Interpretable machine learning with neural nets. arXiv preprint arXiv:2004.13912.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., et al. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115.

Buja, A., Hastie, T., & Tibshirani, R. (1989). Linear smoothers and additive models. *The Annals of Statistics*, 453–510.

Calabrese, R., et al. (2012). Estimating bank loans loss given default by generalized additive models. *UCD geary institute discussion paper Series, WP2012/24*.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1721–1730).

Chang, C.-H., Tan, S., Lengerich, B., Goldenberg, A., & Caruana, R. (2021). How interpretable and trustworthy are gams? In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 95–105).

Changqing, C. (2018). Multi-scale Gaussian process experts for dynamic evolution prediction of complex systems. *Expert Systems with Application*, *99*, 25–31.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., et al. (2015). Xgboost: extreme gradient boosting. *1*, (4), (pp. 1–4). R package version 0.4-2.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition. *Journal of Official Statistics*, *6*(1), 3–73.

Du, M., Liu, N., & Hu, X. (2019). Techniques for interpretable machine learning. *Communications of the ACM*, *63*(1), 68–77.

Fails, J. A., & Olsen Jr, D. R. (2003). Interactive machine learning. In *Proceedings of the 8th international conference on intelligent user interfaces* (pp. 39–45).

Finlay, S. (2014). *Predictive analytics, data mining and big data: Myths, misconceptions and methods*. Springer.

Györfi, L., Kohler, M., Krzyżak, A., & Walk, H. (2002). *vol. 1, A distribution-free theory of nonparametric regression*. Springer.

Hastie, T. J., & Tibshirani, R. J. (1990). *vol. 43, Generalized additive models*. CRC Press.

Hohman, F., Head, A., Caruana, R., DeLine, R., & Drucker, S. M. (2019). Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI conference on human factors in computing systems* (pp. 1–13).

Hyndman, R. (2011). Cyclic and seasonal time series. *Hyndsight Blog*.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Jung, J., Concannon, C., Shroff, R., Goel, S., & Goldstein, D. G. (2017). Simple rules for complex decisions. arXiv preprint arXiv:1702.04690.

Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H., & Wortman Vaughan, J. (2020). Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI conference on human factors in computing systems* (pp. 1–14).

Kumar, V., & Ram, M. (2021). *Predictive analytics: modeling and optimization*. CRC Press.

Langrené, N., & Warin, X. (2019). Fast and stable multivariate kernel density estimation by fast sum updating. *Journal of Computational and Graphical Statistics*, *28*(3), 596–608.

Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S. (2021). Explainable ai: A review of machine learning interpretability methods. *Entropy*, *23*(1), 18.

Lombrozo, T. (2006). The structure and function of explanations. *Trends in Cognitive Sciences*, *10*(10), 464–470.

Lou, Y., Caruana, R., Gehrke, J., & Hooker, G. (2013). Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 623–631).

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., et al. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, *2*(1), 2522–5839.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 4768–4777).

Meske, C., Bunde, E., Schneider, J., & Gersch, M. (2022). Explainable artificial intelligence: objectives, stakeholders, and future research opportunities. *Information Systems Management*, *39*(1), 53–63.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, *267*, 1–38.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Nesterov, Y. E. (1983). A method of solving a convex programming problem with convergence rate $O(k^2)$. *vol. 269*, In *Doklady akademii nauk* (pp. 543–547). Russian Academy of Sciences.

Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. arXiv preprint arXiv:1909.09223.

Pierrot, A., & Goude, Y. (2011). Short-term electricity load forecasting with generalized additive models. In *Proceedings of ISAP power*.

Reinsch, C. (1967). Smoothing by spline functions. *Numerische Mathematik*, *10*(2), 177–183.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144).

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215.

Servén, D., & Brummitt, C. (2018). pyGAM: generalized additive models in python. *10*, Zenodo.doi.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *47*(1), 1–21.

Simkute, A., Luger, E., Jones, B., Evans, M., & Jones, R. (2021). Explainability for experts: A design framework for making algorithms supporting expert decisions more explainable. *Journal of Responsible Technology*, *7*, Article 100017.

Simpson, G. (2014). Modelling seasonal data with GAMs. *From the Bottom of the Heap*.

Sun, X., Wang, Z., Ding, R., Han, S., & Zhang, D. (2022). Puregam: Learning an inherently pure additive model. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 1728–1738).

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139–1147). PMLR.

Tan, S., Caruana, R., Hooker, G., Koch, P., & Gordo, A. (2018). Learning global additive explanations for neural nets using model distillation. arXiv preprint arXiv:1801.08640.

Tay, J. K., & Tibshirani, R. (2020). Reluctant generalised additive modelling. *International Statistical Review*, *88*, S205–S224.

Tomić, N., & Božić, S. (2014). A modified geosite assessment model (M-GAM) and its application on the Lazar Canyon area (Serbia). *International Journal of Environmental Research*, *8*(4), 1041–1052.

Wang, Z. J., Kale, A., Nori, H., Stella, P., Nunnally, M., Chau, D. H., et al. (2021). GAM changer: Editing generalized additive models with interactive visualization. arXiv preprint arXiv:2112.03245.

Wen, Q., Gao, J., Song, X., Sun, L., Xu, H., & Zhu, S. (2019). RobustSTL: A robust seasonal-trend decomposition algorithm for long time series. *33*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 5409–5416).

Yang, Z., Zhang, A., & Sudjianto, A. (2021). GAMI-Net: An explainable neural network based on generalized additive models with structured interactions. *Pattern Recognition*, *120*, Article 108192.

Zarnowitz, V., & Ozyildirim, A. (2006). Time series decomposition and measurement of business cycles, trends and growth cycles. *Journal of Monetary Economics*, *53*(7), 1717–1739.

Zhang, X., Tan, S., Koch, P., Lou, Y., Chajewska, U., & Caruana, R. (2019). Axiomatic interpretability for multiclass additive models. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 226–234).

Zimmermann, H.-J. (1987). *vol. 10*, *Fuzzy sets, decision making, and expert systems*. Springer Science & Business Media.