From Black-Box to White-Box: Interpretable Learning with Kernel Machines

Hao Zhang[1(✉)], Shinji Nakadai[1], and Kenji Fukumizu[2]

[1] NEC Corporation, Tokyo, Japan
h-zhang@cw.jp.nec.com
[2] The Institute of Statistical Mathematics, Tokyo, Japan

Abstract. We present a novel approach to interpretable learning with kernel machines. In many real-world learning tasks, kernel machines have been successfully applied. However, a common perception is that they are difficult to interpret by humans due to the inherent black-box nature. This restricts the application of kernel machines in domains where model interpretability is highly required. In this paper, we propose to construct interpretable kernel machines. Specifically, we design a new kernel function based on random Fourier features (RFF) for scalability, and develop a two-phase learning procedure: in the first phase, we explicitly map pairwise features to a high-dimensional space produced by the designed kernel, and learn a dense linear model; in the second phase, we extract an interpretable data representation from the first phase, and learn a sparse linear model. Finally, we evaluate our approach on benchmark datasets, and demonstrate its usefulness in terms of interpretability by visualization.

1 Introduction

Black-box models such as kernel machines and neural networks have proven to be highly accurate in many real-world applications. However, they are usually difficult to interpret by humans due to the inherent black-box nature. On the other hand, white-box models are naturally interpretable. For instance, generalized additive models (GAMs) [1] enable humans to interpret the relation between input features and output values thanks to the simple additive structure.

Here, a fundamental question to ask is: what does "interpretable" precisely mean? In this paper, based on the notion presented in [2], we refer to the term *interpretability* of predictive models as the following two aspects:

- Effects of input features on prediction of output values are understandable.
- Partial dependences of output values on input features are visualizable.

White-box models (e.g. GAMs) are easy to interpret by humans. Unfortunately, they are generally less accurate than black-box models. Accuracy is an important measure to evaluate predictive models. Interpretability, however, is as

P. Perner (Ed.): MLDM 2018, LNAI 10934, pp. 213–227, 2018.
https://doi.org/10.1007/978-3-319-96136-1_18

crucial as accuracy in many application domains such as life science [3], criminal justice [4] and marketing analytics [5]. For instance, when predicting customer churn rates, a marketer might wish to understand how the predictive model works as well, potentially for the purpose of promoting a campaign to avoid customer migration. In such application domains, white-box models are seemingly more favorable because of their relatively simple structures. However, Professor Leo Breiman argued that

> Interpretability is a way of getting information. But a model does not have to be simple to provide reliable information about the relation between predictor and response variable; neither does it have to be a data model [6].

This suggests that it is not necessary to have a preference for white-box models to pursue interpretability; instead, the key point is how to extract interpretable information from a model [7].

Motivated by this philosophy, we propose to interpret black-box models, in particular, kernel machines. Kernel machines such as support vector machines (SVMs) are a family of powerful nonparametric models [8,9]. The key idea is to *implicitly* map the input data to a high-dimensional space, and compute inner products in this space via a kernel function (a.k.a. the "kernel trick"). Owing to this implicit feature map, kernel machines are able to capture high-order interactions among features. However, the different effects of features and their interactions on the predictive model are not understandable, because everything is packed into the kernel function in a nontransparent way. Due to this black-box nature, standard kernel machines are difficult to interpret by humans.

In this paper, our objective is to construct interpretable kernel machines. Although kernel machines provide us with powerful predictive performances, the interpretability counterpart has not been thoroughly studied yet in the literature. The white-box RBF classifier [10] is an attempt to interpret kernel machines. Unfortunately, its storage and computation costs are extremely high because multiple kernel matrices have to be computed.

In contrast, we design a new kernel function based on random Fourier features (RFF) [11] to avoid computing kernel matrices. This kernel function consists of sub-kernels on *pairwise* features to capture nonlinear interactions. The corresponding feature map is explicitly built, so that efficient off-the-shelf linear algorithms can be exploited. With the designed kernel function, we develop a two-phase learning procedure:

- In the first phase, we explicitly map all feature pairs to a space produced by the designed kernel, and learn a dense linear model.
- In the second phase, we extract an interpretable data representation from the first phase, and learn a sparse linear model.

The reason why we focus on pairwise features is that we wish to capture main effects as well as two-way interactions in the extracted data representation. As for interactions, we only keep those involving two features due to the limitations of human perception and computer graphics for visualization.

The sparse model in the second phase admits an additive structure. This enables us to easily interpret different effects of features on the resulting model. Learning sparse models can be done typically by imposing sparsity-inducing regularizers to the optimization problem. The advantages of sparsity include model interpretability and also computational convenience.

2 Related Work

There have been increasing interests in developing interpretable models in recent years. The class of generalized additive models (GAMs) [1] relate the output values to a sum of univariate functions. Thanks to this additive structure, GAMs can be easily interpreted by humans. It is reported that fitting GAMs by using gradient boosting with regression trees can achieve high accuracy [12]. However, one of its potential weaknesses is the scalability issue due to the sequential manner of boosting.

More recently, post hoc methods such as LIME [13] have been developed to provide explanations for predictive models. These methods typically involve learning an extra model to interpret the fitted model of interest. On the other hand, our approach simultaneously performs prediction and interpretation.

Our approach is also closely related to wrapper methods of feature selection. Broadly, there are two categories of feature selection methods: wrapper and filter. Wrapper methods perform feature selection and prediction simultaneously; while filter methods select important features based on some statistical criterion, which is not directly connected to the learning algorithm.

Lasso [14] and ℓ_1-SVM [15] are two classic wrapper methods for regression and classification respectively. By using ℓ_1-regularizer, important features can be identified. Accordingly, the effects of these features on the resulting model can be interpreted by humans. Moreover, Lasso and ℓ_1-SVM are useful in large-scale problems thanks to the computational efficiency. However, a critical weakness is that nonlinear relations can not be captured. SVM recursive feature elimination (SVM-RFE) [16] is another wrapper method. It is originally designed for gene selection in life science. In SVM-RFE, irrelevant features are iteratively eliminated according to some ranking criterion during SVM training. Like Lasso and ℓ_1-SVM, the linear version of SVM-RFE can only capture linear relations. Although kernelized SVM-RFE is able to handle nonlinearity, it is not a scalable method any more. Sparse additive models (SpAM) [17] are a group of methods for nonparametric regression and classification. The optimization problem in SpAM is convex and can be efficiently solved by the back-fitting algorithm. SpAM is very effective for nonlinear feature selection. Nevertheless, since it does not assume any interaction among features, SpAM might perform poorly if interactions exist in the underlying model.

Our approach can be categorized as a scalable wrapper method of feature selection. However, our goal is different from that of feature selection. Specifically, our approach is to interpret and visualize different effects of input features on prediction, whereas feature selection is to choose a small subset of features that is sufficient to construct a predictive model.

3 Problem Setting

Suppose we have a dataset of N input-output data examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where the vector $\mathbf{x}_i := \left(x_i^{(1)}, \ldots, x_i^{(D)}\right) \in \mathcal{X}$ is the i-th example with D features, and $y_i \in \mathcal{Y}$ is the corresponding prediction target. We have $\mathcal{Y} = \mathbb{R}$ for regression and $\mathcal{Y} = \{-1, +1\}$ for classification. A prediction of y is denoted by $\hat{y}$. Let $\mathbf{x}_i^{(p,q)} = \left(x_i^{(p)}, x_i^{(q)}\right)$ be a pairwise feature representation of $\mathbf{x}_i$, containing the p-th and q-th features.

3.1 Kernel Machines

Kernel machines are successful in many real-world tasks and mathematically well-founded. The core of kernel machines is the kernel function $\kappa\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$,

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_{\mathcal{H}},$$

where $\mathcal{H}$ is a high-dimensional Hilbert space and $\phi\colon \mathcal{X} \to \mathcal{H}$ is the corresponding feature map. The strength of kernel machines is that $\phi(\mathbf{x})$ does not need to be explicitly specified, because the inner product can be computed by κ in an relatively inexpensive way (a.k.a. "kernel trick"). There are several kernel functions successfully used in the literature, such as the polynomial kernel $\kappa_{\text{POL}}(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + 1)^u, u \in \mathbb{N}$ and the Gaussian RBF kernel

$$\kappa_{\text{GAU}}(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|_2^2}{2\sigma^2}\right), \quad \sigma \in \mathbb{R}_{++}.$$

The Gaussian kernel is one of *shift-invariant* kernels, a wide class of kernel functions. A kernel function κ is shift-invariant if $\kappa(\mathbf{x}, \mathbf{z}) = \bar{\kappa}(\mathbf{x} - \mathbf{z})$, for some positive definite function $\bar{\kappa}\colon \mathcal{X} \to \mathbb{R}$. It is easy to see that the Gaussian kernel is shift-invariant, while the polynomial kernel is not. Unfortunately, in spite of its elegant properties, the kernel function κ is essentially a black box. Moreover, the associated optimization usually involves computing the $N \times N$ kernel matrix. This makes kernel machines unappealing in the large-scale scenario.

3.2 Random Fourier Features

Random Fourier features (RFF) [11] is an attractive and popular technique for improving scalability of shift-invariant kernels. The key insight is from a classical theorem [18] in harmonic analysis, which guarantees that

$$\kappa(\mathbf{x}, \mathbf{z}) = \bar{\kappa}(\Delta) = \int_{\mathbb{R}^D} e^{\sqrt{-1}\omega^\mathsf{T}\Delta} d\mathbb{P}(\omega).$$

Since $\bar{\kappa}(\Delta)$ is real-valued, we may have

$$\kappa(\mathbf{x}, \mathbf{z}) = \bar{\kappa}(\Delta) = \int_{\mathbb{R}^D} \cos(\boldsymbol{\omega}^\mathsf{T}\Delta) d\mathbb{P}(\boldsymbol{\omega}) = \mathbb{E}\left[\cos(\boldsymbol{\omega}^\mathsf{T}\Delta)\right].$$

Now build an *explicit* feature map $\hat{\phi}\colon \mathbb{R}^D \to \mathbb{R}^d$ as

$$\hat{\phi}(\mathbf{x}) := \sqrt{\frac{2}{d}} \left(\cos(\boldsymbol{\omega}_1^\top \mathbf{x}), \sin(\boldsymbol{\omega}_1^\top \mathbf{x}), \ldots, \cos(\boldsymbol{\omega}_{d/2}^\top \mathbf{x}), \sin(\boldsymbol{\omega}_{d/2}^\top \mathbf{x})\right), \quad \{\boldsymbol{\omega}_i\}_{i=1}^{d/2} \overset{i.i.d.}{\sim} \mathbb{P}. \tag{1}$$

Then the corresponding kernel function $\hat{\kappa}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ can be written as

$$\hat{\kappa}(\mathbf{x}, \mathbf{z}) = \frac{2}{d} \sum_{i=1}^{d/2} \cos(\boldsymbol{\omega}_i^\top \mathbf{x}) \cos(\boldsymbol{\omega}_i^\top \mathbf{z}) + \sin(\boldsymbol{\omega}_i^\top \mathbf{x}) \sin(\boldsymbol{\omega}_i^\top \mathbf{z}) = \frac{2}{d} \sum_{i=1}^{d/2} \cos(\boldsymbol{\omega}_i^\top \boldsymbol{\Delta}).$$

It is easy to see the condition $\mathbb{E}\left[\hat{\kappa}(\mathbf{x}, \mathbf{z})\right] = \kappa(\mathbf{x}, \mathbf{z})$ is satisfied. Hence $\hat{\kappa}(\mathbf{x}, \mathbf{z})$ is a sampling-based approximation of $\kappa(\mathbf{x}, \mathbf{z})$. Unlike the original feature map ϕ, $\hat{\phi}$ is relatively low-dimensional. This enables us to simply transform the input data with $\hat{\phi}$, and then exploit efficient linear algorithms to approximate the original nonlinear kernel machines.

4 Our Approach

In this section, we present our approach to learning with interpretable and scalable kernel machines (ISK). In ISK, we first design a new kernel function and then develop a two-phase learning procedure.

4.1 Kernel Function

We use an explicit feature map via RFF to approximate a certain kernel function. This kernel could be any shift-variant kernel. In this paper, we focus on the Gaussian kernel. By using RFF to approximate the Gaussian kernel, we have

$$\kappa_{\text{GAU}}(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \approx \hat{\kappa}_{\text{GAU}}(\mathbf{x}, \mathbf{z}) = \langle \hat{\phi}(\mathbf{x}), \hat{\phi}(\mathbf{z}) \rangle,$$

where $\hat{\phi}$ is in the form of Eq. (1) and $\{\boldsymbol{\omega}_i\}_{i=1}^{d/2}$ are randomly sampled according to $\mathcal{N}(\mathbf{0}_D, \sigma^{-2}\mathbf{I}_D)$. Here, $\mathcal{N}(\mu, \boldsymbol{\Sigma})$ denotes the multi-variate Gaussian distribution with mean μ and covariance matrix $\boldsymbol{\Sigma}$.

Now we evaluate pairwise features $\mathbf{x}^{(p,q)}$ on $\hat{\kappa}_{\text{GAU}}$:

$$\hat{\kappa}_{\text{GAU}}(\mathbf{x}^{(p,q)}, \mathbf{z}^{(p,q)}) = \langle \hat{\phi}(\mathbf{x}^{(p,q)}), \hat{\phi}(\mathbf{z}^{(p,q)}) \rangle$$

with the feature map $\hat{\phi}\colon \mathbb{R}^2 \to \mathbb{R}^d$. Then we define our kernel function $\kappa_{\text{ISK}}\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ as an average of sub-kernels on all distinct pairwise features:

$$\kappa_{\text{ISK}}(\mathbf{x}, \mathbf{z}) := \frac{1}{K} \sum_{p=1}^{D} \sum_{q>p}^{D} \hat{\kappa}_{\text{GAU}}(\mathbf{x}^{(p,q)}, \mathbf{z}^{(p,q)}) = \langle \phi_{\text{ISK}}(\mathbf{x}), \phi_{\text{ISK}}(\mathbf{z}) \rangle,$$

where $K = \binom{D}{2}$ is the total number of feature pairs, and the corresponding feature map $\phi_{\text{ISK}} \colon \mathbb{R}^D \to \mathbb{R}^{Kd}$ can be explicitly written as:

$$\begin{aligned}\phi_{\text{ISK}}(\mathbf{x}) := \sqrt{\frac{2}{Kd}} \Big(&\cos(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x}^{(1,2)}), \sin(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x}^{(1,2)}), \ldots, \\ &\cos(\boldsymbol{\omega}_{d/2}^\mathsf{T}\mathbf{x}^{(1,2)}), \sin(\boldsymbol{\omega}_{d/2}^\mathsf{T}\mathbf{x}^{(1,2)}), \ldots, \\ &\cos(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x}^{(D-1,D)}), \sin(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x}^{(D-1,D)}), \ldots, \\ &\cos(\boldsymbol{\omega}_{d/2}^\mathsf{T}\mathbf{x}^{(D-1,D)}), \sin(\boldsymbol{\omega}_{d/2}^\mathsf{T}\mathbf{x}^{(D-1,D)})\Big), \\ &\{\boldsymbol{\omega}_i\}_{i=1}^{d/2} \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}_2, \sigma^{-2}\mathbf{I}_2). \end{aligned} \tag{2}$$

There are also other kernel approximation techniques such as Nyström method [19,20]. The reason why we prefer RFF is that the sub-kernels in our kernel function are of similar forms, and RFF can be utilized for them in common.

4.2 Phase 1: Learning Dense Models

Since the feature map ϕ_{ISK} is explicitly built, we can easily transform the input data and exploit efficient linear algorithms to learn a dense (i.e. non-sparse) linear model

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi_{\text{ISK}}(\mathbf{x}) \rangle + w_0 \tag{3}$$

by solving the following convex optimization problem:

$$\min_{\mathbf{w} \in \mathbb{R}^{Kd}, w_0 \in \mathbb{R}} \quad \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_i), y_i) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2,$$

where $\mathcal{L}(f(\mathbf{x}), y)$ is a convex loss function and λ is a parameter controlling the importance of the regularization term.

For regression tasks where $\mathcal{Y} = \mathbb{R}$, taking the square loss $\mathcal{L}(f(\mathbf{x}), y) = \frac{1}{2}(y - f(\mathbf{x}))^2$ we obtain a standard ridge regression problem; for classification tasks where $\mathcal{Y} = \{-1, +1\}$, taking the hinge loss $\mathcal{L}(f(\mathbf{x}), y) = \max(0, 1 - yf(\mathbf{x}))$ we obtain a vanilla SVM problem.

4.3 Phase 2: Learning Sparse Models

In this phase, we first extract component representations from the fitted model in Eq. (3). To achieve this, we propose to expand trigonometric functions via Taylor series.

By the Taylor series of the cosine and sine functions around the point 0 (a.k.a. the Maclaurin series), we have

$$\begin{aligned}\cos(\boldsymbol{\omega}_i^\mathsf{T}\mathbf{x}) &= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} (\boldsymbol{\omega}_i^\mathsf{T}\mathbf{x})^{2n}, \\ \sin(\boldsymbol{\omega}_i^\mathsf{T}\mathbf{x}) &= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} (\boldsymbol{\omega}_i^\mathsf{T}\mathbf{x})^{2n+1}. \end{aligned} \tag{4}$$

Using the multinomial theorem to expand $(\boldsymbol{\omega}_i^\top \mathbf{x})^u$ as

$$(\boldsymbol{\omega}_i^\top \mathbf{x})^u = \sum_{p=1}^{D} (\omega_i^{(p)} x^{(p)})^u + \sum_{\substack{\sum_{l=1}^{D} r_l = u \\ r_l \neq u}} \binom{u}{r_1, \ldots, r_D} \prod_{1 \leq p \leq D} (\omega_i^{(p)} x^{(p)})^{r_p}. \tag{5}$$

Plugging Eq. (5) into Eq. (4) and let

$$C^{(p)}(\boldsymbol{\omega}_i, \mathbf{x}) := \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} (\omega_i^{(p)} x^{(p)})^{2n} = \cos(\omega_i^{(p)} x^{(p)}),$$

$$C^{(1,\ldots,D)}(\boldsymbol{\omega}_i, \mathbf{x}) := \cos(\boldsymbol{\omega}_i^\top \mathbf{x}) - \sum_{p=1}^{D} C^{(p)}(\boldsymbol{\omega}_i, \mathbf{x}).$$

Then it is easy to see that $\cos(\boldsymbol{\omega}_i^\top \mathbf{x})$ is separated into main effects and interaction effects as

$$\cos(\boldsymbol{\omega}_i^\top \mathbf{x}) = \sum_{p=1}^{D} C^{(p)}(\boldsymbol{\omega}_i, \mathbf{x}) + C^{(1,\ldots,D)}(\boldsymbol{\omega}_i, \mathbf{x}). \tag{6}$$

Similarly, we have

$$\sin(\boldsymbol{\omega}_i^\top \mathbf{x}) = \sum_{p=1}^{D} S^{(p)}(\boldsymbol{\omega}_i, \mathbf{x}) + S^{(1,\ldots,D)}(\boldsymbol{\omega}_i, \mathbf{x}). \tag{7}$$

In Eqs. (6) and (7), the terms $C^{(p)}(\boldsymbol{\omega}_i, \mathbf{x})$ and $S^{(p)}(\boldsymbol{\omega}_i, \mathbf{x})$ correspond to the main effects of the p-th feature, while $C^{(1,\ldots,D)}(\boldsymbol{\omega}_i, \mathbf{x})$ and $S^{(1,\ldots,D)}(\boldsymbol{\omega}_i, \mathbf{x})$ represent interactions. In our approach, only two-way interactions are included, because all of elements in the feature map ϕ_{ISK} are built on pairwise features.

We then rewrite the inner product $\langle \mathbf{w}, \phi_{\text{ISK}}(\mathbf{x}) \rangle$ in Eq. (3) as a sum, and expand each weighted element of $\phi_{\text{ISK}}(\mathbf{x})$ into three sub-components according to Eqs. (6) and (7). By combining these sub-components and centering the data, we obtain the $(D + K)$-dimensional component representation for each data example $\mathbf{x}$ as

$$\tilde{\mathbf{x}} := \left(\tilde{x}^{(1)}, \ldots, \tilde{x}^{(D)}, \tilde{x}^{(1,2)}, \ldots, \tilde{x}^{(D-1,D)} \right), \tag{8}$$

where the component $\tilde{x}^{(p)}$ is computed from the sum of sub-components related to the p-th feature, and the component $\tilde{x}^{(p,q)}$ is built based on the sum of sub-components that represent the interaction between the p-th and q-th features.

It is worth noting that centering the data should be done to ensure that each column has the zero mean and unit variance. Because we need to guarantee all the components are in the same scale so that the interpretation of their effects is not misleading.

With this interpretable component representation in Eq. (8), we learn a sparse linear model:

$$g(\tilde{\mathbf{x}}) = \boldsymbol{\beta}^{\mathsf{T}}\tilde{\mathbf{x}} + \beta_0 \tag{9}$$

by solving the following convex optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{D+K},\beta_0\in\mathbb{R}} \quad \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(g(\tilde{\mathbf{x}}_i), y_i) + \tilde{\lambda}\rho\|\boldsymbol{\beta}\|_1 + \frac{\tilde{\lambda}(1-\rho)}{2}\|\beta\|_2^2,$$

where $\mathcal{L}(g(\tilde{\mathbf{x}}), y)$ is the loss function used in the first phase and $\tilde{\lambda}$ is a regularization parameter.

Here, the regularization term is a convex combination of ℓ_1-norm and ℓ_2-norm, with the parameter $\rho \in [0, 1]$ controlling their ratio. This is known as the *elastic net* regularizer [21]. Like ℓ_1-norm, elastic net can also yield sparse solutions of β, where only some of the coefficients in the fitted model are nonzero. Hence, components that are relevant for predicting the output values can be successfully identified. Only using ℓ_1-norm as the regularizer could be another choice for this sparse model. We instead use a more general regularizer elastic net because it allows us to balance the trade-off between sparseness and smoothness in practice.

4.4 Interpretation

Finally, we can interpret the resulting sparse model in Eq. (9) by checking component *importances* and *partial dependences* of output values on components in Eq. (8).

Since $\{\tilde{\mathbf{x}}_i\}_{i=1}^N$ are centered before fitting the sparse model in Eq. (9), the importance of the p-th component $\tilde{x}^{(p)}$ can be indicated simply by the magnitude of its coefficient in the fitted model, i.e. $|\beta^{(p)}|$. More specifically, $|\beta^{(p)}|$ represents the difference in the output value for each one-unit/category difference in $\tilde{x}^{(p)}$. Similarly, $|\beta^{(p,q)}|$ indicates the importance of the interaction component $\tilde{x}^{(p,q)}$.

Partial dependence plots (PDPs) are graphical renderings for predictive models [22,23]. By showing the dependence of output values on input features, PDPs are helpful for us to better understand the different effects of input features on the resulting model.

Now define the partial dependence function of the p-th component $\tilde{x}^{(p)}$ as a marginal average of g in Eq. (9):

$$g^{(p)}(\tilde{x}^{(p)}) := \mathbb{E}_{\tilde{\mathbf{x}}^{\setminus(p)}}\left[g(\tilde{x}^{(p)}, \tilde{\mathbf{x}}^{\setminus(p)})\right],$$

where $\tilde{\mathbf{x}}^{\setminus(p)}$ is the vector containing all of the elements of $\tilde{\mathbf{x}}$ except for $\tilde{x}^{(p)}$.

Although $g^{(p)}$ cannot be directly accessed, empirical estimation is possible:

$$\bar{g}^{(p)}(\tilde{x}^{(p)}) := \frac{1}{N}\sum_{i=1}^{N} g(\tilde{x}^{(p)}, \tilde{\mathbf{x}}_i^{\setminus(p)}) = \beta^{(p)}\tilde{x}^{(p)} + \frac{1}{N}\sum_{i=1}^{N}\langle\beta^{\setminus(p)}, \tilde{\mathbf{x}}_i^{\setminus(p)}\rangle. \tag{10}$$

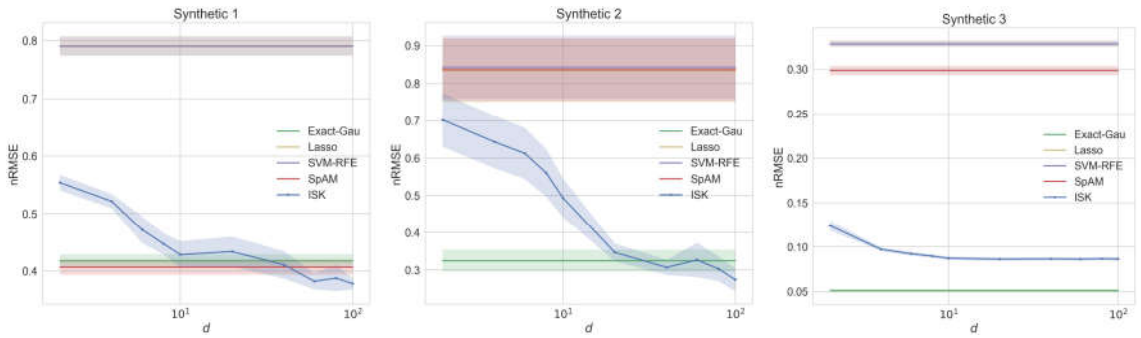

Fig. 1. Results on synthetic data. Normalized root mean square errors (nRMSEs) are averaged on 10 random splits of data. Means and standard errors of nRMSEs are reported. "Exact-Gau" stands for "Kernel ridge regression with the exact Gaussian kernel". The lines of Lasso and SVM-RFE overlap each other because their performances are comparable.

Similarly, the empirical partial dependence function for the interaction component $\tilde{x}^{(p,q)}$ can be written as:

$$\bar{g}^{(p,q)}(\tilde{x}^{(p,q)}) := \beta^{(p,q)}\tilde{x}^{(p,q)} + \frac{1}{N}\sum_{i=1}^{N}\langle \beta^{\setminus(p,q)}, \tilde{\mathbf{x}}_i^{\setminus(p,q)}\rangle. \tag{11}$$

These partial dependence functions can be easily visualized for interpretation of the resulting model.

5 Experiments

In this section, we describe our experimental design and report comparison results. We first conduct synthetic experiments for illustration; then we use several large benchmark datasets for performance comparison; finally, we demonstrate the usefulness of our approach in terms of interpretability by visualization.

We compare our approach with several existing interpretable and scalable *wrapper* methods, including Lasso [14], ℓ_1-SVM [15], SVM-RFE [16] and SpAM [17,24].

5.1 Synthetic Data

To illustrate the properties of our approach, we first generate three synthetic datasets. We generate $N = 1000$ data examples of dimension $D = 5$ for the first two synthetic datasets. Data examples are randomly sampled by following the standard Gaussian distribution. For the third synthetic dataset, we generate $N = 1000$ data examples of dimension $D = 10$, where all of the columns are randomly sampled from the uniform distribution $\mathcal{U}[0, 1]$ except for $x^{(4)}, x^{(5)}, x^{(8)}, x^{(10)} \sim \mathcal{U}[0.6, 1]$. Then three underlying true models are generated as:

$$f_1^*(\mathbf{x}) = -2\sin(2x^{(1)}) + (x^{(2)})^2 + x^{(3)} + \exp(-x^{(4)}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0,1).$$
$$f_2^*(\mathbf{x}) = x^{(1)} \exp(2x^{(2)}) + (x^{(3)})^2 + \epsilon, \quad \epsilon \sim \mathcal{N}(0,1).$$
$$f_3^*(\mathbf{x}) = \pi^{x^{(1)}x^{(2)}}\sqrt{2x^{(3)}} - \sin^{-1}(x^{(4)}) + \log(x^{(3)} + x^{(5)}) - \frac{x^{(9)}}{x^{(10)}}\sqrt{\frac{x^{(7)}}{x^{(8)}}} - x^{(2)}x^{(7)}.$$

Synthetic 1 has a typical additive structure of main effects, which is originally used in [17]; the true model in Synthetic 2 includes a pairwise interaction, and it is generated in [25]; Synthetic 3 involves more complex interactions, and it is used in [26].

We also test kernel ridge regression with the Gaussian kernel (Exact-Gau) for reference. We evaluate all the methods in terms of prediction performance, by using the normalized root mean square error (nRMSE) as the metric. For our approach, we vary values of d in Eq. (2) to see how the performance changes.

Each dataset is randomly split into training data (80%) and test data (20%). We perform 5-fold cross-validation on training data for tuning parameters. The model is finally refitted and evaluated on test data. The entire procedure is repeated for 10 times.

Figure 1 shows the comparison results on synthetic datasets. We can observe that the performance of our approach gets better and better as d increases in all the three cases. This is consistent with the theoretical properties of RFF. When d is large enough, our approach can perform comparably well as Exact-Gau.

The performance of SpAM is extremely good on Synthetic 1 (even better than Exact-Gau). This is because SpAM assumes an additive structure of main effects, which is exactly how we generate data in Synthetic 1. However, when we add interaction effects in Synthetic 2 and 3, SpAM cannot perform well any more. Generally, Lasso and SVM-RFE have comparable worst performances on all the three synthetic datasets. This is not surprising because Lasso and SVM-RFE cannot capture nonlinear relations. On the other hand, Exact-Gau achieves high performance in all the three cases, owing to its capability of modeling infinite-order nonlinear interactions. However, as a standard kernel method, Exact-Gau is difficult to scale up to larger datasets because its computational complexity is quadratic in the number of data example N, while that of our approach is linear in N.

5.2 Benchmark Data

We conduct comparison experiments using public benchmark data as listed in Table 1, including five regression and five classification datasets.

Most of these datasets are from the application domains where interpretability is highly required, such as "Parkinsons" from life science and "Bank" from marketing analytics. We decided not to include Exact-Gau in benchmark experiments because of its high costs of storage and computation. For regression tasks, we use the normalized root mean square error (nRMSE) as the metric; for classification tasks, we use the error rate as the metric. Error rate is calculated as the proportion of incorrect predictions of the class label.

Table 1. Specifications of benchmark data. Five for regression and five for classification.

Dataset	# of examples	# of features	% of pos.
Parkinsons	5875	20	-
Pumadyn	8192	8	-
CalHousing	20640	8	-
Kin40k	40000	8	-
Protein	45730	9	-
Spambase	4601	57	39.40
Eye	14980	14	44.88
Credit	30000	23	22.12
Bank	41188	20	11.27
Cod-RNA	59535	8	33.33

Table 2. Results on benchmark datasets. Normalized root mean square errors (nRMSEs) and error rates are averaged on 10 random splits of data. Means and standard deviations of nRMSEs and error rate (%) are reported.

	Lasso/ℓ_1-SVM	SVM-RFE	SpAM	ISK
Parkinsons	88.90 (1.41)	88.87 (1.56)	82.88 (1.34)	**56.49 (2.33)**
Pumadyn	79.06 (1.16)	79.02 (1.14)	74.90 (0.86)	**61.74 (3.70)**
CalHousing	60.95 (1.02)	60.98 (1.11)	59.08 (1.06)	**57.03 (1.50)**
Kin40k	100.09 (0.75)	100.10 (0.74)	97.38 (0.67)	**82.98 (2.33)**
Protein	84.77 (0.48)	84.85 (0.54)	82.41 (0.37)	**80.05 (0.99)**
Spambase	7.82 (1.01)	7.75 (0.82)	6.91 (1.48)	**6.38 (0.89)**
Eye	39.20 (1.49)	39.99 (1.36)	34.78 (2.06)	**19.92 (3.40)**
Credit	19.13 (1.00)	19.87 (1.44)	19.89 (0.20)	**17.90 (0.39)**
Bank	9.75 (0.23)	9.55 (0.35)	9.44 (0.15)	**9.15 (0.33)**
Cod-RNA	6.16 (0.12)	6.18 (0.13)	5.76 (0.44)	**4.96 (0.21)**

Table 2 shows the comparison results on benchmark datasets. From the results, we can see our approach yields the smallest prediction errors and performs better than other methods on all the datasets. SpAM has the second best performance in general. The reason why our approach performs better than SpAM might be that our approach is able to detect interaction effects between features. Lasso/ℓ_1-SVM and SVM-RFE perform the worst among all the methods. This might be due to the linearity of Lasso/ℓ_1-SVM and the usage of linear SVM in SVM-RFE instead of the kernel-based version for the purpose of scalability.

5.3 Visualization

Finally, we demonstrate how to interpret the resulting model by checking component importances and visualizing partial dependences. We compute these quantities based on the sparse model in Eq. (9) fitted on the whole dataset. Due to the space limitation, we only show the figures from the "CalHousing" data [27].

Table 3. Features in "CalHousing".

Name	Description
MedInc	Median income
HouseAge	Housing median age
AveRooms	Average number of rooms
AveBedrms	Average number of bedrooms
Population	Population in each block group
AveOccup	Average occupancy in each house
Latitude	Geographic coordinate (north-south)
Longitude	Geographic coordinate (east-west)

"CalHousing" consists of $N = 20460$ examples, each of which represents the aggregated data of a block group in California from the 1990 Census. The prediction target is the median house value in each block group, and the features ($D = 8$) include demographic, geographical and residential information. The feature information of "CalHousing" is summarized in Table 3.

We perform our approach on the "CalHousing" data. We first check component importances by looking at the coefficients of the fitted sparse model in Eq. (9). Then we calculate the *relative* importance for each component. Since the components with low importances do not have significant effects on the resulting model, we are more interested in top components (e.g. those with relative importance ≥ 0.2).

Figure 2 shows the ranking of top components on the "CalHousing" data. The main effect of MedInc is identified as the most important component for predicting house value, followed by location-related components: the main effects of Latitude and Longitude, together with the interaction between them. AveOccup and AveRooms have less than half the importance of MedInc, so they are somehow relatively unimportant among these top 6 components.

We then calculate the partial dependences of house value on these top ranking components by using Eqs. (10) and (11).

The first three plots in Fig. 3 show the partial dependences on main effect components. We can observe that the partial dependence of house value is generally monotonic increasing as MedInc increases over the main body of data. House value has a linearly decreasing partial dependence on AveOccup. The partial dependence of house value on AveRooms is a non-monotonic, where the

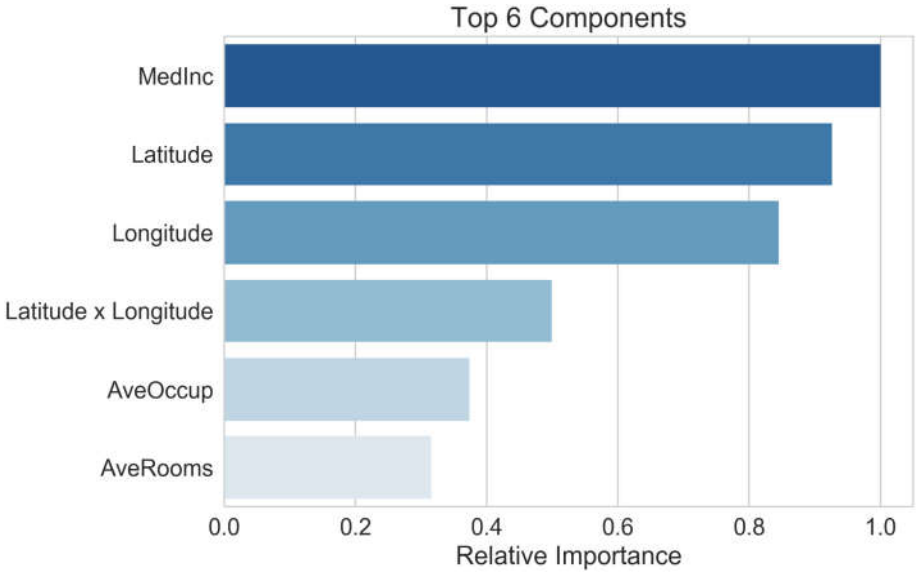

Fig. 2. Ranking of top 6 components (relative importance ≥ 0.2) on CalHousing data.

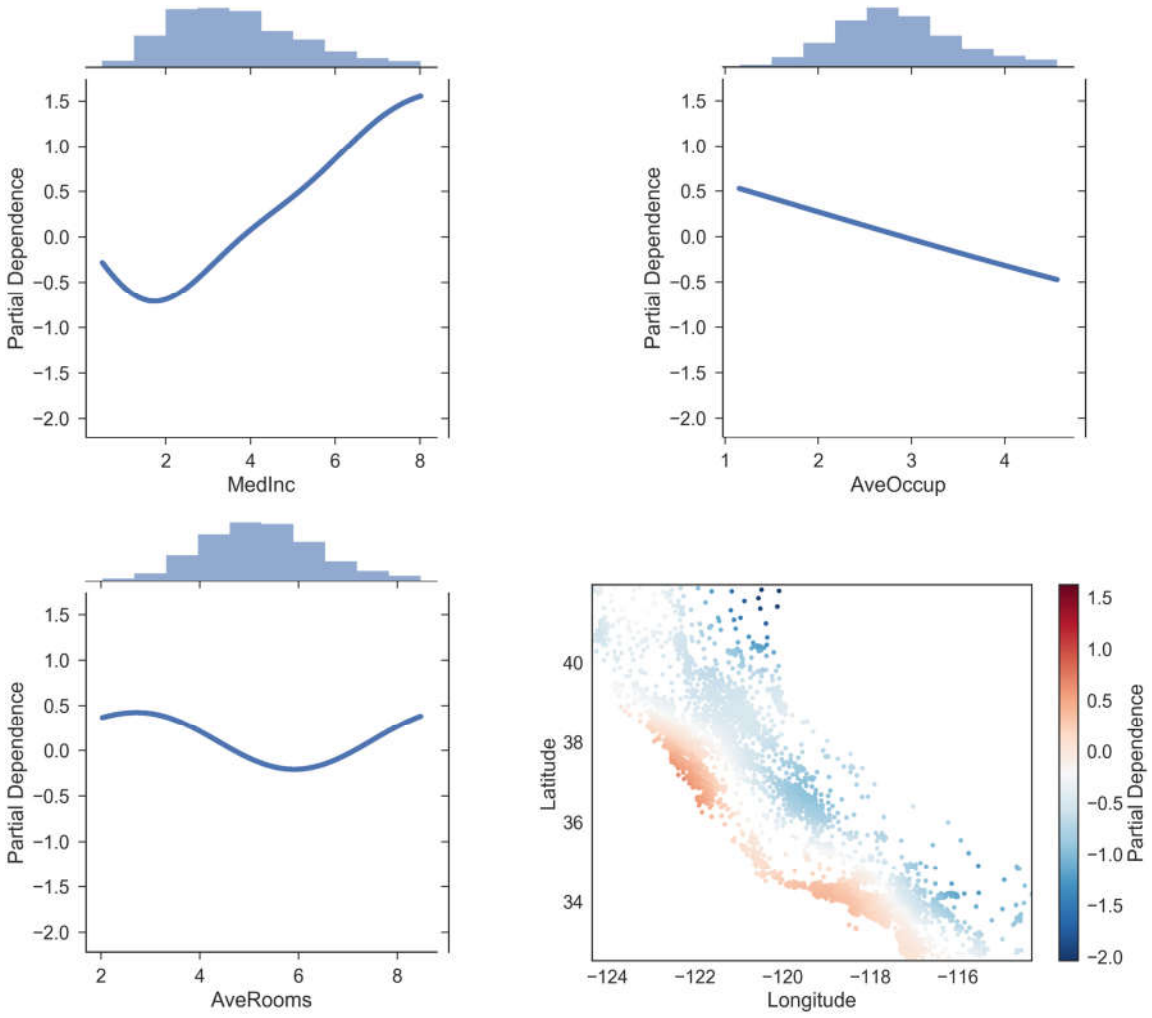

Fig. 3. Partial dependence plots on "CalHousing".

minimum is approximately at 6 rooms, roughly corresponding to the highest proportion in the data.

The partial dependences on location-related components are rather interesting. The last plot in Fig. 3 shows the partial dependence on the interaction between Latitude and Longitude. House value is seen to largely depend on this interaction effect along the coastline of California, especially in the area around (37, −122), which is near the Bay Area of California.

6 Conclusion

We proposed a novel approach to interpretable learning with kernel machines. We first designed a new kernel function based on random Fourier features for the scalability purpose, and then developed a learning procedure including two phases: in the first phase, we map all pairs of features to an explicit feature space produced by the designed kernel to learn a dense linear model; in the second phase, we use a component extraction trick to represent the original data in an interpretable way for learning a sparse linear model. Experimental results on several large benchmark datasets showed the predictive power of our approach, and the visualization demonstrates its usefulness for interpretability.

References

1. Hastie, T.J., Tibshirani, R.J.: Generalized Additive Models. Chapman & Hall/CRC, Boca Raton (1990)
2. Lipton, Z.C.: The mythos of model interpretability. In: ICML 2016 Workshop on Human Interpretability in Machine Learning (WHI2016) (2016)
3. Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., Elhadad, N.: 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 (KDD2015), pp. 1721–1730. ACM (2015)
4. Zeng, J., Ustun, B., Rudin, C.: Interpretable classification models for recidivism prediction. J. Royal Stat. Soc. Ser. A (Stat. Soc.) **180**, 689–722 (2016)
5. Letham, B., Letham, L.M., Rudin, C.: Bayesian inference of arrival rate and substitution behavior from sales transaction data with stockouts. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2016), pp. 1695–1704. ACM, New York (2016)
6. Breiman, L.: Statistical modeling: the two cultures (with comments and a rejoinder by the author). Stat. Sci. **16**(3), 199–231 (2001)
7. Chang, B.H.W.: Kernel machines are not black boxes - on the interpretability of kernel-based nonparametric models. Ph.D. thesis, University of Toronto (2014)
8. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
9. Vapnik, V., Golowich, S.E., Smola, A.J.: Support vector method for function approximation, regression estimation and signal processing. In Jordan, M.I., Petsche, T. (eds.) Advances in Neural Information Processing Systems (NIPS 1996), vol. 9, pp. 281–287. MIT Press (1997)

10. Van Belle, V., Lisboa, P.: White box radial basis function classifiers with component selection for clinical prediction models. Artif. Intell. Med. **60**(1), 53–64 (2014)
11. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) Advances in Neural Information Processing Systems (NIPS 2007), vol. 20, pp. 1177–1184. Curran Associates, Inc. (2008)
12. Lou, Y., Caruana, R., Gehrke, J.: Intelligible models for classification and regression. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2012), pp. 150–158. ACM (2012)
13. Ribeiro, M.T., Singh, S., Guestrin, C.: "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 (KDD 2016), pp. 1135–1144. ACM (2016)
14. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Royal Stat. Soc. Ser. B Methodol. **58**, 267–288 (1996)
15. Zhu, J., Rosset, S., Tibshirani, R., Hastie, T.J.: 1-norm support vector machines. In: Thrun, S., Saul, L.K., Schölkopf, P.B. (eds.) Advances in Neural Information Processing Systems (NIPS 2003), vol. 16, pp. 49–56. MIT Press (2004)
16. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1–3), 389–422 (2002)
17. Ravikumar, P., Lafferty, J., Liu, H., Wasserman, L.: Sparse additive models. J. Royal Stat. Soc. Ser. B (Stat. Methodol.) **71**(5), 1009–1030 (2009)
18. Bochner, S.: Lectures on Fourier Integrals. Princeton University Press, Princeton (1959)
19. Williams, C.K.I., Seeger, M.: Using the Nyström method to speed up kernel machines. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems (NIPS 2000), vol. 13, pp. 682–688. MIT Press (2001)
20. Drineas, P., Mahoney, M.W.: On the Nyström method for approximating a Gram matrix for improved kernel-based learning. J. Mach. Learn. Res. **6**, 2153–2175 (2005)
21. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. Royal Stat. Soc. Ser. B (Stat. Methodol.) **67**(2), 301–320 (2005)
22. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001)
23. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics, 2nd edn. Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7
24. Zhao, T., Liu, H.: Sparse additive machine. In: Lawrence, N.D., Girolami, M.A. (eds.) Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2012), pp. 1435–1443 (2012)
25. Yamada, M., Jitkrittum, W., Sigal, L., Xing, E.P., Sugiyama, M.: High-dimensional feature selection by feature-wise non-linear lasso. Neural Comput. **26**(1), 185–207 (2014)
26. Lou, Y., Caruana, R., Gehrke, J., Hooker, G.: Accurate intelligible models with pairwise interactions. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2013), pp. 623–631. ACM (2013)
27. Pace, R.K., Barry, R.: Sparse spatial autoregressions. Stat. Probab. Lett. **33**(3), 291–297 (1997)