Abstract

Kernel methods provide a flexible and theoretically grounded approach to nonlinear and nonparametric learning. While memory and run-time requirements hinder their applicability to large datasets, many low-rank kernel approximations, such as random Fourier features, were recently developed to scale up such kernel methods. However, these scalable approaches are based on approximations of isotropic kernels, which cannot remove the influence of irrelevant features. In this work, we design random Fourier features for a family of automatic relevance determination (ARD) kernels, and introduce RFFNet, a new large-scale kernel method that learns the kernel relevances’ on the fly via first-order stochastic optimization. We present an effective initialization scheme for the method’s non-convex objective function, evaluate if hard-thresholding RFFNet’s learned relevances yield a sensible rule for variable selection, and perform an extensive ablation study of RFFNet’s components. Numerical validation on simulated and real-world data shows that our approach has a small memory footprint and run-time, achieves low prediction error, and effectively identifies relevant features, thus leading to more interpretable solutions. We supply users with an efficient, PyTorch-based library, that adheres to the scikit-learn standard API and code for fully reproducing our results.

Keywords: supervised learning, kernel methods, feature importance
1. Introduction

Statistical learning methods based on kernels are successfully applied in many fields, including biology [56], social sciences [20], physics [44], and astronomy [14, 62]. Given their nonparametric nature and many theoretical guarantees, kernel methods allow for principled modeling of complex relationships in real-world data. These methods have also sparked extensive methodological research in statistics, leading to the development of kernel-based tools for feature selection [22, 25], causal inference [68], hypothesis testing [24, 37, 58, 54], and privacy [5]. In particular, cornerstones of machine learning, such as kernel ridge regression, Gaussian processes [50], kernel principal component analysis [55], support vector machines [10], and neural tangent kernels [23] are based on kernels.

Nevertheless, the applicability of kernel methods is limited by two main factors: scalability and interpretability. Firstly, since these methods operate on all pairs of observations, they require computation (e.g. matrix inversions) and storage that scales at least quadratically in the sample size, which can be prohibitive for large datasets. Fortunately, several kernel approximations were recently developed for scaling up kernel methods, such as random Fourier features methods (RFF) [48, 34, 11] and the Nyström approximation [52, 15].

Secondly, because kernel methods implicitly embed the original feature space into a high-dimensional ambient space, they often lose information about which features of input data are relevant for prediction [9]. This, in turn, translates into an interpretability shortage and also leads to poor predictive performance in problems with many irrelevant variables [31, 6]. This shortage is particularly significant considering the widespread presence of kernel methods in the toolsets of statistics practitioners.

1.1. Method Overview and Novelty

To address the interpretability issue, a common approach is to introduce feature weighting in the kernel parametrization. This process generates the so-called automatic relevance determination (ARD) kernels [50], as the ARD Gaussian kernel

\[ k_\theta(x, x') = \exp \left[ -\frac{1}{2} \sum_{j=1}^{d} \theta_j^2 (x_j - x'_j)^2 \right], \]
Figure 1: RFFNet combines fast kernel approximations to ARD kernels to carefully initialized and efficient stochastic gradient descent methods. The core part of RFFNet is implemented as a single PyTorch layer and can be seamlessly connected to other PyTorch-based models, making RFFNet highly modular.

where $x_j$ is the $j$-th component of the feature vector $x$ and $\theta_j$ is the relevance of the $j$-th feature. In practice, it is expected that if the relevance vector $\theta$ is estimated using available data, then $\theta_i$ would be shrunk to zero if $x_i$ corresponds to an irrelevant feature.

This work introduces RFFNet (see Figure[1]), a novel approach for large-scale, interpretable kernel methods. Our method relies on three key components:

1. **Fast kernel approximation via random Fourier features**: This approach significantly reduces the number of parameters to be estimated and decreases the computational overhead of using kernels.

2. **Well-designed automatic relevance determination kernels**: We demonstrate that ARD kernels correspond to spectral densities wherein kernel relevances act as scale parameters. We then show that it is possible to decouple $\theta$ from the random features map that approximates an ARD kernel’s evaluations and promptly estimate $\theta$ with data.

3. **Efficient first-order stochastic optimization methods**: carefully initialized parameters and mini-batching help coping with RFFNet’s nonconvex objective function.
In practice, given a training sample \( \{(x_i, y_i)\}_{i=1}^n \) and an ARD kernel \( k_\theta \), **RFFNet** is designed to mimic the solution to the following problem: find the relevances \( \theta \) and the prediction function \( g \) that

\[
\min_{\theta \in \mathbb{R}^p, g \in \mathcal{H}_\theta} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g(x_i)) + \lambda \|g\|_{\mathcal{H}_\theta}^2,
\]

where \( \mathcal{H}_\theta \) is the reproducing kernel Hilbert space (RKHS) associated with \( k_\theta \). In its general form, (1) is a known problem in feature selection for kernel methods [8, 25]. However, as we will show, the restriction to ARD kernels makes (1) particularly well-suited to random features approximations. In addition, possible drawbacks of optimizing (1) can be tackled with modern stochastic first-order methods [29]. This is the direction taken by **RFFNet**.

### 1.2. Relation to other work

There is an extensive literature on kernel approximations and its applications to machine learning [48, 49, 34, 11, 21, 26, 40, 52, 66, 33, 47]. However, much of the focus has been on approximating isotropic kernels, which weigh relevant and irrelevant features equally in the kernel.

There are few exceptions to this trend. A La Carte [66] leverages on Fastfood [34] to learn the spectrum of expressive kernels via marginal likelihood optimization. It is similar to Sparse Spectrum Gaussian Process Regression [33], which uses the inducing points approximation [47] instead. These methods significantly differ from **RFFNet**, since they: (i) require solving two separate problems, the marginal likelihood optimization and the kernel ridge matrix inversion, and (ii) are restricted to the regression setting. Falkon [52] is more akin to our approach. It is based on the Nyström approximation to the kernel matrix and uses first-order optimization to solve kernel ridge regression problems. Falkon [41] also supports optimizing kernel hyperparameters, such as the relevances of an ARD kernel, but this requires minimizing an additional objective. Nevertheless, interpretability is rarely a focal point in these approaches, which prioritize low memory usage and run-time.

Our approach is closer in spirit to Sparse Random Fourier Features [18], which uses random Fourier features approximations for ARD kernels, tailoring the objective function and optimization constraints to induce feature selection. Still, significant differences set us apart. First, **RFFNet** handles any differentiable loss function, while SRFF is specifically designed for squared
error loss problems in regression. Second, **RFFNet** is based on jointly minimizing the kernel relevances and remaining parameters with efficient stochastic gradient descent methods. In fact, using mini-batches is an indispensable component of **RFFNet**. SRFF, instead, is based on alternately minimizing two objectives, a procedure that can cause indefinite cycling of the algorithm [46] and often leads to poor performance. Third, the relevances output by SRFF after a single run are unreliable sources of feature importances. Finally, **RFFNet** is implemented as an efficient library and can be promptly integrated into data analysis pipelines based on the widely adopted scikit-learn standard.

Our work also overlaps with the literature in kernel learning [69, 59, 39, 35, 32, 30, 16]. Nonetheless, kernel learning is often based on alignment objectives, which seek to produce a meaningful representation of data using kernels [59]. In this scenario, even after learning the kernel, users would need to train a kernel-based predictive model to produce outputs for given labeled data. Within the literature of kernel learning, Automated Spectral Kernel Learning (ASKL) [35] has the most resemblance with **RFFNet**. However, ASKL does not update the lengthscales of the kernel it seeks to learn and thus falls short of removing the influence of irrelevant features. Consequently, it is not possible to subsidize interpretative claims based on ASKL-learned spectral density frequencies.

Finally, other approaches to remove irrelevant features in kernel methods include cross-validation [27] and recursive feature backward elimination [19, 38]. These methods, however, do not scale with the number of features and the sample size. An alternative approach more related to our method is to develop a loss function that includes feature-wise relevances as part of the objective function. This is done by Allen [1], which proposes an iterative feature extraction method. Unfortunately, since this procedure does not use kernel approximations, it has prohibitive memory and run-time requirements.

Recently, Jordan et al. [25] described a new sparsity-inducing mechanism for kernel methods. Although the mechanism is based on optimizing a vector of weights that enters the objective as a data scaling, similar to [1], the solution is specific to kernel ridge regression and metric learning. In addition, it does not employ any kernel matrix approximation and thus has limited applicability to large datasets.
1.3. Notation and Organization

For any integer \( n \geq 1 \), we denote by \([n]\) the set \{1, 2, \ldots, n\}. In addition, we indicate by \( \circ \) the element-wise vector multiplication; that is, if \( x, y \in \mathbb{R}^d \), then \( x \circ y = (x_1y_1, \ldots, x_dy_d) \). We write \( I_d \) for the \( d \times d \) identity matrix and \( 1_d = (1, \ldots, 1) \) for the \( d \)-dimensional vector of ones. If \( z \in \mathbb{C} \), then \( z^* \) is its complex conjugate.

The remainder of the paper is organized as follows. We give an overview of kernel methods, random Fourier features, and automatic relevance determination (ARD) kernels in Section 2. In Section 3, we introduce our method and its properties. In Section 4, we validate our approach in simulated and real-world datasets. Finally, in Section 5, we conclude with a brief discussion and directions for future work. All proofs are deferred to Appendix A.

2. Background

2.1. Kernel methods

We follow the standard formalization of the supervised learning setting [57, 42]. Let \( \mathcal{X} \subseteq \mathbb{R}^d \) be the instance space, \( \mathcal{Y} \) a label space, and \( \rho(x, y) \) a probability measure on \( \mathcal{X} \times \mathcal{Y} \). Throughout this paper, we assume \( \mathcal{Y} \subseteq \mathbb{R} \) for regression tasks and \( \mathcal{Y} = \{0, 1\} \) for binary classification tasks. We denote by \( \ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+ \cup \{+\infty\} \) a loss function (e.g., squared error loss, cross-entropy). Given a finite training sample \( \{(x_i, y_i)\}_{i=1}^n \) sampled independently from the joint measure \( \rho \), the goal of supervised learning is to find a hypothesis \( f : \mathcal{X} \rightarrow \mathcal{Y} \) with small expected loss or generalization error \( R(f) = \mathbb{E}[\ell(f(X), Y)] \), where the expectation is taken with respect to the pair \( (X, Y) \sim \rho \).

In kernel methods, we restrict \( f \) to the Reproducing Kernel Hilbert Space (RKHS) \( \mathcal{H} \) uniquely associated to a positive semi-definite (PSD) kernel \( k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \). We will consider kernel methods that result from solving the broad class of regularized loss problems on \( \mathcal{H} \) given by

\[
 f^* \in \arg \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2, \tag{2}
\]

where \( \lambda > 0 \) is the regularization hyperparameter, and \( \| \cdot \|_{\mathcal{H}} \) is the RKHS norm (see, for instance, Wainwright [63] for a explicit characterization).
The representer theorem [28, 42] guarantees that the problem in (2) has a solution of the form \( f^*(\cdot) = \sum_{i=1}^{n} \alpha_i^* k(x_i, \cdot) \) with \( \alpha^* \in \mathbb{R}^n \). It can be readily shown that \( \alpha^* \) satisfies

\[
\alpha^* \in \arg\min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, (K\alpha)_i) + \lambda \alpha^\top K \alpha,
\]

where \( K \) is the kernel matrix, with elements \( K_{ij} = k(x_i, x_j) \), and \((K\alpha)_i = \sum_{j=1}^{n} \alpha_j k(x_i, x_j)\).

If we choose \( \ell(y, y') = (y - y')^2 \), the squared error loss, then problem (3) is known as kernel ridge regression (KRR). In this case, the solution to (3) is \( \alpha^* = (K + n\lambda I)^{-1} y \), where \( y = (y_1, \ldots, y_n)^\top \). Hence the scalability problem: computing the KRR estimator requires \( \Theta(n^3) \) time and \( \Theta(n^2) \) space [2], which can be prohibitive for large datasets. In general, computing a solution to (3) has, at least, \( O(n^2) \) time and space complexities [52].

To reduce this computational cost, it is usual to resort to kernel matrix approximations [48, 53, 34, 52] that can effectively scale kernel methods to large datasets, as discussed in Section 1.2.

2.2. Random Fourier Features

Random Fourier features [48, 49] is a widely used and theoretically grounded [53, 36] framework for scaling up kernel methods. Its guiding idea is to construct a feature map that approximates kernel evaluations of a given shift-invariant PSD kernel.

Let \( k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \) be a continuous, PSD, and shift-invariant kernel of the form \( k(x, x') = G(x - x') \). Then, by Bochner’s Theorem [42], \( k \) is the Fourier transform of a bounded non-negative measure \( p(\cdot) \). That is,

\[
G(x) = \int_{\mathcal{X}} p(\omega) e^{i\omega^\top x} d\omega.
\]

If we further assume that \( G(0) = 1 \), then \( p(\cdot) \) is a probability density called the spectral density of the kernel. In this case, the kernel function can be

\[1\text{Here, since } \mathcal{X} \subseteq \mathbb{R}^d, \text{ the domain is a locally compact set.}\]
written as
\[
k(x, x') = G(x - x') = \int_{\mathcal{X}} e^{i\omega^T(x-x')}p(\omega)d\omega
= \int_{\mathcal{X}} \phi(x)\phi^*(x')p(\omega)d\omega
= \mathbb{E}_\omega[\phi(x)\phi^*(x')],
\] (4)
where \(\phi(x)\) is defined as \(\phi(x) = e^{i\omega^T x}\). Since we assumed \(k\) is real-valued, then (4) can be further simplified as
\[
k(x, x') = \mathbb{E}_\omega[\cos(\omega^T(x - x'))].
\]
From this point, the construction of random Fourier features proceeds as follows [48]: sample \(\{\omega_i\}_{i=1}^s \sim p(\omega)\) independently and \(\{b_i\}_{i=1}^s \sim \text{Uniform}(0, 2\pi)\), then the feature map \(z : \mathbb{R}^d \to \mathbb{R}^s\) with entries
\[
z_i(x) = \sqrt{2/s} \cos(\omega_i^T x + b_i)
\] (5)
satisfies
\[
\mathbb{E}_{\omega,b}[z(x)^T z(x')] = \mathbb{E}_\omega[\cos(\omega^T(x - x'))] = k(x, x').
\]
In other words, the Monte-Carlo estimate
\[
\hat{k}(x, x') = z(x)^T z(x') = \frac{1}{s} \sum_{i=1}^s z_i(x)z_i(x')
\] (6)
is an unbiased estimator of \(k(x, x')\), which can be used to approximate kernel evaluations.

However, the benefits of learning with random Fourier features are not fully evident when working at the level of kernel matrix approximations. First, observe that the function \(\hat{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}\) defined in [6] is itself a PSD kernel [63], and thus, it is uniquely associated to a reproducing kernel Hilbert space \(\mathcal{H}\) (not necessarily contained in \(\mathcal{H}\)). Thus, we can also solve problem (2) over \(\mathcal{H}\) [6]. In this case, by the representer theorem, the optimal solution has the form
\[
\hat{f}^*(x) = \sum_{i=1}^s \beta_i^* z_i(x) = z(x)^T \beta^*,
\] (7)
where $\beta^* \in \mathbb{R}^s$ satisfies

$$\beta^* \in \arg \min_{\beta \in \mathbb{R}^s} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, z(x_i)^\top \beta) + \lambda \|\beta\|^2. \quad (8)$$

This is an optimization problem in $\mathbb{R}^s$ instead of $\mathbb{R}^n$ and can be solved much more efficiently than their full kernel matrix counterpart in (3). For instance, if $\ell$ is chosen as the squared error loss, then problem (8) can be written as

$$\beta^* = \arg \min_{\beta \in \mathbb{R}^s} \frac{1}{n} \|y - Z \beta\|^2_2 + \lambda \|\beta\|^2, \quad (9)$$

where $Z \in \mathbb{R}^{n \times s}$ is defined as the matrix which has its $i$-th row equal to $z(x_i)^\top$. The solution to (9) is $\beta^* = (Z^\top Z + n\lambda I_s)^{-1} Z^\top y$, which can be computed in $O(ns^2)$ time and $O(ns)$ storage, rendering random Fourier features much attractive if $s \ll n$. A fundamental question is whether meeting this requirement on $s$ compromises the statistical quality of the resulting kernel estimator. Several papers have answered this question in the negative \cite{36, 53, 3, 2}, demonstrating that RFF-based estimators asymptotically (in the sample size) match the optimality of full-kernel estimators in the $s \ll n$ regime. Despite these results, except for Yao et al. \cite{67}, existing literature does not offer guidance on numerically determining the optimal $s$, creating a gap between theory and practice. In our approach, RFFNet, users are encouraged to adjust the number of random features until their expected accuracy is achieved. Crucially, our experiments show that even a small, fixed number of random features gives satisfactory results.

### 2.3. Automatic Relevance Determination (ARD) kernels

Automatic Relevance Determination (ARD) kernels \cite{43} are widely used for variable selection in Bayesian regression (e.g., Gaussian processes \cite{50, 12}, sparse Bayesian learning \cite{61}), support vector machines \cite{27, 17, 1}, and self-penalizing objectives \cite{25, 51}. This family of kernels, which includes the usual Gaussian, Laplace, Cauchy, and Matérn kernels, is generated by introducing continuous feature weights in shift-invariant kernels, adding a layer of interpretability atop them by controlling how features contribute to the kernel value.

**Definition 1.** A function $k_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is an automatic relevance determination (ARD) kernel if it is a continuous, positive-semidefinite, and
shift-invariant kernel of the form

\[ k_\theta(x, x') = G[\theta \circ (x - x')], \]

(10)

and is uniquely parameterized by \( \theta \in \mathbb{R}^d \), the relevance vector.

ARD kernels are naturally interpretable. Take \( \mathcal{H}_\theta \) as the RKHS associated with an ARD kernel \( k_\theta \). Each entry \( \theta_i \) of the relevance vector is the reciprocal of the lengthscales of variation of the \( i \)-th coordinate of functions \( f \in \mathcal{H}_\theta \); that is, the smaller the \( \theta_i \), the slower the functions of \( \mathcal{H} \) will change in the direction of the \( i \)-th feature. In the extreme case \( \theta_i = 0 \), these functions will not depend on the \( i \)-th feature. Hence, removing the influence of a feature requires shrinking to zero the corresponding relevance parameter.

In the Bayesian learning setting, this shrinkage is accomplished by giving \( \theta \) a hyper-prior and maximizing the Type-II marginal likelihood with respect to \( \theta \) \([5, 12]\), which naturally leads to sparse \( \theta \). In contrast, within the empirical risk minimization (ERM) setting, filtering out the irrelevant features might improve the method’s predictive performance by aligning the data representation, embodied by the kernel, with the data itself \([10, 9]\). Therefore, including \( \theta \) as an additional parameter in the empirical risk minimization should sparsify \( \theta \) and underscore pertinent features to the given predictive task. Crucially, in the context of ERM, the accuracy of predictions indicates how trustworthy the interpretation is: solutions with lower errors are likely to match the data better and how it’s represented. As a result, striving for the lowest prediction error and reliable interpretation are naturally linked objectives in this scenario.

Nevertheless, the definition of ARD kernels in (10) does not rule out dependence on terms involving more than one feature. For instance, the kernel \( k_\theta : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R} \) with law

\[ k_\theta(x, x') = e^{-\left[ \theta_1^2|x_1-x'_1|^2 - 2\theta_1\theta_2(x_1-x'_1)(x_2-x'_2) + \theta_2^2|x_2-x'_2|^2 \right]} \]

is a valid ARD kernel. However, if features are fully redundant, i.e. \( x_1 = x_2 \) and \( x'_1 = x'_2 \), then

\[ k_\theta(x, x') = e^{-[(\theta_1 - \theta_2)^2|x_1-x'_1|^2]} \]

Yet, since features are equally important, we expect \( \theta_1 = \theta_2 \). This choice would render the kernel constant, implying that none of the features are relevant. Product ARD kernels avoid this degenerate scenario by isolating the influence of each feature in the kernel value.
Definition 2 (Product ARD kernel). A function $k_\theta : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a product ARD kernel if

$$k_\theta(x, x') = \prod_{i=1}^{d} k_{\theta_i}(x_i, x'_i) = \prod_{i=1}^{d} G[\theta_i(x_i - x'_i)],$$

(11)

where each $k_{\theta_i}$ is an ARD kernel of the form (10) with relevance $\theta_i \in \mathbb{R}_+^+$. We remark that limiting our attention to product ARD kernels is not restrictive. The most commonly used kernels in the literature are product ARD kernels constrained by the condition that $\theta_i = \theta_j = 1/\sigma^2$ for all $i, j \in [d]$, where $\sigma^2$ is a bandwidth parameter. As we will see, product ARD kernels are crucial to ensure RFFNet’s computational efficiency.

3. Overview of RFFNet

In this section, we introduce RFFNet, our large-scale and interpretable kernel method based on random Fourier features for product ARD kernels, and carefully initialized first-order optimization methods. First, we discuss two essential properties of product ARD kernels. Then, we discuss RFFNet objective function and how we train RFFNet. We conclude by introducing an algorithm for variable selection based on thresholding the relevance vector output by an RFFNet instance. All proofs are deferred to Appendix A.

3.1. Random Fourier features for product ARD kernels

Two key properties of ARD kernels render them particularly well-suited to a random Fourier features approximation. The first property is a simple consequence of the pairing of features and relevances in general ARD kernels. In fact, let $k_\theta$ be an ARD kernel, then

$$k_\theta(x, x') = G[\theta \circ (x - x')] = G[\mathbf{1}_d \circ (\theta \circ x - \theta \circ x')] = k_{1_d}(\theta \circ x, \theta \circ x').$$

Thus, any random Fourier features map $z : \mathbb{R}^d \rightarrow \mathbb{R}^s$ that approximates $k_{1_d}$ can be equally used to approximate $k_\theta$ without compromising the statistical properties of the approximation. The next proposition formalizes this.

Proposition 1. Let $k_\theta : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ be an ARD kernel. Let $z : \mathbb{R}^d \rightarrow \mathbb{R}^s$, $s \geq 1$, be the random Fourier features map for $k_{1_d}$, the isotropic version of $k_\theta$. Then,

$$\hat{k}_\theta(x, x') = z(\theta \circ x)^\top z(\theta \circ x'),$$

(12)

is an unbiased estimator of $k_\theta(x, x')$. 

11
In practice, the most significant consequence of this proposition is that to approximate an ARD kernel \( k_\theta \), we need to sample the spectral density of \( k_{1_d} \), which is independent of the relevances. The relevance vector can be introduced later as a scaling of the features. In light of the discussion in Section 2.3, this greatly reduces the computational cost of including the relevance vector in empirical risk minimization procedures, as changes in the relevance during the minimization do not require resampling the entire pool of random features. This property has been already highlighted in Gregorová et al. [18].

The second property, intrinsic to product ARD kernels, stems from the form of their spectral density, characterized in the following proposition.

**Proposition 2.** Let \( k_\theta : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) be a product ARD kernel. Then, its spectral density \( p_\theta \) satisfies

\[
p_\theta(\omega_1, \ldots, \omega_d) = \prod_{i=1}^{d} \frac{1}{|\theta_i|} p\left(\frac{\omega_i}{\theta_i}\right),
\]

where \( p(\cdot) \) is the spectral density of \( k_1 \), that is \( k_{\theta_i} \) with \( \theta_i = 1 \).

This result shows that each relevance \( \theta_i \) is a scale parameter of the corresponding spectral density \( p_i \). As such, the sign of each \( \theta_i \) is irrelevant. When including \( \theta \) in RFFNet’s objective function, we can treat the relevances as an unconstrained parameter\(^2\). In contrast, other methods restrict the values of \( \theta \) to specific subsets of \( \mathbb{R}^d_+ \), such as simplices [18, 8] and product of intervals [1]. Imposing these constraints requires projecting \( \theta \) each time it is updated, hindering its interpretation of a vector of lengthscales of variation, which, by definition, should be unbounded.

### 3.2. The objective function and training

**RFFNet** builds on the properties above of product ARD kernels to add a layer of interpretability to kernel methods with minimal computational overhead. For a given user-specified product ARD kernel \( k_\theta \) whose spectral

---

\(^2\)To clarify with an analogy: the standard error is a scale parameter of the normal distribution. Our assertion is analogous to stating that if \( Z \sim \mathcal{N}(0, 1) \) and \( \sigma > 0 \), then both \( \sigma Z \) and \( -\sigma Z \) are distributed as \( \mathcal{N}(0, \sigma^2) \).
density can be sampled, RFFNet solves

\[
\min_{\beta \in \mathbb{R}^s, \theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, z(\theta \circ x_i)^\top \beta) + \lambda \|\beta\|^2_2,
\]

(14)

where \(z : \mathbb{R}^d \to \mathbb{R}^s\) is the random Fourier features map corresponding to \(k_{1,d}\), \(\lambda > 0\) is the regularization parameter, and the loss function \(\ell\) is taken as the squared loss in regression and the cross-entropy loss in classification. Notice that this is a minor modification of (8) incorporating the outcomes of Propositions 1 and 2: the random features map is sampled only at initialization, as it does not depend on \(\theta\), and \(\theta\) is included in the objective as an unconstrained optimization variable.

RFFNet’s objective function (14) is highly non-convex due to the oscillatory behavior of the random features map and the parity symmetry with respect to \(\theta\). Even if the objective might exhibit a favorable optimization landscape with “natural” input distributions, as discussed in Appendix G, the landscape with real-world data, notably in the small sample setting, is affected by these random features’ oscillations. Nevertheless, for the aforementioned loss functions, the objective has Lipschitz continuous gradients in each block \((\beta, \theta)\) of coordinates. This regularity suggests we solve (14) using carefully initialized first-order optimization methods. We give a meta-algorithm describing how RFFNet is trained in Algorithm 1. Subsequently, we describe the default SAMPLE, INITIALIZE, and OPTIMIZE steps. Importantly, these defaults were only established after analyzing the ablation studies in Appendix D.

**Algorithm 1** Default training of RFFNet

1: function Fit(training data \(\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n\), number of random features \(s\), regularization strength \(\lambda\), max optimization iterations \(T\), learning rate \(\eta\), batch size \(B\), validation fraction \(\xi\), patience \(K\))
2: Sample \(\{b_i\}_{i=1}^s \sim \text{Unif}(0, 2\pi)\)
3: \(\{\omega_i\}_{i=1}^s \leftarrow \text{SAMPLE}(s)\)
4: Use \(\{(\omega_i, b_i)\}_{i=1}^s\) to construct \(z : \mathbb{R}^d \to \mathbb{R}^s\) as in (5).
5: \(\theta\)\text{\init} \leftarrow \text{INITIALIZE}(\mathcal{D})\) \hfill \(\triangleright\) See (15)
6: \((\beta\*, \theta\*) \leftarrow \text{OPTIMIZE}(\mathcal{D}, z, \theta\text{\init}, T, \lambda, \eta, B, \xi, K)\) \hfill \(\triangleright\) See Algorithm 2
7: return regressor or scorer \(f\* (x) = z(\theta\* \circ x)^\top \beta\*\).
8: end function
In the default setting, the Sample step samples from the spectral density of the Gaussian isotropic kernel, i.e. \( \omega_i \sim N(0, I_d) \) in step 3 of Algorithm 1. With this, according to Proposition 1, the random features map \( z \) of step 4 is such that \( \hat{k}_\theta(x, x') \) is an unbiased estimator of the Gaussian ARD kernel

\[
\hat{k}_\theta(x, x') = \exp\left[-\frac{1}{2}\|\theta \circ (x - x')\|_2^2\right].
\]

The relevance vector is initialized in the Initialize step as

\[
\theta_{\text{init}} = \frac{1}{d} \left( \max\{x_i : i \in [n]\} - \min\{x_i : i \in [n]\} \right), \tag{15}
\]

where the max and min are element-wise, i.e. they act in each entry of the vectors \( x_i \). Intuitively, assuming the training data is standardized, features with the greatest range (i.e., the difference between the maximum and minimum) should vary in smaller lengthscales and thus be initialized with greater relevances.

For the Optimize step, we use the Adam optimizer \([29]\) with mini-batches, treating \( \theta \) and \( \beta \) as a single block of coordinates. The \( \ell_2 \) penalty on \( \beta \) in (14) is applied by composing the iterates of \( \beta \) with the proximal operator of the \( \ell_2 \) norm

\[
\text{prox}_{\lambda \cdot \|\cdot\|_2^2} \eta(\beta) = \frac{\beta}{1 + 2\lambda \eta}, \tag{16}
\]

where \( \eta \) is the learning rate. We used proximal operators to allow users to easily implement and experiment with other regularization terms on \( \beta \) and \( \theta \). Finally, we used early stopping and checkpointing as training heuristics. Details of the optimization procedure are described in Algorithm 2.

As for the implementation, we remark that solving (14) is equivalent to solving a regularized loss minimization problem in the hypothesis space \( \mathcal{F} = \{ x \mapsto \beta^T z(\theta \circ x) : \beta \in \mathbb{R}^s, \theta \in \mathbb{R}^d \} \). Since \( z_i(x) = \sqrt{2/s} \cos(\omega_i^\top x + b_i) \), \( \mathcal{F} \) is the space of one hidden-layer neural networks with cosine activation functions. We use this equivalence to implement RFFNet as a neural network model in PyTorch and leverage PyTorch’s efficient automatic differentiation engine to compute the gradients in Algorithm 2.

3.3. Thresholding for variable selection

The relevance vector \( \theta \) output by RFFNet is not exactly sparse. Even if entries corresponding to irrelevant features are shrunk during training, they
Algorithm 2 Minimizing RFFNet’s objective

1: function Optimize($\mathcal{D}, \theta_{\text{init}}, T, \lambda, \eta, B, \xi, K$)
2: Validation split: $\mathcal{D}' \leftarrow \xi$ fraction of samples of $\mathcal{D}$.
3: $\theta \leftarrow \theta_{\text{init}}$. ▷ See (15).
4: for $i \in \{1, \ldots, T\}$ do
5: Sample batches of size $B$ from $\mathcal{D} \setminus \mathcal{D}'$.
6: for each batch do
7: $(\beta, \theta) \leftarrow \text{Adam}(\beta, \theta, \eta)$.
8: $\beta \leftarrow \text{prox}_{\lambda \|\|_{2,1}}(\beta)$. ▷ See (16).
9: end for
10: Evaluate the loss of $f^{(i)}(x) = \mathbf{z}(\theta^{(i)} \circ x)^\top \beta^{(i)}$ on $\mathcal{D}'$.
11: Store $(\beta^{(i)}, \theta^{(i)})$.
12: if loss on $\mathcal{D}'$ reduced in every previous $K$ iterations then
13: Stop training. ▷ Early stopping.
14: end if
15: end for
16: $(\beta^*, \theta^*) \leftarrow (\beta^{(i)}, \theta^{(i)})$ with the smallest loss on $\mathcal{D}'$. ▷ Checkpointing.
17: return $(\beta^*, \theta^*)$
18: end function

are not forced to be identically zero. This does not oppose our purpose of creating an interpretable kernel method. We expect the relevances to be powerful tools for feature engineering and model interpretation, giving insights about the relative significance of features within the given predictive task. Nevertheless, it is natural to ask if thresholding the entries of the relevance vector can yield a valid variable selection rule, capable of identifying the active features while controlling the false discovery rate.

To this end, we introduce TopK, a simple procedure to automatically include/exclude variables from a trained RFFNet model based on the model’s scores (under a suitable scoring function) on a held-out dataset. We described TopK in Algorithm 3. The scoring rule in Algorithm 3 can be chosen as the negative squared error loss for regression and the AUC for classification problems.
Algorithm 3 Variable selection using RFFNet’s relevances

1: function TopK(selection dataset \( \mathcal{D}_s \), fitted RFFNet model \( f^* \) with relevance vector \( \theta^* \), scoring function \( r : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \))
2: \hspace{1em} Order the relevances in descending order.
3: \hspace{1em} for \( i \in [d] \) do
4: \hspace{2em} \( \theta_{\text{thresh}} \leftarrow [\theta^*_0, \theta^*_1, \ldots, \theta^*_{(d-i)}, \theta^*_{(d-1)}, 0, \ldots, 0] \). \( \triangleright \) Include the \( i \) greatest relevances
5: \hspace{2em} \( \theta^* \leftarrow \theta_{\text{thresh}}. \)
6: \hspace{2em} Score \( f^* \) with the new \( \theta^* \) in \( \mathcal{D}_s \).
7: \hspace{2em} Store the value of the score, say \( r_i \) and the vector \( \theta_{\text{thresh}} \).
8: \hspace{1em} end for
9: \hspace{1em} Get the relevance vector \( \theta^*_{\text{thresh}} \) associated with the highest score.
10: \( \theta_s \leftarrow \theta^*_{\text{thresh}}. \)
11: return \( \theta_s \).
12: end function

4. Results

In this section, we report the results of the evaluation of RFFNet in simulated and real-world regression and classification datasets. We include here only predictive metrics (mean squared error for regression and AUC for classification), leaving results about RAM usage and run time to Appendix B.

Benchmarks. For regression, we compared RFFNet to kernel ridge regression (KRR) with an isotropic Gaussian kernel, approximate kernel ridge regression with Fastfood \[34\] and Nyström \[65\, 64\, 13\] feature maps for the isotropic Gaussian kernel, Gaussian Processes Regression (GPR) \[64\] with an ARD Gaussian kernel, EigenPro regressor \[40\] with a Gaussian isotropic kernel, and to Sparse Random Fourier Features (SRFF) \[18\] with a Gaussian kernel. For classification, we compared RFFNet to logistic regression with Fastfood and Nyström feature maps for the isotropic Gaussian kernel, and to the EigenPro classifier with a Gaussian isotropic kernel. Among these benchmarks, only SRFF and GPR output feature relevances.

Datasets. We compared the above benchmarks against RFFNet in 4 simulated regression datasets and 2 simulated classification datasets. For regression, the datasets gse1 and gse2 originate from Gregorová et al. \[18\], while jse2 and jse3 originate from Jordan et al. \[25\]. These datasets were used to evaluate the
ability of kernel-based algorithms to identify active features in the simulated regression function. For classification, the classification and moons are native to scikit-learn [45], where they are widely used to evaluate the performance of classification algorithms.

To complete our assessment, we evaluated RFFNet in 8 real-world datasets (4 regression and 4 classification datasets). These datasets comprise a range of sample sizes \( n \) and dimensionality \( d \) that allow us to pinpoint the settings where RFFNet is expected to perform better. A complete description of the simulated and real-world datasets and the sources to download the real-world datasets is provided in Appendix C. We also provide code to preprocess these datasets and fully reproduce the results in this section.

**Hyperparameters.** RFFNet was evaluated using its default configuration with a regularization parameter of \( \lambda = 10^{-4} \) motivated by the analysis in Appendix D. Benchmarks were evaluated using their default hyperparameters. For a fair comparison, we do not optimize (e.g., via cross-validation) the hyperparameters of RFFNet nor any of the benchmarks. Notwithstanding, we strongly encourage users to include hyperparameter tuning methods in their data analysis pipelines.

### 4.1. Simulations

First, we compared all benchmarks in the simulated datasets. We fixed the sample size as \( n = 5000 \) and generated 10 replicas of each dataset. We report the average and one standard error of the performance metrics in these replicas. In Table 1, we notice that RFFNet outperforms all benchmarks by a large margin in the regression datasets. Figure 2 helps explain this: RFFNet correctly identifies the active features in each dataset, removing the spurious influence of the irrelevant ones. Table 2 shows that all methods have comparable performance for the classification datasets. This occurs because both datasets have decision boundaries that are learnable using the nonlinear feature maps used in the benchmark models. Finally, the run-time and RAM measurements in Appendix F show that fitting the relevances did not incur additional computational overhead for RFFNet, which outplayed GPR and SRFF.

### 4.2. Variable selection

We test whether TopK (see Algorithm 3 yields a variable selection rule with a high True Discovery Rate (TDR) that controls the False Discovery
Table 1: Mean squared error on the hold-out sample for the simulated regression datasets. RFFNet outperformed all baselines. The best performances for each metric are displayed in boldface.

| Model      | gse1        | gse2        | jse2        | jse3        |
|------------|-------------|-------------|-------------|-------------|
| EigenPro   | 0.100 ± 0.004 | 4.489 ± 0.287 | 2.652 ± 1.759 | 0.037 ± 0.005 |
| Fastfood   | 0.082 ± 0.004 | 5.112 ± 0.285 | 10.381 ± 2.100 | 0.055 ± 0.011 |
| GPR        | 0.081 ± 0.004 | 5.134 ± 0.269 | 26.718 ± 4.239 | 0.661 ± 0.036 |
| Kernel Ridge | 0.091 ± 0.004 | 4.085 ± 0.260 | 2.018 ± 1.585 | 0.032 ± 0.005 |
| Nyström    | 0.083 ± 0.004 | 5.084 ± 0.288 | 7.031 ± 2.211 | 0.147 ± 0.010 |
| RFFNet     | **0.073 ± 0.004** | **1.865 ± 0.222** | **1.359 ± 1.386** | **0.012 ± 0.002** |
| SRFF       | 0.084 ± 0.004 | 5.509 ± 0.248 | 29.996 ± 13.526 | 0.227 ± 0.052 |

Figure 2: Relevances for a realization of the gse1, gse2, jse2 and jse3 simulated datasets. The labels on the x-axis indicate the active features for each dataset (see Appendix C). The relevances output by RFFNet peak exactly on the active features of these datasets.

Table 2: Mean classification AUC on the hold-out sample for the simulated classification datasets. On average, RFFNet outperforms all baselines. The best performances for each model are displayed in boldface. Since all models considered fit non-linear decision boundaries, they perfectly fit the simple moons dataset.

| Model      | classification | moons |
|------------|----------------|------|
| EigenPro   | 0.959 ± 0.023  | 1.00 ± 0.00 |
| Fastfood   | 0.960 ± 0.018  | 1.00 ± 0.00 |
| Nyström    | 0.964 ± 0.020  | 1.00 ± 0.00 |
| RFFNet     | **0.98 ± 0.016** | **1.00 ± 0.00** |

Rate (FDR). We used the negative of the squared error loss as the scoring function. In Figure 3, we tested TopK for increasing sample sizes in the
\textbf{gse1} and \textbf{gse2} datasets. We report the average of each metric, along with the standard error, over 10 replicas of each dataset. Observe that increasing the sample size improves the identification of the relevant features, as the mean TDR increases while the variance decreases. In addition, the FDR rate is kept low in all situations and monotonically reduces as the sample size increases.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure3}
\caption{The empirical False/True Discovery Rates (FDR/TDRs) for the simulated datasets, \textbf{gse1} and \textbf{gse2}. Increasing the sample size controls the FDR and improves the identification of active features.}
\end{figure}

\subsection*{4.3. Real-world datasets}

Finally, we report the results of applying \textbf{RFFNet} to real data. We compared all benchmarks via 10-fold cross-validation and reported the average and standard error of the prediction metric in the held-out sample. In Table 3, we show the results for the regression datasets. We observe that \textbf{RFFNet} substantially outperforms SRFF, which did not converge for 3 out of 4 datasets used, and GPR, which incurred large errors for most tasks. \textbf{RFFNet} performed similarly to all benchmarks in the \textbf{abalone} dataset and outperformed all methods in the \textbf{compact} dataset. In the \textbf{powerplant} dataset, we believe that the low number of features discouraged the convergence of \textbf{RFFNet} to a lower error solution. The results in Appendix F show that
RFFNet maintained both a small run-time and memory usage due to the random Fourier features approximation. Table 4 summarizes the result for the classification datasets. RFFNet significantly outperformed the benchmarks in the amazon and higgs datasets. This is the result of correctly identifying relevant features for the predictive task, as depicted in Figure 4. In the amazon dataset, where the objective is to predict the sentiment (positive or negative) of a food review, the most relevant features intuitively have strong correlations with positive sentiments. In the higgs dataset, where the goal is to distinguish between physical processes leading to the production of Higgs bosons to background processes, the most relevant features are the high-level covariates created in Baldi et al. [4] to have greater distinguishing power between these processes. Importantly, as discussed in the supplementary results in Appendix F, RFFNet superior’s predictive performance in these datasets do not come at prohibitive run-time and RAM usage.

Table 3: Mean squared error on the hold-out sample for the real-world regression datasets. The best performances for each model are displayed in boldface. OOM: out-of-memory (see Appendix B). FC: failed to converge.

| Model   | abalone     | compact     | powerplant | yearprediction |
|---------|-------------|-------------|------------|----------------|
| EigenPro | 4.449 ± 0.402 | 42.546 ± 5.701 | 17.664 ± 0.888 | OOM            |
| Fastfood | 4.507 ± 0.347 | 41.995 ± 4.924 | 17.031 ± 1.06 | 88.156 ± 1.043 |
| GPR     | 4.904 ± 0.391 | 540.443 ± 37.107 | 41.79 ± 4.81 | OOM            |
| Kernel Ridge | 4.447 ± 0.406 | 35.3 ± 5.669 | 15.555 ± 0.959 | OOM            |
| Nyström | 4.408 ± 0.396 | 38.43 ± 10.196 | 15.092 ± 0.939 | 81.421 ± 1.106 |
| RFFNet  | 4.851 ± 0.417 | 9.765 ± 1.254 | 29.762 ± 1.636 | 95.367 ± 1.385 |
| SRFF    | 5.267 ± 0.419 | FC          | FC          | FC             |

Table 4: Mean classification AUC on the hold-out sample for the real-world classification datasets. RFFNet performs better in large-scale datasets. The best performances for each model are displayed in boldface.

| Model   | a9a         | amazon     | higgs      | w8a         |
|---------|-------------|------------|------------|-------------|
| EigenPro | 0.867 ± 0.006 | OOM        | OOM        | 0.956 ± 0.010 |
| Fastfood | 0.899 ± 0.004 | 0.867 ± 0.005 | 0.701 ± 0.004 | 0.947 ± 0.014 |
| Nyström | 0.894 ± 0.004 | 0.825 ± 0.007 | 0.723 ± 0.002 | 0.967 ± 0.008 |
| RFFNet  | 0.890 ± 0.004 | 0.922 ± 0.002 | 0.774 ± 0.002 | 0.949 ± 0.008 |
(a) Relevances for the top 7 most relevant (in red) and 7 least relevant (in blue) features for the *amazon* dataset. Notice how the relevant features are, indeed, more informative to evaluate the sentiment of a food review.

(b) Relevance for the top 7 most relevant (in red) and 7 least relevant (in blue) features for the *higgs* dataset. The most relevant features are exactly the high-level features in Baldi et al. [4] that better distinguish Higgs-related events from background ones.

Figure 4: Relevance patterns for the *amazon* (top) and *higgs* (bottom) datasets. RFFNet associates greater relevance to features that are intuitively (for the *amazon* dataset) and scientifically relevant (for the *higgs* dataset).
5. Final remarks

In this paper, we introduced RFFNet, a new kernel method that yields interpretable prediction rules for large-scale datasets. RFFNet reliance on carefully designed random Fourier features and stochastic first-order methods grants low run-time and memory usage even for datasets exceeding millions of samples. Additionally, RFFNet’s defaults are established through comprehensive ablation studies, providing users with a sensible starting point for applying the method to their data.

Moreover, automatic relevance determination kernels bridge the gap between kernel methods’ nonparametric, high-dimensional nature and the practitioner’s necessity for interpretable solutions based on the original features. The kernel relevances give users a reliable metric of feature importance, revealing the presence of irrelevant covariates and underscoring that active features may have distinct length scales of variation.

We validated our approach in a series of experiments, wherein RFFNet exhibited a performance on par with other kernel methods, notably in the large sample scenario with many inactive features. In the simulated datasets, RFFNet showed promising predictive performance and precise ability to identify the relevant features. For the real-world datasets, we observed that RFFNet consistently outperforms or matches the performance of many kernel-based learning algorithms.

Many open venues for future work remain. On the theory side, does the inclusion of the relevances in the empirical risk minimization procedure drastically increase the problem’s sample complexity? Furthermore, can we use the relevances during training to reduce the number of parameter updates, akin to active set methods in sparse linear regression? If possible, this could improve RFFNet scalability even further. On the applications side, we want to leverage RFFNet layered structure to incorporate it in models like neural networks, aiming for interpretable prediction on structured inputs such as images and tensors.

Acknowledgements

We thank Carlos Pagani Zanini, Eduardo Fonseca Mendes, Júlio Michael Stern, and Roberto Imbuzeiro for helpful suggestions during the early stages of this work. MPO thanks Flávia Castro Motta for attentively reviewing the paper. MPO was supported through grant 2021/02178-8, São Paulo Research
Foundation (FAPESP). RI was supported through grants 309607/2020-5, 422705/2021-7, 305065/2023-8 Brazilian National Counsel of Technological and Scientific Development (CNPq), and grants 2019/11321-9, 2023/07068-1, São Paulo Research Foundation (FAPESP).

Appendix A. Proofs

**Lemma 1** (Relation between spectral densities of $k_\theta$ and $k_{1_d}$). Let $p_\theta$ be the spectral density of the PSD shift-invariant kernel $k_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $p$ be the spectral density of $k_{1_d}$ with $\theta = 1_d$. Then,

$$p_\theta(\omega) = \frac{1}{|\theta_1 \cdots \theta_d|} p\left(\frac{1}{\theta} \omega\right), \quad (A.1)$$

where $\omega \circ 1/\theta = (\omega_1/\theta_1, \ldots, \omega_d/\theta_d)$.

**Proof.** The density of the spectral measure of $k_\theta$ is, by Bochner’s Theorem,

$$p_\theta(\omega) = \frac{1}{(2\pi)^d} \int e^{-i\omega^\top(x-x')} k_\theta(x, x') d\delta$$

$$= \frac{1}{(2\pi)^d} \int e^{-i\omega^\top \delta} G(\theta \circ \delta) d\delta,$$

where $\delta = x - x'$. Define the new variable $\delta' = \theta \circ \delta = (\theta_1 \delta_1, \ldots, \theta_d \delta_d)$, then, by the multivariate change of variables theorem, we get

$$p_\theta(\omega) = \frac{1}{(2\pi)^d} \int e^{-i(\omega_1^{1/\theta})^\top \delta'} G(\delta') \frac{1}{|\theta_1 \cdots \theta_d|} d\delta'$$

$$= \frac{1}{|\theta_1 \cdots \theta_d|} \frac{1}{(2\pi)^d} \int e^{-i(\omega_1^{1/\theta})^\top \delta'} G(\delta') d\delta'$$

$$= \frac{1}{|\theta_1 \cdots \theta_d|} p\left(\frac{1}{\theta} \omega\right). \quad (A.2)$$

with $p(\cdot)$ the spectral measure of the kernel with $\theta = 1_p$.

Proof of Proposition 1. By \(A.1\), if we sample $\omega \sim p(\cdot)$, then $\theta \circ \omega \sim p_\theta(\cdot)$. Now, by Bochner’s Theorem, with $\omega' \sim p_\theta(\cdot)$,

$$k_\theta(x, x') = \mathbb{E}_{\omega'}[\cos \omega'^\top(x - x')]$$

$$= \mathbb{E}_\omega[\cos(\theta \circ \omega)^\top(x - x')]$$

$$= \mathbb{E}_\omega[\cos \omega^\top(\theta \circ (x - x'))],$$

23
where $\omega \sim p(\cdot)$. Let $z : \mathbb{R}^d \to \mathbb{R}^s$ be the RFF for $k_\theta$ with $\theta = 1_d$, then

$$k_\theta(x, x') = \mathbb{E}_{\omega, \theta}[z(\theta \circ x)^\top z(\theta \circ x')]$$

which shows that

$$\hat{k}_\theta(x, x') = z(\theta \circ x)^\top z(\theta \circ x')$$

is an unbiased estimator of $k_\theta(x, x')$. \qed

**Proof of Proposition 2.** Since $k_\theta$ is a product ARD kernel

$$k_\theta(x, x') = \prod_{i=1}^d k_{\theta_i}(x_i, x'_i) = \prod_{i=1}^d G[\theta_i(x_i - x'_i)],$$

the spectral density of $p_\theta$ reads

$$p_\theta(\omega) = \frac{1}{(2\pi)^d} \int e^{-i\omega^\top(x-x')}k_\theta(x, x')d\delta$$

$$= \prod_{i=1}^d \frac{1}{2\pi|\theta_i|} \int e^{-i\frac{\omega_i}{\theta_i}\delta'_i}G(\delta'_i)d\delta'_i,$$

where $\delta'_i = \theta_i(x_i - x'_i)$. As each term in the product is the spectral density of the one-dimensional ARD kernels $k_{\theta_i}$, we conclude that

$$p_\theta(\omega) = \prod_{i=1}^d \frac{1}{|\theta_i|} p\left(\frac{\omega_i}{\theta_i}\right),$$

where $p$ is the spectral density of any $k_{\theta_i}$ with $\theta_i = 1$. \qed

**Appendix B. Experimental details**

**Appendix B.1. Code availability**

Code to reproduce the experiments is available at

[https://github.com/mpotto/rffnet](https://github.com/mpotto/rffnet)

**Appendix B.2. Computing infrastructure**

Experiments were run with an Intel Core i7-8700 CPU with 3.20GHz, 6 cores, 12 threads, and 54 GB of RAM. GPUs were not used to facilitate the reproduction of the results. We use the OOM (out-of-memory) to indicate that a particular experiment exceeded the memory budget.
Appendix C. Datasets

Simulated datasets were created as follows:

- **gse1 [18]**: features are sampled as $X \sim \mathcal{N}(0, I_d)$, with $d = 18$. The noise variables are sampled as $\varepsilon \sim \mathcal{N}(0, \sigma = 0.1)$. The response is:
  $$y = \sin[(x_1 + x_3)^2] \sin(x_7 x_8 x_9) + \varepsilon.$$

- **gse2 [18]**: features are sampled as $X \sim \mathcal{N}(0, I_d)$, with $d = 100$. The noise variables are sampled as $\varepsilon \sim \mathcal{N}(0, \sigma = 0.1)$. The response is:
  $$y = \log[(x_{11} + x_{12} + x_{13} + x_{14} + x_{15})^2] + \varepsilon.$$

- **jse2 [25]**: features are sampled as $X \sim \mathcal{N}(0, \Sigma)$, where $\Sigma$ is the $d \times d$ covariance matrix with $d = 10$ and $\Sigma_{ij} = 0.5|^{i-j}|$. The noise variables are sampled as $\varepsilon \sim \mathcal{N}(0, \sigma = 0.1)$. The response is
  $$y = x_1^3 + x_2^3 + \varepsilon.$$

- **jse3 [25]**: features are sampled as $X \sim \mathcal{N}(0, I_d)$, with $d = 10$. The noise variables are sampled as $\varepsilon \sim \mathcal{N}(0, \sigma = 0.1)$. The response is
  $$y = x_1 x_2 + \varepsilon.$$

- **classification**: this is a default binary classification synthetic dataset in scikit-learn [45].

- **moons**: this is a default binary classification synthetic dataset in scikit-learn [45].

Table C.5 gives a basic description of the real-world datasets, which were retrieved from the following sources:

- **abalone**: [https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/)

- **compact**: [https://www.dcc.fc.up.pt/~ltorgo/Regression/compact.tar.gz](https://www.dcc.fc.up.pt/~ltorgo/Regression/compact.tar.gz)
Table C.5: Description of real-world datasets: sample size of train and test splits, and number of features.

| Dataset   | Train ($n$) | Test ($n_{test}$) | Features ($d$) |
|-----------|-------------|-------------------|----------------|
| abalone   | 3759        | 419               | 8              |
| compact   | 7372        | 820               | 21             |
| powerplant| 43056       | 4784              | 4              |
| yearprediction | 463810   | 51535             | 90             |
| a9a       | 43957       | 4885              | 123            |
| w8a       | 58230       | 6470              | 300            |
| amazon    | 327753, 36418 | 3000             |                |
| higgs     | 9900000     | 1100000           | 28             |

- powerplant: [https://archive.ics.uci.edu/dataset/294/combined+cycle+power+plantp](https://archive.ics.uci.edu/dataset/294/combined+cycle+power+plantp)
- yearprediction: [https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/)
- a9a: [https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary)
- amazon: [https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews](https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews)
- higgs: [https://archive.ics.uci.edu/dataset/280/higgs](https://archive.ics.uci.edu/dataset/280/higgs)
- w8a: [https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary)

Scripts to download and preprocess these datasets are available in the experiments/data folder.

Appendix D. Ablation studies

This section evaluates the impact of fixing the sampling, initialization, and optimization methods in Algorithm[1]. In addition, we seek to understand the influence of regularization on the model. We use the evaluation in this
section to establish sensible defaults for RFFNet. The performance of RFFNet with these default settings is presented in Section 4. Simulated datasets used in this section are described in Appendix C. Importantly, all ablation studies were performed only in the regression setting using the squared error loss.

Appendix D.1. Sampling

We tested three spectral density samplers: Gaussian, Laplace, and Cauchy. Sampling from each of these corresponds to producing an approximation of the Gaussian, Cauchy, and Laplace kernels, respectively, as outlined in Table D.6. In this experiment, we fixed the initialization scheme as the Constant strategy described in Appendix D.2 and the PALM-Adam algorithm described in Appendix D.3. The experiment is repeated 20 times. We report the average (solid lines) and 1 standard error bounds (shaded region). In Figure D.5, we see that sampling from the Gaussian and Laplace distributions often led to the same final validation loss. However, the convergence with the Gaussian sampler is usually faster than the Laplace density. For this reason, we choose the Gaussian spectral density as the default for RFFNet.

Figure D.5: Validation squared error loss as a function of the training epoch for the sampling strategies. Using the Gaussian sampling scheme led to faster convergence in all cases.
Table D.6: Usual shift-invariant PSD kernels and the densities of their corresponding spectral measures.

| Sample      | $k_{1_d}(x, x')$                                                                 | $p(\omega)$                                                                 |
|-------------|----------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| Gaussian    | $\exp\left(-\frac{1}{2}\|x - x'\|_2^2\right)$                                 | $(2\pi)^{-\frac{d}{2}}\exp\left[-\frac{\|\omega\|_2^2}{2}\right]$          |
| Laplace     | $\exp[-\|x - x'\|_1]$                                                           | $\prod_{i=1}^{d} \frac{1}{\pi(1+\omega_i^2)}$                              |
| Cauchy      | $\prod_{i=1}^{d} \frac{2}{1+(x_i-y_i)^2}$                                       | $\exp[-\|\omega\|_1]$                                                      |

Appendix D.2. Initialization

As the source of the objective function’s non-convexity is the dependence on $\theta$, we focus on initialization strategies only for this parameter. We tested three initialization methods that users can choose from: Constant, Restarter, and Regressor. Given the training sample $\{(x_i, y_i)\}_{i=1}^{n}$, these schemes work as follows:

1. Constant: the relevances are initialized as
   \[
   \theta_{\text{init}} = \theta_\star := \frac{1}{d}\left(\max\{x_i : i \in [n]\} - \min\{x_i : i \in [n]\}\right),
   \tag{D.1}
   \]
   where the max and min are feature-wise, i.e. they act in each entry of the vectors $x_i$ (as mentioned in Section 4).

2. Regressor: we use $\{(x_i, y_i)\}_{i=1}^{n}$ to train a Lasso [60] regressor, whose output is taken as having the form $f^*(x) = \beta \top x$. Then, we initialize $\theta$ with the trained model weights $\theta_{\text{init}} = \beta$.

3. Restarter: the relevances are first sampled as $\theta_{\text{sample}} \sim N(\theta_\star, \frac{1}{d}I_d)$, where $\theta_\star$ was defined in (D.1). Then, the model is optimized for 10 iterations, and the final training loss is recorded. This process is restarted 10 times from $\theta_{\text{sample}}$. $\theta_{\text{init}}$ is assigned to the relevance vector that led to the smallest training loss among the 10 runs.

For this experiment, we fix the optimization method as the PALM-Adam algorithm described in Appendix D.3 and the sampling method as the Gaussian one described in Appendix D.1. The experiment is repeated 20 times. We report the average (solid lines) and 1 standard error bounds (shaded region). In Figure D.6, we plot the validation loss during training for the
simulated regression datasets described in Appendix C. While the Restarter strategy usually gives superior initialization results, characterized by faster error decay in the early stages of training, we opt for the Constant strategy for two primary reasons. Firstly, it is much faster, as it does not require multiple training runs from distinct initializations. Secondly, the number of epochs needed for this strategy to reach the loss plateau first reached by the Restarter one is small. Finally, the Regressor strategy takes much longer to reach the same loss plateau and requires training the Lasso model at initialization, which can introduce additional computational overhead.

![Graph showing validation squared error loss as a function of the training epoch for the initialization strategies.](image)

Figure D.6: Validation squared error loss as a function of the training epoch for the initialization strategies. The Constant method is reliable and faster, avoiding multiple model initializations (Restarter) or training auxiliary models (Regressor).

**Appendix D.3. Optimization**

We tested four first-order stochastic gradient-based optimization algorithms: vanilla Stochastic Gradient Descent (SGD) and the Adam algorithm [29], each with two blocks (alternating minimization) or a single block of coordinates. As described in the main text, the $\ell_2$ penalty on $\beta$ in (14) is applied in either case by composing the iterates with the proximal operator of the $\ell_2$ norm. Because of this, the two block optimization methods are a version of the Proximal Alternating Linearized Minimization (PALM) [7].
while the single block methods are Proximal Stochastic Gradient Descent methods. In this section’s experiments, we did not use early stopping as a training heuristic. We used the sampling method as Gaussian and initialized $\theta$ via the Constant strategy. As before, the experiments are repeated 10 times. We report the average (solid line) and 1 standard error bounds (shaded regions). We show in Figures D.7 the validation loss history during training. Notice that SGD and PALM-SGD exhibited similar performance, as did Adam and PALM-Adam. However, the Adam and PALM-Adam category shows superior performance, converging much faster to a solution in the most challenging datasets (gse1 and gse2). Since Adam is simpler than PALM-Adam, we opt for using Adam with a single block of coordinates.

Figure D.7: Validation squared error loss as a function of the training epoch for optimization algorithms. The Adam and Adam and PALM-Adam algorithms exhibit comparable performance, much superior to the SGD and PALM-SGD counterparts. Since Adam is simpler than PALM-Adam, we choose Adam as the default optimization algorithm for RFFNet.

Appendix E. Effect of regularization strength

Using the default settings of RFFNet, we evaluated the effect of the regularization parameter $\lambda$ on the identification of the relevant features. As there
may be some interaction between $\lambda$ and $\theta$ (both increasing $\lambda$ and shrinking $\theta$ increase the smoothness), we wanted to determine if there existed sensible default values for $\lambda$ that do not impose too much smoothness on the RFFNet predictor, unintentionally preventing the identification of the relevant features. In Figure E.8, we plot the squared error loss in the validation set during training. Notice that setting $\lambda$ too high (e.g., $\lambda \in \{10, 100\}$) precludes the convergence to a low-error solution in most cases. However, all solutions with $\lambda < 1$ attain the same validation loss plateau. What is the impact of choosing $\lambda$ on the relevance pattern identified by RFFNet? We analyzed this question for the gse1 and gse2 datasets in Figures E.9 and E.10. We fix the sample size as $n = 5000$. In both cases, we see that fixing a high $\lambda$ value prevents the identification of the first and third features in gse1 and all the features in gse2. In contrast, setting smaller values of $\lambda$ may increase the variance of the relevance estimates, but does not prevent the identification of relevant features and guarantee convergence to a low validation error solution, as shown in Figure E.8. For these reasons, we have established the default regularization parameter as $\lambda = 10^{-4}$.

![Figure E.8: Validation squared error loss as a function of the training epoch for different values of the regularization parameter $\lambda$. Higher values of $\lambda$ prevent the convergence of RFFNet to a solution with a low validation error.](image-url)
Figure E.9: Relevance pattern for gse1 dataset with $n = 5000$ samples and varying $\lambda$. The highest value of $\lambda$ precludes the identification of the first and third features. Reducing $\lambda$ increases the variance of the $\theta$ estimates but improves the distinction between relevant and irrelevant features.

Appendix F. Additional experiment results

Appendix F.1. Run-time measurements

Table F.7: Run-time (in seconds) for the simulated regression datasets.

| Model   | gse1       | gse2       | jse2       | jse3       |
|---------|------------|------------|------------|------------|
| EigenPro| 3.568 ± 0.076 | 3.485 ± 0.052 | 3.85 ± 0.031 | 3.891 ± 0.065 |
| Fastfood| 0.015 ± 0.001 | 0.019 ± 0.001 | 0.014 ± 0.001 | 0.014 ± 0.001 |
| GPR     | 108.689 ± 0.185 | 132.261 ± 0.143 | 106.098 ± 0.083 | 105.758 ± 0.094 |
| Kernel Ridge | 1.73 ± 0.002 | 1.75 ± 0.003 | 1.743 ± 0.001 | 1.747 ± 0.004 |
| Nyström | 0.009 ± 0.000 | 0.008 ± 0.001 | 0.009 ± 0.00 | 0.01 ± 0.001 |
| RFFNet  | 1.838 ± 0.062 | 4.028 ± 0.008 | 3.761 ± 0.007 | 2.741 ± 0.526 |
| SRFF    | 5.264 ± 2.153 | 34.42 ± 9.395 | 8.98 ± 10.766 | 12.714 ± 8.075 |
Figure E.10: Relevance pattern for gse2 dataset with $n = 5000$ samples and varying $\lambda$. The highest value of $\lambda$ precludes the identification of all relevant features. Reducing $\lambda$ improves the identification of features altogether.

Table F.8: Run-time (in seconds) for the simulated classification datasets.

| Model       | classification | moons        |
|-------------|----------------|--------------|
| EigenPro    | $3.789 \pm 0.112$ | $3.625 \pm 0.108$ |
| Fastfood    | $0.249 \pm 0.038$ | $0.273 \pm 0.012$ |
| Nyström     | $0.189 \pm 0.021$ | $0.091 \pm 0.015$ |
| RFFNet      | $2.805 \pm 0.979$ | $3.904 \pm 0.004$ |

Table F.9: Run-time (in seconds) for the real-world regression datasets. OOM: out-of-memory.

| Model       | abalone         | compact         | powerplant      | yearprediction  |
|-------------|-----------------|-----------------|-----------------|-----------------|
| EigenPro    | $2.036 \pm 0.018$ | $3.218 \pm 0.075$ | $26.016 \pm 0.056$ | OOM             |
| Fastfood    | $0.006 \pm 0.001$ | $0.012 \pm 0.001$ | $0.062 \pm 0.001$ | $3.431 \pm 0.013$ |
| GPR         | $17.864 \pm 0.123$ | $78.957 \pm 0.367$ | $1878.938 \pm 0.454$ | OOM             |
| Kernel Ridge| $0.183 \pm 0.007$ | $0.865 \pm 0.009$ | $65.332 \pm 0.086$ | OOM             |
| Nyström     | $0.004 \pm 0.00$ | $0.008 \pm 0.00$ | $0.035 \pm 0.00$ | $1.806 \pm 0.003$ |
| RFFNet      | $0.607 \pm 0.125$ | $1.707 \pm 0.571$ | $12.563 \pm 1.983$ | $84.319 \pm 15.361$ |
| SRFF        | $15.343 \pm 5.339$ | $2.898 \pm 1.456$ | $71.513 \pm 72.241$ | $265.975 \pm 140.602$ |
Table F.10: Run-time (in seconds) for the real-world classification datasets. OOM: out-of-memory.

| Model  | a9a            | amazon          | higgs           | w8a    |
|--------|----------------|-----------------|-----------------|--------|
| EigenPro | 31.886 ± 0.065 | OOM             | OOM             | 89.453 ± 0.212 |
| Fastfood | 1.659 ± 0.144  | 116.829 ± 21.278 | 1495.365 ± 10.466 | 2.988 ± 0.032  |
| Nyström  | 1.013 ± 0.019  | 13.208 ± 0.676  | 378.041 ± 122.552 | 1.345 ± 0.033  |
| RFFNet   | 3.226 ± 1.118  | 109.041 ± 40.542 | 800.508 ± 310.327 | 4.023 ± 1.625  |

Appendix F.2. Memory measurements

Table F.11: RAM usage (in MB) for the simulated regression datasets.

| Model  | gse1            | gse2            | jse2            | jse3    |
|--------|-----------------|-----------------|-----------------|---------|
| EigenPro | 440.772 ± 4.639 | 493.362 ± 11.506 | 457.034 ± 8.701 | 454.787 ± 8.273 |
| Fastfood | 443.818 ± 6.483 | 453.142 ± 6.608 | 441.748 ± 6.308 | 441.777 ± 6.313 |
| GPR     | 897.864 ± 76.904 | 951.018 ± 86.28 | 894.622 ± 76.644 | 893.975 ± 76.439 |
| Kernel Ridge | 434.895 ± 6.001 | 477.738 ± 9.359 | 437.968 ± 4.703 | 438.134 ± 4.582 |
| Nyström  | 442.928 ± 4.844 | 459.605 ± 7.743 | 443.193 ± 5.298 | 446.374 ± 7.932 |
| RFFNet   | 434.291 ± 7.541 | 463.314 ± 10.797 | 436.339 ± 4.793 | 435.628 ± 4.523 |
| SRFF     | 686.998 ± 62.039 | 637.538 ± 72.795 | 677.287 ± 63.376 | 659.675 ± 65.885 |

Table F.12: RAM usage (in MB) for the simulated classification datasets.

| Model  | classification       | moons          |
|--------|----------------------|----------------|
| EigenPro | 447.761 ± 3.525       | 437.864 ± 7.465 |
| Fastfood | 557.096 ± 59.894     | 543.893 ± 69.235 |
| Nyström  | 529.239 ± 55.554      | 524.916 ± 61.374 |
| RFFNet   | 449.875 ± 6.953       | 441.696 ± 9.223 |

Table F.13: RAM usage (in MB) for the real-world regression datasets. OOM: out-of-memory.

| Model  | abalone          | compact         | powerplant      | yearprediction |
|--------|------------------|-----------------|-----------------|----------------|
| EigenPro | 459.742 ± 9.543  | 455.401 ± 16.655 | 445.329 ± 4.163 | OOM            |
| Fastfood | 450.48 ± 13.959  | 437.146 ± 7.194  | 437.583 ± 1.93  | 1649.55 ± 1.488 |
| GPR     | 519.453 ± 22.149 | 655.176 ± 84.886 | 7985.012 ± 1194.386 | OOM           |
| Kernel Ridge | 443.39 ± 5.607  | 465.184 ± 18.198 | 474.495 ± 8.587 | OOM            |
| Nyström  | 428.799 ± 2.099  | 438.998 ± 7.674  | 434.381 ± 2.475 | 1656.081 ± 4.51 |
| RFFNet   | 429.401 ± 2.586  | 432.024 ± 6.467  | 455.584 ± 7.649 | 1647.918 ± 6.724 |
| SRFF     | 495.449 ± 21.55  | 524.502 ± 43.78  | 454.3 ± 8.655  | 1662.739 ± 3.649 |
Table F.14: RAM usage (in MB) for the real-world classification datasets. OOM: out-of-memory.

| Model    | a9a     | amazon  | higgs   | w8a     |
|----------|---------|---------|---------|---------|
| EigenPro | 599.486 | OOM     | OOM     | 931.943 |
| Fastfood | 805.98  | ± 116.562 | 8397.65 | ± 337.782 | 8680.3 ± 8.591 | 1210.881 ± 76.276 |
| Nyström  | 790.729 | ± 125.592 | 5475.613 | ± 403.35 | 8680.3 ± 8.591 | 1228.162 ± 161.9 |
| RFFNet   | 599.699 | ± 16.64 | 4781.993 | ± 8.011 | 8680.3 ± 8.591 | 942.079 ± 2.453 |

Appendix G. A toy model for RFFNet’s landscape

(a) Increasing the oracle relevance degrade the identification of relevant features.

(b) Increasing the sample size improves the identification of relevant features.

In this section, we assess the effect of the sample sizes $n$, the number of random features $s$, and the value of the population relevance vector $\theta^*$ (within a restricted data-generating model) in RFFNet’s objective. We seek to understand how RFFNet, even with a highly non-convex objective function,
can identify the relevant features of the data. Our simulations manifest the parity symmetry (with respect to the entries of the relevance vector) in the loss landscape, as well as a type of effective convexity that might emerge in the large-sample setting.

We fix the number of features as $d = 2$ for visualization purposes only. We consider a “natural” input distribution with $x_i \sim \mathcal{N}(0, I_p)$, $i \in [n]$, where $n$ is the sample size. We work on a completely specified model for the true regression function: if $k_{\theta^*}$ is the ARD Gaussian kernel, with $\theta^* = (\theta_1, \theta_2)$ the population relevance vector, we choose the regression function as

$$f^*(x) = \alpha k_{\theta^*}(x_1, x),$$

where we sample $\alpha, x_1 \sim \mathcal{N}(0, 1)$.

We consider the loss landscape with respect to the relevances as the following empirical risk functional:

$$h_{\beta}(\theta_1, \theta_2) = \frac{1}{n} \sum_{i=1}^{n} \ell \left( f^*(x_i), f_{\theta, \beta}(x_i) \right), \quad (G.1)$$

where $f^* \in \mathcal{H}^*$, the RKHS uniquely associated to $k_{\theta^*}$ and $f_{\theta, \beta} \in \mathcal{F}_{\theta, \beta}$; the RFFNet’s hypothesis class,

$$\mathcal{F}_{\theta, \beta} = \{ f_{\theta, \beta}(.): \beta^\top z(\theta \circ .) : \beta \in \mathbb{R}^s, \theta \in \mathbb{R}^d \},$$

with $z : \mathbb{R}^d \to \mathbb{R}^s$ the random Fourier features map for $k_{1_d}$. For our simulations, to factor out the influence of $\beta$, we choose it based on the fact that

$$f^*(x) = \alpha k_{\theta^*}(x_1, x) \approx \alpha z(\theta^* \circ x_1)^\top z(\theta^* \circ x)$$

$$= \left( \alpha z(\theta^* \circ x_1) \right)^\top z(\theta^* \circ x)$$

$$\approx \left( \alpha z(\theta \circ x_1) \right)^\top_i z(\theta \circ x),$$

where the last approximation holds when $\theta \to \theta^*$, and the first when $s$ is sufficiently big. Thus, we take $\beta$ in (G.1) as

$$\beta = \alpha z(\theta \circ x_1).$$

In Figure G.11a, we fixed the number of samples as $n = 500$ and the number of random features as $s = 300$. For the left panel, we chose $\theta^* = \ldots$
(2, 2), while for the right panel, $\theta^* = (7, 7)$. We observe that increasing the relevance vectors degrades the identification of the relevant features, as the loss landscape becomes flat around the possible population relevances. A possible reason is that greater relevances cause functions in $\mathcal{H}^*$ to oscillate in smaller length scales; as a consequence, samples of the true regression function may be aliased, hindering the detection of the relevant features within the dataset.

In Figure (G.11b), we fixed $\theta^* = (2, 2)$. For the left panel, we choose $(n, s) = (50, 200)$, while for the right panel $(n, s) = (1000, 200)$. We observe that increasing the sample size smooths the loss landscape, damping the ripples created by the oscillatory nature of the random Fourier features map. As a consequence, the loss wells associated with the population relevances become more pronounced.

In conjunction, these figures indicate that the loss landscape of RFFNet, in the large-sample setting with “low-frequency” regression functions, might exhibit beneficial properties. We have seen that a simple choice of $\beta$ renders the RFFNet objective “effectively” convex within each orthant of $\mathbb{R}^d$, with minima that correspond exactly to the population relevance parameter.

Scripts to reproduce these figures are supplied along the code and available in the experiments/convexity folder.

References

[1] Allen, G.I., 2013. Automatic feature selection via weighted kernels and regularization. Journal of Computational and Graphical Statistics 22, 284–299. doi:10.1080/10618600.2012.681213.

[2] Avron, H., Kapralov, M., Musco, C., Musco, C., Velingker, A., Zandieh, A., 2017. Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees, in: Precup, D., Teh, Y.W. (Eds.), Proceedings of the 34th International Conference on Machine Learning, PMLR. pp. 253–262. URL: https://proceedings.mlr.press/v70/avron17a.html

[3] Bach, F., 2017. On the equivalence between kernel quadrature rules and random feature expansions. Journal of Machine Learning Research 18, 1–38. URL: http://jmlr.org/papers/v18/15-178.html
[4] Baldi, P., Sadowski, P., Whiteson, D., 2014. Searching for exotic particles in high-energy physics with deep learning. Nature Communications 5. doi:10.1038/ncomms5308.

[5] Balog, M., Tolstikhin, I., Schölkopf, B., 2018. Differentially private database release via kernel mean embeddings, in: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning, PMLR. pp. 414–422. URL: https://proceedings.mlr.press/v80/balog18a.html.

[6] Bertin, K., Lecué, G., 2008. Selection of variables and dimension reduction in high-dimensional non-parametric regression. Electronic Journal of Statistics 2, 1224–1241. doi:10.1214/08-EJS327.

[7] Bolte, J., Sabach, S., Teboulle, M., 2014. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Math. Program., Ser. A 146, 459–494.

[8] Chen, J., Stern, M., Wainwright, M.J., Jordan, M.I., 2017. Kernel feature selection via conditional covariance minimization, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/b7fede84c2be02ccb9c77107956560eb-Paper.pdf.

[9] Chen, Y., Li, Y., Liu, K., Ruan, F., 2023. Kernel learning in ridge regression ”automatically” yields exact low rank solution. doi:10.48550/ARXIV.2310.11736, arXiv:2310.11736.

[10] Cortes, C., Vapnik, V., Saitta, L., 1995. Support-vector networks. Machine Learning 1995 20:3 20, 273–297. doi:10.1007/BF00994018.

[11] Curtó, J.D., Zarza, I.C., Yang, F., Smola, A., de la Torre, F., Ngo, C.W., van Gool, L., 2020. Mckernel: A library for approximate kernel expansions in log-linear time. arXiv:1702.08159.

[12] Dance, H., Paige, B., 2022. Fast and scalable spike and slab variable selection in high-dimensional gaussian processes, in: Camps-Valls, G., Ruiz, F.J.R., Valera, I. (Eds.), Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, PMLR. pp. 7976–8002. URL: https://proceedings.mlr.press/v151/dance22a.html.
[13] Drineas, P., Mahoney, M.W., 2005. On the nystrom method for approximating a gram matrix for improved kernel-based learning. Journal of Machine Learning Research 6, 2153–2175. URL: http://jmlr.org/papers/v6/drineas05a.html

[14] Foreman-Mackey, D., Agol, E., Ambikasaran, S., Angus, R., 2017. Fast and scalable gaussian process modeling with applications to astronomical time series. The Astronomical Journal 154, 220. doi:10.3847/1538-3881/aa9332

[15] Gittens, A., Mahoney, M., 2013. Revisiting the nystrom method for improved large-scale machine learning, in: Dasgupta, S., McAllester, D. (Eds.), Proceedings of the 30th International Conference on Machine Learning, PMLR, Atlanta, Georgia, USA. pp. 567–575. URL: https://proceedings.mlr.press/v28/gittens13.html

[16] Gönen, M., Alpaydin, E., 2011. Multiple kernel learning algorithms. Journal of Machine Learning Research 12, 2211–2268. URL: http://jmlr.org/papers/v12/gonen11a.html

[17] Grandvalet, Y., Canu, S., 2002. Adaptive scaling for feature selection in svms, in: Becker, S., Thrun, S., Obermayer, K. (Eds.), Advances in Neural Information Processing Systems, MIT Press. URL: https://proceedings.neurips.cc/paper_files/paper/2002/file/ee26fc66b1369c762533bedafbfcaf6-Paper.pdf

[18] Gregorová, M., Ramapuram, J., Kalousis, A., Marchand-Maillet, S., 2018. Large-scale nonlinear variable selection via kernel random features. arXiv:1804.07169

[19] Guyon, I., Weston, J., Barnhill, S., Vapnik, V., 2002. Gene selection for cancer classification using support vector machines. Machine Learning 46, 389–422. doi:10.1023/A:1012487302797

[20] Hainmueller, J., Hazlett, C., 2014. Kernel regularized least squares: Reducing misspecification bias with a flexible and interpretable machine learning approach. Political Analysis 22, 143–168. doi:10.1093/pan/mpt019

[21] Han, I., Zandieh, A., Avron, H., 2022. Random gegenbauer features for scalable kernel methods, in: Chaudhuri, K., Jegelka, S., Song, L.,
Szepesvari, C., Niu, G., Sabato, S. (Eds.), Proceedings of the 39th International Conference on Machine Learning, PMLR. pp. 8330–8358. URL: https://proceedings.mlr.press/v162/han22g.html

[22] He, X., Wang, J., Lv, S., 2021. Efficient kernel-based variable selection with sparsistency. Statistica Sinica 31, 2123–2151. URL: https://www.jstor.org/stable/27089331.

[23] Jacot, A., Gabriel, F., Hongler, C., 2018. Neural tangent kernel: Convergence and generalization in neural networks. Advances in neural information processing systems 31.

[24] Jitkrittum, W., Kanagawa, H., Schölkopf, B., 2020. Testing goodness of fit of conditional density models with kernels, in: Peters, J., Sontag, D. (Eds.), Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR. pp. 221–230. URL: https://proceedings.mlr.press/v124/jitkrittum20a.html.

[25] Jordan, M.I., Liu, K., Ruan, F., 2021. On the self-penalization phenomenon in feature selection. arXiv:2110.05852.

[26] Kar, P., Karnick, H., 2012. Random feature maps for dot product kernels, in: Lawrence, N.D., Girolami, M. (Eds.), Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, PMLR, La Palma, Canary Islands. pp. 583–591. URL: https://proceedings.mlr.press/v22/kar12.html.

[27] Keerthi, S., Sindhwani, V., Chapelle, O., 2006. An efficient method for gradient-based adaptation of hyperparameters in svm models, in: Schölkopf, B., Platt, J., Hoffman, T. (Eds.), Advances in Neural Information Processing Systems, MIT Press. URL: https://proceedings.neurips.cc/paper/2006/file/cc431f7ec4437de061c2577a4603995-Paper.pdf.

[28] Kimeldorf, G., Wahba, G., 1971. Some results on tchebycheffian spline functions. Journal of Mathematical Analysis and Applications 33, 82–95. URL: https://www.sciencedirect.com/science/article/pii/0022247X71901843, doi 10.1016/0022-247X(71)90184-3.

[29] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.
[30] Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A., 2011. Lp-norm multiple kernel learning. Journal of Machine Learning Research 12, 953–997. URL: http://jmlr.org/papers/v12/kloft11a.html

[31] Lafferty, J., Wasserman, L., 2008. Rodeo: Sparse, greedy nonparametric regression. The Annals of Statistics 36, 28–63.

[32] Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I., 2004. Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research 5, 27–72. URL: https://www.jmlr.org/papers/v5/lanckriet04a.html

[33] Lázaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C.E., Figueiras-Vidal, A.R., 2010. Sparse spectrum gaussian process regression. Journal of Machine Learning Research 11, 1865–1881. URL: http://jmlr.org/papers/v11/lazaro-gredilla10a.html

[34] Le, Q., Sarlos, T., Smola, A., 2013. Fastfood - computing hilbert space expansions in loglinear time, in: Dasgupta, S., McAllester, D. (Eds.), Proceedings of the 30th International Conference on Machine Learning, PMLR, Atlanta, Georgia, USA. pp. 244–252. URL: https://proceedings.mlr.press/v28/le13.html

[35] Li, J., Liu, Y., Wang, W., 2020. Automated spectral kernel learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4618–4625.

[36] Li, Z., Ton, J.F., Oglic, D., Sejdinovic, D., 2021. Towards a unified analysis of random fourier features. Journal of Machine Learning Research 22, 1–51. URL: http://jmlr.org/papers/v22/20-1369.html

[37] Liu, F., Xu, W., Lu, J., Sutherland, D.J., 2021. Meta two-sample testing: Learning kernels for testing with limited data, in: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 5848–5860. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/2e6d9c6052e99fcdfa61d9b9da273ca2-Paper.pdf

[38] Louw, N., Steel, S.J., 2006. Variable selection in kernel fisher discriminant analysis by means of recursive feature elimination. Computa-
[39] Lu, J., Hoi, S.C., Wang, J., Zhao, P., Liu, Z.Y., 2016. Large scale online kernel learning. Journal of Machine Learning Research 17, 1–43. URL: http://jmlr.org/papers/v17/14-148.html.

[40] Ma, S., Belkin, M., 2017. Diving into the shallows: A computational perspective on large-scale shallow learning, in: Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/hash/bf424cb7b0dea050a42b9739eb261a3a-Abstract.html.

[41] Meanti, G., Carratino, L., De Vito, E., Rosasco, L., 2022. Efficient hyperparameter tuning for large scale kernel ridge regression, in: Camps-Valls, G., Ruiz, F.J.R., Valera, I. (Eds.), Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, PMLR. pp. 6554–6572. URL: https://proceedings.mlr.press/v151/meanti22a.html.

[42] Mohri, M., Rostamizadeh, A., Talwalkar, A., 2018. Foundations Of Machine Learning, Second Edition. Adaptive Computation and Machine Learning series, MIT Press.

[43] Neal, R.M., 1996. Bayesian Learning For Neural Networks. Springer New York. doi:10.1007/978-1-4612-0745-0.

[44] Paine, A.E., Elfving, V.E., Kyriienko, O., 2023. Quantum kernel methods for solving regression problems and differential equations. Phys. Rev. A 107, 032428. doi:10.1103/PhysRevA.107.032428.

[45] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[46] Powell, M.J., 1973. On search directions for minimization algorithms. Mathematical Programming 4, 193–201. doi:10.1007/BF01584660.
[47] Quiñonero-Candela, J., Rasmussen, C.E., 2005. A unifying view of sparse approximate gaussian process regression. Journal of Machine Learning Research 6, 1939–1959. URL: http://jmlr.org/papers/v6/quinonero-candela05a.html

[48] Rahimi, A., Recht, B., 2007. Random features for large-scale kernel machines, in: Platt, J., Koller, D., Singer, Y., Roweis, S. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03db5392effeb8f18fda755-Paper.pdf

[49] Rahimi, A., Recht, B., 2008. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. Advances in Neural Information Processing Systems 21.

[50] Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes For Machine Learning.

[51] Ruan, F., Liu, K., Jordan, M.I., 2021. Taming nonconvexity in kernel feature selection – favorable properties of the laplace kernel. arXiv:2106.09387

[52] Rudi, A., Carratino, L., Rosasco, L., 2017. Falkon: An optimal large scale kernel method, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/05546b0e38ab9175cd905eeb3c6eb76-Paper.pdf

[53] Rudi, A., Rosasco, L., 2017. Generalization properties of learning with random features, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/61b1fb3f59e28c67f3925f3c79be81a1-Paper.pdf

[54] Scetbon, M., Meunier, L., Romano, Y., 2022. An asymptotic test for conditional independence using analytic kernel embeddings, in: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (Eds.), Proceedings of the 39th International Conference on Machine
[55] Schölkopf, B., Smola, A.J., 2002. Learning With Kernels: Support Vector Machines, Regularization, Optimization, And Beyond. The MIT Press.

[56] Schölkopf, B., Tsuda, K., Vert, J.P., 2004. Kernel Methods In Computational Biology. MIT Press.

[57] Shalev-Shwartz, S., Ben-David, S., 2014. Understanding Machine Learning: From Theory To Algorithms. Cambridge University Press. URL: http://ebooks.cambridge.org/ref/id/CBO9781107298019.

[58] Shekhar, S., Kim, I., Ramdas, A., 2022. A permutation-free kernel two-sample test, in: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. pp. 18168–18180. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/731b952bdc833485eb72f458cdd5c489-Paper-Conference.pdf.

[59] Sinha, A., Duchi, J.C., 2016. Learning kernels with random features, in: Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://papers.nips.cc/paper_files/paper/2016/hash/e70611883d2760c8bbaf84acb29e3446-Abstract.html.

[60] Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) 58, 267–288. URL: https://www.jstor.org/stable/2346178.

[61] Tipping, M.E., 2001. Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research 1, 211–244. URL: https://jmlr.csail.mit.edu/papers/v1/tipping01a.html.

[62] Wachman, G., Khariton, R., Protopapas, P., Alcock, C.R., 2009. Kernels for periodic time series arising in astronomy, in: Buntine, W., Grobelnik, M., Mladenčić, D., Shawe-Taylor, J. (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer, Berlin, Heidelberg. pp. 489–505. doi:10.1007/978-3-642-04174-7_32
[63] Wainwright, M.J., 2019. High-Dimensional Statistics. Cambridge University Press. URL: https://www.cambridge.org/core/product/identifier/9781108627771/type/book.

[64] Williams, C., Seeger, M., 2000. Using the nyström method to speed up kernel machines, in: Advances in Neural Information Processing Systems, MIT Press. URL: https://papers.nips.cc/paper_files/paper/2000/hash/19de10adbaa1b2ee13f77f679fa1483a-Abstract.html.

[65] Yang, T., Li, Y.F., Mahdavi, M., Jin, R., Zhou, Z.H., 2012. Nyström method vs random fourier features: A theoretical and empirical comparison, in: Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2012/hash/621bf66ddb7c962aa0d22ac97d69b793-Abstract.html.

[66] Yang, Z., Wilson, A., Smola, A., Song, L., 2015. A la Carte – Learning Fast Kernels, in: Lebanon, G., Vishwanathan, S.V.N. (Eds.), Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, PMLR, San Diego, California, USA. pp. 1098–1106. URL: https://proceedings.mlr.press/v38/yang15b.html.

[67] Yao, J., Erichson, N.B., Lopes, M.E., 2023. Error estimation for random fourier features, in: Ruiz, F., Dy, J., van de Meent, J.W. (Eds.), Proceedings of The 26th International Conference on Artificial Intelligence and Statistics, PMLR. pp. 2348–2364. URL: https://proceedings.mlr.press/v206/yao23a.html.

[68] Zhang, R., Muandet, K., Schölkopf, B., Imaiizumi, M., 2021. Instrument space selection for kernel maximum moment restriction. arXiv:2106.03340

[69] Zien, A., Ong, C., 2007. Multiclass multiple kernel learning, in: ICML 2007, Max-Planck-Gesellschaft. ACM Press, New York, NY, USA. pp. 1191–1198. doi:10.1145/1273496.1273646

45