UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Scalable and interpretable kernel methods based on random Fourier features

Mateus Piovezan Otto

Dissertação de Mestrado do Programa Interinstitucional de Pós-Graduação em Estatística (PIPGEs)

ICMC USP

SÃO CARLOS

Mateus Piovezan Otto

Scalable and interpretable kernel methods based on random Fourier features

Dissertation submitted to the Institute of Mathematics and Computer Science – ICMC-USP and to the Department of Statistics – DEs-UFSCar – in accordance with the requirements of the Statistics Interagency Graduate Program, for the degree of Master in Statistics. *FINAL VERSION*

Concentration Area: Statistics

Advisor: Prof. Dr. Rafael Izbicki

USP – São Carlos
March 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Piovezan Otto, Mateus
P662s Scalable and interpretable kernel methods based
on random Fourier features / Mateus Piovezan Otto;
orientador Rafael Izbicki. -- São Carlos, 2023.
79 p.

Dissertação (Mestrado - Programa
Interinstitucional de Pós-graduação em Estatística) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2023.

1. Kernel methods. 2. Feature importance. 3.
Machine learning. 4. Optimization. I. Izbicki,
Rafael, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Mateus Piovezan Otto

Métodos de kernel escaláveis e interpretáveis baseados em random Fourier features

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP e ao Departamento de Estatística – DEs-UFSCar, como parte dos requisitos para obtenção do título de Mestre em Estatística – Programa Interinstitucional de Pós-Graduação em Estatística. *VERSÃO REVISADA*

Área de Concentração: Estatística

Orientador: Prof. Dr. Rafael Izbicki

USP – São Carlos
Março de 2023

This work is dedicated to my brothers,
Guilherme and Pedro.

ACKNOWLEDGEMENTS

I'm extremely grateful to my advisor, Rafael Izbicki, for his enthusiastic guidance, cordiality, patience, and friendliness. Rafael's ability to pose many interesting questions, ever since we met, left me with a deep impression and shaped how I see research in statistics. Thank you, Rafael!

I am immensely indebted to my parents, Ana and Ednilson, for their endless support and willingness to be present in my life by any means. Thank you for the long talks, travels, jokes, and my two brothers, Guilherme and Pedro, who are also my best friends. You inspire me as much as you make me laugh. I would also like to thank my grandmothers, Lúcia and Salete, for their kindness, love, good weekends, and delicious food.

My most loving and sincere gratitude to my girlfriend, Flávia. Loving and being with you is so fulfilling! You appeared in my life and painted the world with brighter colors. Thank you for the good moments, laughs, and the most adoring memories I could ever carry.

I am also grateful to my professors, Luis Ernesto Salasar, Andressa Cerqueira, Luís Gustavo Esteves, and George Lucas Moraes Pezzot, for the compelling and well-structured courses they taught. It was a pleasure to learn with you. I would like to extend my sincere gratitude to the members of my qualification committee, Roberto Imbuzeiro Oliveira and Julio Michael Stern, for their valuable suggestions and comments during the exam.

Special thanks to my friends Felipe Alves, Felipe Picoli, Gabriel, Giuliano, and Guilherme, for the good moments and conversations we had. Also, great thanks to my friends from PIPGES, Luana, Luben, Paulo, and Rodrigo; it was a pleasure to meet and talk to you.

I would like to acknowledge PIPGES' officers, especially Julio Cezar de Barros and Monique da Conceição, for their celerity in solving any problems and answering any inquiries. Likewise, my sincere thanks to the officers of the ICMC library for their cordiality and for maintaining a great place to study.

Lastly, I would like to sincerely thank FAPESP for the financial support during the execution of this project through grant 2021/02178-8, São Paulo Research Foundation (FAPESP).

*"Problems worthy of attack
prove their worth by fighting back."*
(Piet Hein)

RESUMO

OTTO, M. P. **Métodos de kernel escaláveis e interpretáveis baseados em random Fourier features**. 2023. 79 p. Dissertação (Mestrado em Estatística – Programa Interinstitucional de Pós-Graduação em Estatística) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Métodos de kernel são uma classe de modelos de aprendizado de máquina baseados em kernels positivo semidefinidos, que servem como medidas de similaridade entre covariáveis. Exemplos de métodos de kernel incluem a regressão ridge com kernels, as máquinas de vetor de suporte e os splines suavizadores. Apesar do seu amplo uso, os métodos de kernel possuem duas desvantagens significativas. Em primeiro lugar, ao operar sobre todos os pares de observações, eles demandam grande quantidade de memória e computação, o que impossibilita sua aplicação em grandes conjuntos de dados. Este problema pode ser resolvido através de aproximações da matriz do kernel via *random Fourier features* ou precondicionadores. Em segundo lugar, a maioria dos kernels tratam todas as covariáveis disponíveis como igualmente relevantes, desconsiderando seu impacto na predição. Isso resulta em um descréscimo na interpretabilidade, uma vez que a influência de covariáveis irrelevantes não é mitigada. Neste trabalho, nós estendemos a teoria de *random Fourier features* para os kernels com Determinação Automática de Relevância e propomos um novo método de kernel que integra a otimização dos parâmetros do kernel ao treinamento. Os parâmetros do kernel reduzem o efeito das covariáveis irrelevantes e podem ser utilizados para seleção de variáveis pós-processamento. O método proposto é avaliado em diversos conjuntos de dados e comparado a algoritmos convencionais de aprendizado de máquina.

Palavras-chave: Métodos de kernel; importância de covariáveis; aprendizado de máquina; otimização.

ABSTRACT

OTTO, M. P. **Scalable and interpretable kernel methods based on random Fourier features**. 2023. 79 p. Dissertação (Mestrado em Estatística – Programa Interinstitucional de Pós-Graduação em Estatística) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Kernel methods are a class of statistical machine learning models based on positive semidefinite kernels, which serve as a measure of similarity between data features. Examples of kernel methods include kernel ridge regression, support vector machines, and smoothing splines. Despite their widespread use, kernel methods face two main challenges. Firstly, due to operating on all pairs of observations, they require a large amount of memory and calculation, making them unsuitable for use with large datasets. This issue can be solved by approximating the kernel function via random Fourier features or preconditioners. Secondly, most used kernels consider all features to be equally relevant, without considering their actual impact on the prediction. This results in decreased interpretability, as the influence of irrelevant features is not mitigated. In this work, we extend the random Fourier features framework to Automatic Relevance Determination (ARD) kernels and proposes a new kernel method that integrates the optimization of kernel parameters during training. The kernel parameters reduce the effect of irrelevant features and might be used for post-processing variable selection. The proposed method is evaluated on several datasets and compared to conventional algorithms in machine learning.

Keywords: Kernel methods; feature importance; machine learning; optimization.

LIST OF SYMBOLS

$[n]$ — The set $\{1, \ldots, n\}$, for $n \in \mathbb{N}$.

$\arg\min_{x \in C} f(x)$ — The set $\{x \in C : f(x) = \min_{z \in C} f(z)\}$.

$\mathbb{R}^+$ — The set $\{x \in \mathbb{R} : x \geq 0\}$.

$\circ$ — The element-wise multiplication, $x \circ y = (x_1 y_1, \ldots, x_p y_p)$ for $x, y \in \mathbb{R}^p$.

$\mathbf{1}_p$ — The vector $(1, \ldots, 1) \in \mathbb{R}^p$.

I_p — The $p \times p$ identity matrix.

CONTENTS

CHAPTER 1

INTRODUCTION

Statistical learning methods based on kernel functions have been successfully used in many fields, ranging from feature selection (JORDAN; LIU; RUAN, 2021; HE; WANG; LV, 2018) and causal inference (ZHANG *et al.*, 2021) to privacy (BALOG; TOLSTIKHIN; SCHöLKOPF, 2018) and hypothesis testing (LIU *et al.*, 2021; SHEKHAR; KIM; RAMDAS, 2022; JITKRITTUM; KANAGAWA; SCHöLKOPF, 2020; SCETBON; MEUNIER; ROMANO, 2022). Given their nonparametric nature and many theoretical guarantees, these methods allow for principled modelling of complex relationships in real-world data. In particular, accomplished regression and classification methods such as Kernel Ridge Regression and Support Vector Machines (CORTES; VAPNIK; SAITTA, 1995) are based on kernels.

Unfortunately, because kernel methods operate on all pairs of observations, they possess stringent memory and time requirements, which hinders their applicability. Fortunately, approximate solvers were recently developed for scaling up kernel methods, based on random Fourier features (RAHIMI; RECHT, 2008a; CURTó *et al.*, 2017; LE; SARLOS; SMOLA, 2014) or preconditioners, such as the Nyström method (RUDI; CARRATINO; ROSASCO, 2017). However, these scalable approaches are based on approximations of isotropic kernels, which weigh all features equally in the kernel, and cannot remove the influence of features that are irrelevant to the predictive task at hand. This translates into an interpretability shortage and leads to poor predictive performance in problems with many irrelevant variables (LAFFERTY; WASSERMAN, 2008; BERTIN; LECUÉ, 2008).

Indeed, a common approach for improving kernel methods' interpretability is to assign a distinct weight to each feature in the kernel parametrization. This process generates the so-called Automatic Relevance Determination (ARD) kernels (RASMUSSEN;

WILLIAMS, 2006), such as the ARD gaussian kernel

$$k_\lambda(x, x') = \exp\left[-\frac{1}{2}\sum_{j=1}^{p} \lambda_j^2 (x_j - x'_j)^2\right],$$

where x_i is the i-th component of the feature vector x and λ_i is the relevance of the i-th feature. In practice, it is expected that if the relevance vector λ is estimated using available data, then λ_i would be automatically set to zero for irrelevant features.

In this work, we propose RFFNet, a new approach for fitting kernel methods that is scalable and interpretable. On the one hand, our method relies on kernel approximations with random Fourier features, which significantly reduces the number of parameters to be estimated and decreases the computational cost of training kernel methods. On the other hand, contrary to standard random Fourier features applications, our framework effectively employs ARD kernels. We show that ARD kernels correspond to spectral densities that have the kernel relevances as "scale" parameters. Through this, we show that it is possible to decouple λ from the RFF map that approximates evaluations of an ARD kernel and, therefore easily estimate λ with data.

1.1 Summary of contributions

The main contributions of this work are:

- proposing a new scalable kernel method for classification and regression tasks;
- proposing random Fourier features for ARD kernels tailored for identifying relevant features and measuring feature importance;
- implementing a block stochastic gradient descent method that shows good empirical performance in solving the objective function introduced by our approach;
- proposing an adaptation of our approach suited for measuring feature importance in neural networks;
- implementing our method in a modular and efficient manner with the PyTorch framework (PASZKE *et al.*, 2019).

1.2 Relation to prior work

One of the simplest approaches to improve kernel methods in the presence of irrelevant features is to assign a weight, or relevance, to each feature in the definition of the kernel function. However, tuning such parameters is difficult (KEERTHI; SINDHWANI;

CHAPELLE, 2006). For instance, cross-validation cannot be used to choose these values, as this would require a grid over a possibly high-dimensional feature space.

Other approaches, such as Guyon *et al.* (2002) and Louw and Steel (2006), perform recursive backward elimination of features. Although effective in low dimensions, this tends to be slow in problems with many features.

An alternative approach that is more related to our method is to develop a loss function that includes the relevances as part of the objective function. This is done by Allen (2013), which proposes an iterative feature extraction method. Unfortunately, this procedure does not scale with the sample size because of memory and computational processing requirements. An alternative approach, Sparse Random Fourier Features (SRFF), is developed by Gregorová *et al.* (2018), which uses random Fourier features as the basis for feature selection with ARD kernels. However, their solution is specific to the regression setting (with mean squared error loss) and keeps the number of random features fixed, which hinders SRFF predictive performance. Additionally, SRFF is based on a two-step exact minimization procedure, minimizing first with respect to the parameters of the kernel expansion, then with respect to the relevances. It is known that this procedure can cause indefinite cycling, preventing convergence (POWELL, 1973).

Recently, Jordan, Liu and Ruan (2021) 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 ARD kernels, the solution is specific to Kernel Ridge Regression and Metric Learning and does not employ any kernel matrix approximation.

1.3 Organization

Chapter 2 reviews the supervised learning setting, gives a primer on optimization and presents the basics of kernel methods. Chapter 3 describes our approach for improving kernel methods' scalability and interpretability. Chapter 4 presents how the method was evaluated and validates some aspects of our approach. Chapter 5 compares our approach to established baselines in simulation and real data experiments. Chapter 6 concludes the dissertation, giving directions for future work.

CHAPTER 2

BACKGROUND

This chapter reviews basic definitions, facts, and notation from the supervised learning framework, optimization, and the theory of kernel methods. In particular, we emphasize the origin of the scalability and interpretability bottlenecks for usual kernel methods.

2.1 Supervised learning

We follow the standard formalization of statistical machine learning (ML) tasks as in Shalev-Shwartz and Ben-David (2014). Let $\mathcal{X} \subseteq \mathbb{R}^p$ be the input space and $\mathcal{Y}$ be the output space. For instance, $\mathcal{Y} \subseteq \mathbb{R}$ in regression, and $\mathcal{Y} = \{0, 1\}$ in binary classification. Let (X, Y) be a random vector defined on $\mathcal{X} \times \mathcal{Y}$ with joint probability distribution $\mathbb{P}_{X,Y}$. We call X the vector of *features* and Y the *response*.

Given a finite training sample $(x_i, y_i)_{i\in[n]}$ of independent and identically distributed realizations of $(X, Y) \sim \mathbb{P}_{X,Y}$, the goal of supervised learning is to train or estimate a function $f : \mathcal{X} \to \mathcal{Y}$ that approximates the relationship between the input X and the output Y. Conventionally, we restrict this function to a certain function space $\mathcal{H}$, whose elements are mappings between $\mathcal{X}$ and $\mathcal{Y}$. For instance, $\mathcal{H}$ may be taken as the space of linear maps, two-layer neural networks, or decision stumps (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018).

To measure how well functions in $\mathcal{H}$ approximate the relationship between X and Y, one introduces the notion of a loss function. A function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R} \cup \{+\infty\}$ is a loss function if $\ell(y, y) = 0$ for all $y \in \mathcal{Y}$ and $\ell(y, y') \geq 0$ for all $y, y' \in \mathcal{Y}$. For example, the most common loss function for regression problems is the squared error loss $\ell(y, y') = (y - y')^2$.

By specifying a loss function, the goal of supervised learning can be recast as an optimization problem in $\mathcal{H}$. Specifically, the objective is to find a function $f^* \in \mathcal{H}$ that minimizes the loss $\ell(f(X), Y)$ in expectation,

$$f^* \in \underset{f \in \mathcal{H}}{\arg\min}\, \mathbb{E}_{X,Y} \left[\ell(f(X), Y)\right]. \tag{2.1}$$

In practice, however, computing this expectation is unfeasible since we do not have access to the joint probability $\mathbb{P}_{X,Y}$. Instead, one must resort to the empirical expectation over the training sample $(x_i, y_i)_{i \in [n]}$. In this case, one finds f^* such that

$$f^* \in \underset{f \in \mathcal{H}}{\arg\min} \sum_{i=1}^{n} \ell(f(x_i), y_i). \tag{2.2}$$

This procedure is the defining rule for the Empirical Risk Minimization (ERM) paradigm in statistical machine learning. Nevertheless, although the ERM rule suffices for learning many families of learning problems (SHALEV-SHWARTZ; BEN-DAVID, 2014), it can lead to unstable estimates f^* and completely disregards the complexity of the function class $\mathcal{H}$. For instance, if $\mathcal{H}$ is a rich function space (such as the space of deep neural networks), the ERM rule may lead to severe over-fitting.

The Regularized Loss Minimization (RLM) paradigm is adopted to cope with the pointed drawbacks of ERM. RLM is a modification of the Structural Risk Minimization (SRM) paradigm (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018) and seeks to control the complexity of functions in $\mathcal{H}$ by introducing a regularization function that penalizes complex hypotheses in $\mathcal{H}$. The loss and the regularization function are then jointly minimized on the training sample, generating an optimal estimate f^* as

$$f^* \in \underset{f \in \mathcal{H}}{\arg\min} \left[\sum_{i=1}^{n} \ell(f(x_i), y_i) + \mu \mathcal{R}(f)\right], \tag{2.3}$$

where $\mathcal{R} : \mathcal{H} \to \mathbb{R}^+$ is the regularization function and $\mu > 0$ controls the regularization strength. Choosing an appropriate μ is central to guarantee that f^* generalizes to unseen realizations of (X, Y), and it is usually accomplished via cross-validation, although other approaches exist (BERTRAND *et al.*, 2021; PEDREGOSA, 2016). In turn, the choice of regularization function $\mathcal{R}$ is commonly tied to the structure of $\mathcal{H}$ (WAINWRIGHT, 2019). For instance, if $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{H}$ is the space of linear functions

$$\mathcal{H} = \{x \to \beta^\intercal x + b : \beta \in \mathbb{R}^p, b \in \mathbb{R}\},$$

then natural choices of regularization are $\mathcal{R}(f) = \|\beta\|_2^2$ or $\mathcal{R}(f) = \|\beta\|_1$, widely known as ℓ_2 and ℓ_1 regularizations, respectively. In section 2.3, we will see a natural regularization function for kernel methods.

2.2 Optimization

Training supervised learning models requires solving an optimization problem in the function space $\mathcal{H}$. In fact, statistics and machine learning are intimately connected to optimization. For instance, obtaining M-estimators (GEER, 2000) or maximum a posteriori estimates requires solving maximization problems, and there are many relations between sampling and optimization, some of them just recently established (CHEWI *et al.*, 2022; CHEN *et al.*, 2022).

In this section, we briefly cover theoretical and algorithmic aspects of optimization in both the convex and non-convex settings. Even if the main optimization problem tackled in this work is non-convex, convex optimization is considered a source of solution heuristics, convex relaxations for non-convex problems (via convex surrogate losses, for instance), and bounds for global optimization (BOYD; VANDERBERGHE, 2004), which justifies this review.

Convex analysis

We begin with the basic definition of convex sets, the building blocks of convex analysis. Intuitively, a set is convex if every point in the set can be connected to every other by a straight path entirely lying in the set.

Definition 1 (Convex set)**.** A subset C of $\mathbb{R}^n$ is convex if $(1-\lambda)x + \lambda y \in C$ for all $x, y \in C$ and $0 < \lambda < 1$.

Trivially, $\mathbb{R}^n$, $\varnothing$, and any singleton $\{x\}$ for $x \in \mathbb{R}^n$ are convex sets. We give some other examples next.

Example 1. In each of the following cases, C is a convex subset of $\mathbb{R}^n$.

1. $C = B(x, \varepsilon)$, with $B(x, \varepsilon) = \{y \in \mathbb{R}^n : d(x, y) < \varepsilon\}$ an open ball centered at $x \in \mathbb{R}^n$ with radius $\varepsilon > 0$.
2. C a half-space, say $C = \{x \in \mathbb{R}^n : \langle x, b\rangle \leq \alpha, b \in \mathbb{R}^n, b \neq 0, \alpha \geq 0\}$.
3. $C = \bigcap_{i \in I} C_i$, where $(C_i)_{i \in I}$ is a family of convex sets.

Convex sets have many interesting geometrical properties, such as the separating hyperplane theorem (ROCKAFELLAR, 1970) and the supporting hyperplane theorem (BOYD; VANDERBERGHE, 2004). These properties make the study of these objects a very enticing subject on its own. Nevertheless, we will not delve into these results and point the interested reader to the aforementioned references. Our focus will be on convex functions, which lie at the heart of modern optimization.

Definition 2 (Convex function)**.** Let $f : S \to \mathbb{R} \cup \{\pm\infty\}$ be a function defined on $S \subset \mathbb{R}^n$. Then f is convex if its epigraph, $\operatorname{epi} f = \{(x, \mu) : x \in S, \mu \in \mathbb{R}, f(x) \leq \mu\}$, is a convex subset of $\mathbb{R}^{n+1}$.

Alternatively, these functions can be characterized by the following proposition, which usually appears in the literature as the definition of convex functions.

Proposition 1 (Convex function)**.** Let f be a function from a convex set C to $[-\infty, +\infty]$. Then f is convex if and only if

$$f\left((1-\lambda)x + \lambda y\right) \leq (1-\lambda)f(x) + \lambda f(y),$$

for all $x, y \in C$ and $0 < \lambda < 1$.

Proof. See Bauschke and Combettes (2017). □

In practice, Proposition 1 can hardly help asserting if a given function is convex. The following second-order condition is the primary tool used to do so.

Theorem 1. Let C be a convex set and $f : C \to (-\infty, +\infty]$ be a convex function. Assume that f is twice differentiable in C. Then f is convex if and only if its Hessian is positive semidefinite,

$$\nabla^2 f(x) \succeq 0$$

for all $x \in C$.

Proof. See, for instance, Rockafellar (1970). □

With this criterion, we can prove that the following functions are all convex on $\mathbb{R}$.

Example 2 (Convex functions defined on $\mathbb{R}$)**.**

1. $f(x) = e^{\alpha x}, \alpha \in \mathbb{R}$,
2. $f(x) = x^p$ if $x \geq 0$, $f(x) = +\infty$ if $x < 0$, for $1 \leq p < +\infty$.
3. $f(x) = (\alpha^2 - x^2)^{-\frac{1}{2}}$ if $|x| < \alpha$, $f(x) = +\infty$ if $|x| \geq \alpha$, where $\alpha > 0$.
4. $f(x) = -\log x$ if $x > 0$, $f(x) = +\infty$ if $x \geq 0$.

These functions and functional operations that preserve convexity can be used to construct convex optimization objectives or suggest convex relaxations for non-convex objectives.

In optimization, one is interested in a particular class of convex functions denominated proper. These functions are not constantly equal to $+\infty$, which makes them liable to minimization.

Definition 3 (Proper convex function)**.** Let $f : C \to (-\infty, +\infty]$ be a convex function. f is called proper if $f(x) < +\infty$ for some $x \in C$.

With this property, we can show the following proposition. This result underlines the fundamental importance of convexity in minimization problems.

Proposition 2. Let $f : C \to (-\infty, +\infty]$ be a proper convex function. Then every local minimizer of f is a global minimizer.

Proof. Take $x \in C$ and $\varepsilon > 0$ such that $f(x) = \min f(B(x; \varepsilon))$, with $B(x, \varepsilon)$ an open ball centered at x with radius ε. Let $y \in C \setminus B(x, \varepsilon)$ and set $\alpha = 1 - \varepsilon / \|x - y\|$ and $z = \alpha x + (1 - \alpha) y$. Then $\alpha \in (0, 1)$ and $z \in B(x, \varepsilon)$. Thus, from the fact that x is a local minimizer and the convexity of f, we deduce that

$$f(x) \leq f(z) \leq f\left(\alpha x + (1 - \alpha) y\right) \leq \alpha f(x) + (1 - \alpha) f(y),$$

which implies that $f(x) \leq f(y)$. Thus, since y is arbitrary, we conclude that every local minimizer of f is a global minimizer. □

For other results concerning the existence and uniqueness of minimizers of convex functions, see Bauschke and Combettes (2017).

2.2.1 A primer on convex optimization

We now summarize some algorithmic aspects of convex optimization. The emphasis will be on descent methods, a family of optimization procedures that includes the widespread gradient descent methods. In fact, gradient descent methods are the optimization workhorse of modern machine learning, with applications ranging from generalized linear models (BERTRAND *et al.*, 2022) and kernel learning (CARRATINO; RUDI; ROSASCO, 2018), to neural networks (KINGMA; BA, 2014) and deep reinforcement learning (SCHULMAN *et al.*, 2017).

As a general setting, consider the unconstrained and smooth optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x), \tag{2.4}$$

where $f : \mathbb{R}^n \to (-\infty, +\infty]$ is a convex and differentiable objective function with $\operatorname{dom}(f) = \mathbb{R}^n$. We assume that there exists an optimal point $x^* = \arg\min_{x \in \mathbb{R}^n} f(x)$ and that f is L-smooth, that is, f has L-Lipschitz continuous gradients.

Descent methods

A very natural approach to problem (2.4) is to find a sequence of iterates $x^{(k)}$ of the form

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}, \qquad k = 1, 2, \ldots, \tag{2.5}$$

where $t^{(k)} > 0$ is the step-size and $\Delta x^{(k)}$ is the search direction at the k-th iteration, such that

$$f(x^{(k+1)}) < f(x^{(k)}).$$

Methods of this type are called descent methods.

From the convexity of f (IZMAILOV; SOLODOV, 2018), it can be shown that

$$\nabla f(x^{(k)})^\intercal \Delta x^{(k)} < 0,$$

so $\Delta x^{(k)}$ must make an acute angle with the negative gradient.

Gradient descent

When the search direction at the k-th iteration $\Delta x^{(k)}$ is taken as the negative gradient, $\Delta x^{(k)} = -\nabla f(x^{(k)})$, the iteration scheme in (2.5) reduces to

$$x^{(k+1)} = x^{(k)} - t^{(k)} \nabla f(x^{(k)}), \qquad k = 1, 2, \ldots, \tag{2.6}$$

which is the defining rule of the gradient descent method. The iterations in (2.6) proceed until a stopping criterion is met. The stopping criterion is sometimes chosen to check whether the iterations result in a sufficiently small gradient, $\|\nabla f(x^k)\| \leq \varepsilon$, with $\varepsilon > 0$, indicating that the method is close to a stationary point.

In the setting where the objective function f is L-smooth and convex, the iterations Equation 2.6 have a convergence guarantee as long as the step size is not too big.

Theorem 2. Gradient descent with fixed step size $t \leq 1/L$ satisfies

$$f\left(x^{(k)}\right) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}. \tag{2.7}$$

Proof. See section A.1. □

Theorem 2 shows that gradient descent has a convergence rate of $O(1/k)$. This implies that to get a bound like $f(x^{(k)}) - f(x^*) \leq \varepsilon$, one needs $O(1/\varepsilon)$ iterations, which is called sub-linear convergence.

In the case f is not convex, it is not possible to derive a bound like (2.7), as the problem of finding the minima of non-convex functions is much more complex. Nevertheless, it is possible to bound the norm of the gradients for non-convex optimization with gradient descent, as shown below.

Theorem 3. Gradient descent with fixed step size $t \leq 1/L$ satisfies

$$\min_{0 \leq i \leq k} \|\nabla f(x^{(i)})\|_2^2 \leq \frac{2}{t(k+1)} \left[f(x^{(0)}) - f(x^*) \right].$$

Proof. See section A.2. □

Notice that, in this context, gradient descent has a slower rate of convergence, needing $O(1/\sqrt{\varepsilon})$ iterations to find a point x such that with $\|\nabla f(x)\|_2 \leq \varepsilon$.

Proximal gradient descent

In many situations, as will be the case for RFFNet, the objective f can be split into differentiable and possibly non-differentiable components. Suppose that

$$f(x) = g(x) + h(x), \tag{2.8}$$

with g convex, differentiable, with $\operatorname{dom}(g) = \mathbb{R}^n$, and h convex, not necessarily differentiable. In this case, the gradient descent updates for the objective must be modified to account for the nonsmoothness of h (PARIKH; BOYD, 2013).

The defining rule for the proximal gradient descent method is

$$x^{(k+1)} = \operatorname{prox}_{h,t}\left(x^{(k)} - t^{(k)}\,\nabla g\left(x^{(k)}\right)\right), \qquad k = 1, 2, \ldots, \tag{2.9}$$

where $\operatorname{prox}_{h,t}$ is the proximal operator of h.

Definition 4 (Proximal operator)**.** The proximal operator $\operatorname{prox}_{h,t} : \mathbb{R}^n \to \mathbb{R}^n$ of h, with $t > 0$, is defined as

$$\operatorname{prox}_{h,t}(x) = \arg\min_z \left(\frac{1}{2t}\|z - x\|_2^2 + h(z)\right).$$

These operators can be computed in closed form for numerous functions that figure in optimization problems.

Example 3 (Examples of the proximal operator)**.** Suppose $\lambda > 0$.

1. Let $h(x) = \lambda\|x\|_2^2$, then
$$\operatorname{prox}_{\lambda\|\cdot\|_2^2,t}(x) = \frac{tx}{t + 2\lambda},$$
which is a shrinkage of the vector x.

2. Let $h(x) = \lambda\|x\|_1$, then
$$\operatorname{prox}_{\lambda\|\cdot\|_1,t}(x) = S_{\lambda t}(x),$$
where $S_{\lambda t}(x)$ is the soft-thresholding operator, with components
$$[S_\lambda(x)]_i = \begin{cases} x_i - \lambda, & x_i > \lambda, \\ x_i + \lambda, & x_i < -\lambda, \\ 0, & x_i \in [-\lambda, \lambda]. \end{cases}$$

3. Let C be a convex set and $h(x) = \iota_C(x)$,

$$\iota_C(x) = \begin{cases} 0, & x \in C, \\ +\infty, & x \notin C, \end{cases}$$

then

$$\operatorname{prox}_{\iota_C, t}(x) = \arg\min_{z \in C} \|z - x\|_2^2 = P_C(x),$$

where $P_C(x)$ is the projection of x into the convex set C.

If we assume that g is L-smooth and we fix a constant step size $t^{(k)} \leq 1/L$, then proximal gradient descent has a convergence rate of $O(1/k)$, matching the rate of gradient descent. Nevertheless, because PGD involves computing the proximal operator, it has greater computational complexity.

Stochastic gradient descent

Nowadays, stochastic algorithms for gradient descent are dominant in the optimization for machine learning scene. These algorithms reduce the number of iterations needed to approximate a stationary point and have reduced memory requirements (KINGMA; BA, 2014).

Let

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

be an average of function and consider the minimization problem of f

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

From the linearity of gradients,

$$\nabla \left(\sum_{i=1}^{n} f_i(x) \right) = \sum_{i=1}^{n} \nabla f_i(x),$$

the gradient descent iterates of (2.6) read

$$x^{(k)} = x^{(k-1)} - t^{(k)} \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x), \quad k = 1, 2, \ldots \tag{2.10}$$

The central idea of stochastic gradient descent (SGD) is to replace the full gradient average in (2.10) by a realization of an unbiased estimator of this quantity. In mini-batch stochastic gradient descent, for instance, at each iteration, we sample a uniformly random subset $B_k \subset [n]$, with $|B_k| = b \ll n$, where b is called batch-size, and estimate the full gradient as

$$\left[\frac{1}{b} \sum_{i \in B_k} \nabla f_i(x) \right], \tag{2.11}$$

which has the property that

$$\mathbb{E}\left[\frac{1}{b}\sum_{i\in B_k}\nabla f_i(x)\right]=\nabla f(x),$$

so that (2.11) is, in fact, an unbiased estimator of the full gradient.

If we combine the gradient descent iterations with the gradient estimation procedure, we get the full stochastic gradient algorithm:

1. Fix an initial point $x^{(0)}, t^{(0)}$.
2. For $k = 1, 2, \dots,$ repeat:
   a) Get B_k with $|B_k| = b$ uniformly from $[n]$;
   b) Update t_k according to the step size policy;
   c) $x^{(k)} = x^{(k-1)} - t_k\frac{1}{b}\sum_{i\in B_k}\nabla f_i\left(x^{(k-1)}\right)$
   d) Check if the stopping criterion is met.

Observe that SGD integrates two separate components: a gradient estimation method and the gradient descent algorithm. Meantime, the gradient estimation is decoupled from the descent algorithm and can be employed for other optimization procedures relying on the computation of gradients, the proximal gradient method of (2.9) alike (NITANDA, 2014). In fact, our proposed optimization algorithm, as described in Chapter 3, uses the proximal gradient descent method with stochastic gradients.

2.3 Kernel methods

Among the plethora of machine learning models, kernel-based methods are extensively used for their ability to model nonlinear dependencies between the input and output spaces (HOFMANN; SCHÖLKOPF; SMOLA, 2008). Besides that, kernel methods elicit "natural" functions spaces, known as Reproducing Kernel Hilbert Spaces (RKHS), with interesting computational and statistical properties that make these methods well-grounded.

In this section, we propose a progressive construction of kernel methods, building upon feature maps[1] and kernelization (SCHÖLKOPF; SMOLA, 2002). We then describe the formal characterization of an RKHS and elucidate the aforementioned properties of this space. Throughout this section, we try to emphasize the origin of the scalability and interpretability limitations of kernel methods and how they can be addressed.

[1] Also known as basis transformations (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Feature maps

Machine learning and statistics have been very well-developed for linear models. Still, real-world data analysis problems usually require nonlinear methods to detect the existent associations between input and output spaces. Yet, representing functions between $\mathcal{X}$ and $\mathcal{Y}$ as linear models is convenient and, sometimes, a necessary approximation.

The core idea of feature maps is to augment the vector of features X with additional variables, which are transformations of X, that hopefully capture the nonlinear associations present in data. This process generates a transformed input space $\mathcal{X}'$, where we then apply linear models.

Let us define $\phi_i : \mathbb{R}^p \to \mathbb{R}$ the ith feature map of X, $i \in [s]$. For instance, $\phi_1(X) = X_1$, $\phi_1(X) = X_1 X_2 X_3$ or $\phi_1(X) = \log X_1$ are valid candidates. Additionally, define $\phi : \mathbb{R}^d \to \mathbb{R}^s$ as

$$\phi(X) = (\phi_1(X), \dots, \phi_p(X)) \, ,$$

the vector feature map or concatenation of feature maps.

In the spirit of section 2.1, we then define the function space

$$\mathcal{F} = \{f : f(x) = \beta^\intercal \phi(x), \beta \in \mathbb{R}^p\} \tag{2.12}$$

of linear models in the extended features $\phi_1(X), \dots, \phi_p(X)$. For instance, if $\phi_i(X) = X_i$ for all $i = 1, \dots, d$, we recover the space of linear models in the original input space $\mathcal{X}$.

Often, the choice of feature maps aims to improve the flexibility of the functions in $\mathcal{F}$, increasing its capacity to approximate mappings between $\mathcal{X}$ and $\mathcal{Y}$. However, to account for the possibility of over-fitting, we also need to constrain the growth of the capacity of $\mathcal{F}$ via regularization. In this space, a convenient choice of regularization function is $\mathcal{R}(f) = \|\beta\|_2^2$, an ℓ_2 regularization on the coefficients.

Regularized loss minimization with feature maps

Now, consider the optimization problem of (2.3) within the space $\mathcal{F}$ defined in (2.12) and a with $R(f) = \|\beta\|_2^2$. We restrict our attention to the squared error loss $\ell(f(x_i), y_i) = (y_i - f(x_i))^2$. In this setting, the problem (2.3) reads

$$f^* \in \operatorname*{arg\,min}_{f \in \mathcal{F}} \left[\sum_{i=1}^{n} (y_i - f(x_i))^2 + \mu \mathcal{R}(f) \right], \tag{2.13}$$

which can be rewritten, in terms of the coefficients, as

$$\beta^* \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p} \left[\sum_{i=1}^{n} (y_i - \beta^\intercal \phi(x_i))^2 + \mu \|\beta\|_2^2 \right], \tag{2.14}$$

with $\mu > 0$. Defining the design matrix $\mathbf{\Phi}$ of the augmented features as

$$\mathbf{\Phi} = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_p(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_n) & \cdots & \phi_p(x_n) \end{pmatrix},$$

the problem (2.14) can be restated as

$$\beta^* \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p} \left[\|y - \mathbf{\Phi}\beta\|_2^2 + \mu\|\beta\|_2^2 \right], \tag{2.15}$$

where $y = (y_1, \ldots, y_n)^\intercal$ is the vector of responses in the training sample.

The solution β^* to (2.15) is the well-known ridge estimator,

$$\beta^* = (\mathbf{\Phi}^\intercal\mathbf{\Phi} + \lambda I_p)^{-1}\mathbf{\Phi}^\intercal y, \tag{2.16}$$

where I_p is the $p \times p$ identity matrix. The ridge estimator of (2.16) is in a so-called primal form. By using the matrix identity (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018, Lemma 11.12)

$$(\mathbf{\Phi}^\intercal\mathbf{\Phi} + \lambda I_p)^{-1}\mathbf{\Phi}^\intercal = \mathbf{\Phi}^\intercal(\mathbf{\Phi}\mathbf{\Phi}^\intercal + \lambda I_n)^{-1},$$

we can rewrite (2.16) in its dual-form[2],

$$\beta^* = \mathbf{\Phi}^\intercal(\mathbf{\Phi}\mathbf{\Phi}^\intercal + \lambda I_n)^{-1}y, \tag{2.17}$$

which is associated with the optimal function

$$f^*(\cdot) = \beta^{\intercal *}\phi(\cdot) = y^\intercal(\mathbf{\Phi}\mathbf{\Phi}^\intercal + \lambda I_n)^{-1}\mathbf{\Phi}^\intercal\phi(\cdot). \tag{2.18}$$

Notice that the terms of the matrix $\mathbf{\Phi}\mathbf{\Phi}^\intercal$ have the form

$$(\mathbf{\Phi}\mathbf{\Phi}^\intercal)_{ij} = \sum_{k=1}^{p} \mathbf{\Phi}_{ik}\mathbf{\Phi}_{jk} = \phi(x_i)^\intercal\phi(x_j),$$

and, similarly,

$$\mathbf{\Phi}^\intercal\phi(\cdot) = (\phi(x_1)^\intercal\phi(\cdot), \ldots, \phi(x_n)^\intercal\phi(\cdot))^\intercal.$$

That is, the vector feature map enters the solution of the estimate in (2.18) only through inner products, having no standalone role.

Remarkably, for carefully chosen feature maps, these inner products have a closed-form representation that is much simpler to compute than the feature maps themselves, often involving computations only with the original features. This property is widely known in the machine learning community as the “kernel trick”. We give an example of this phenomenon in the next example.

[2] This nomenclature arises from duality theory in optimization (BOYD; VANDERBERGHE, 2004).

Example 4. Let $\phi : \mathbb{R}^d \to \mathbb{R}^p$, with $p = d + \binom{d}{2}$, be a vector feature map defined as

$$\phi(x) = \begin{bmatrix} x_i^2, & i = 1, \ldots, d \\ \sqrt{2} x_i x_j, & i < j \end{bmatrix}.$$

Then

$$\phi(x)^\intercal \phi(y) = (x^\intercal y)^2,$$

that is, the inner product of the basis functions can be computed from the original features $x, y \in \mathbb{R}^d$ without explicitly performing the computations related to the feature maps.

Kernelization

Many machine learning algorithms have optimization problems or closed-form solutions for the optimal estimates that can be written solely in terms of the inner products of feature maps. As we discussed, however, the feature maps enter the solution with a secondary role: the central object is the expression, often easier to compute, that emerges from their inner product.

In kernel methods, guided by the previous observation, one proceeds reversely. Instead of building feature maps and augmenting feature spaces, one defines an expression for the inner product specified only in terms of the original features. This expression must satisfy some prerequisites, which specify a positive semidefinite kernel function.

Definition 5 (Positive semidefinite kernel function)**.** A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semidefinite (PSD) kernel if k is symmetric and for all $n \geq 1$ and elements $\{x_i : i \in [n]\} \subset \mathcal{X}$, the $n \times n$ matrix with elements $\mathbf{K}_{ij} = k(x_i, x_j)$ is positive semidefinite.

Observe that, as expected, inner products of vector feature maps are a particular case of positive semidefinite kernels.

Proposition 3. Let $\phi : \mathcal{X} \to \ell^2(\mathbb{N})$ be a vector feature map. Then the function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined as

$$k(x, y) = \phi(x)^\intercal \phi(y) \tag{2.19}$$

is a positive semidefinite kernel.

Proof. First, k is symmetric, as $k(x, y) = \phi(x)^\intercal \phi(y) = \phi(y)^\intercal \phi(x) = k(y, x)$. Now, let $\{x_i : i \in [n]\} \subset \mathcal{X}$ and $\alpha \in \mathbb{R}^n$. Then,

$$\begin{aligned} \alpha^\intercal \mathbf{K} \alpha = \sum_{i,j=1}^n \alpha_i \alpha_j k(x_i, x_j) = \sum_{i,j=1}^n \alpha_i \alpha_j \phi(x_i)^\intercal \phi(x_j) &= \sum_{i,j=1}^n \alpha_i \phi(x_i)^\intercal \alpha_j \phi(x_j) \\ &= \left\| \sum_{i=1}^n \alpha_i \phi(x_i) \right\|_2^2 \geq 0. \end{aligned}$$

Since $\{x_i : i \in [n]\}$ and $\alpha \in \mathbb{R}^n$ are arbitrary, we conclude that k is a PSD kernel. □

In effect, the correspondence between PSD kernels and vector feature maps is a very rigorous result in the theory of kernel methods. Mercer's Theorem (MERCER, 1909) guarantees that every PSD kernel implicitly defines a (possibly infinite) vector feature map such that this kernel is of the form stated in (2.19).

This property lets us interpret kernels as representations of the inner product of vector feature maps in the original input space. In this sense, every algorithm which can be written solely in terms of inner products of the features can be "kernelized", that is, have the inner products substituted by an appropriate kernel.

Construction of kernel methods

We construct the function space associated with kernel methods in the spirit of section 2.1. Although abstract, this construction is general and allows us to define these methods precisely. We base our approach on Wainwright (2019), albeit the original theory was developed by Aronszajn (1950).

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PSD kernel. Consider the space $\mathcal{H}'$ defined as

$$\mathcal{H}' = \left\{ \sum_{i=1}^{n} \alpha_i k(x_i, \cdot) : \alpha \in \mathbb{R}^n, (x_i)_{i \in [n]} \subset \mathcal{X} \right\}, \tag{2.20}$$

which is the linear span of the kernel function partially evaluated in points of the input space $\mathcal{X}$.

$\mathcal{H}'$ is a vector space with the usual notions of sum and scalar multiplication of functions. Let $f, g \in \mathcal{H}'$, then $f(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)$ and $g(\cdot) = \sum_{j=1}^{m} \alpha'_j k(x'_j, \cdot)$. We endow $\mathcal{H}'$ with the inner product

$$\langle f, g \rangle_{\mathcal{H}'} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \alpha'_j k(x_i, x'_j), \tag{2.21}$$

which can be proven independent of the choice of representation for f and g.

Notice that, from (2.21),

$$\langle f, k(x, \cdot) \rangle_{\mathcal{H}'} = \sum_{i=1}^{n} \alpha_i k(x_i, x) = f(x),$$

which is known as the *reproducing property*, since $k(x, \cdot)$ acts as the reproducer of the evaluation functional $L_x(f) = f(x)$. Since all functions in $\mathcal{H}'$ are linear combinations of the kernel function, this property means that inner products in $\mathcal{H}'$ are reduced to kernel evaluations.

The inner product space $(\mathcal{H}', \langle \cdot, \cdot \rangle_{\mathcal{H}'})$ is not a Hilbert space: it remains to extend $\mathcal{H}'$ to a complete inner product space with the given reproducing kernel. By properly defining the limits of functions in $\mathcal{H}'$, one can complete $\mathcal{H}'$ in such a manner that the resulting space $\mathcal{H} \supset \mathcal{H}'$ have k as a reproducing kernel. Additionally, the space $\mathcal{H}$ is the unique Hilbert space with k as reproducing kernel (WAINWRIGHT, 2019).

Regularized loss minimization in RKHS: the scalability problem

In section 2.3, we mentioned that RKHS has interesting computational and statistical properties. The first of these properties is the kernel trick, which exempts us from explicitly calculating high-dimensional feature maps using PSD kernels that act only in the original features.

The second property is that solutions for the Regularized Loss Minimization (RLM) procedure with $\mathcal{H}$ as an RKHS admits a simple representation in $\mathcal{H}$. In what follows, we state this property, known as the Representer's Theorem, and discuss its consequences for the scalability of kernel methods.

Theorem 4 (Representer's theorem)**.** Let $\mathcal{X}$ be the input space, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R} \cup \{+\infty\}$ an arbitrary loss function, and $\mathcal{H}$ the unique RKHS associated to the PSD kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Then

$$f^* \in \arg\min_{f\in\mathcal{H}} \sum_{i=1}^{n} \ell(f(x_i), y_i) + \mu||f||^2_{\mathcal{H}}, \quad \mu > 0, \tag{2.22}$$

admits a representation of the form

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot), \quad \alpha_i \in \mathbb{R}. \tag{2.23}$$

Proof. See, for instance, Wainwright (2019) or Schölkopf and Smola (2002). □

Notice that, despite involving an infinite-dimensional optimization in $\mathcal{H}$, the solution to the RLM procedure is reduced to a n-dimensional optimization problem, which can be tackled numerically.

In fact, since $f^* \in \mathcal{H}'$, then

$$\|f^*\|^2_{\mathcal{H}} = \langle f^*, f^* \rangle_{\mathcal{H}'} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i k(x_i, x_j)\alpha_j = \alpha^\intercal \mathbf{K}\alpha,$$

which prompts us to rewrite (2.22) as the problem

$$\alpha^* \in \arg\min_{\alpha\in\mathbb{R}^n} \sum_{i=1}^{n} \ell(f^*(x_i), y_i) + \mu\alpha^\intercal \mathbf{K}\alpha.$$

We explore this result in the following two examples.

Example 5 (Minimal norm interpolation)**.** Let the loss function be

$$\ell(f(x_i), y_i) = \begin{cases} 0, & f(x_i) = y_i, \\ +\infty, & \text{otherwise.} \end{cases}$$

With this choice, (2.22) can be written as a constrained optimization problem

$$f^* \in \arg\min_{f\in\mathcal{H}} \|f\|^2_{\mathcal{H}} \qquad \text{such that } f(x_i) = y_i \text{ for } i \in [n].$$

Let $\mathbf{K}$ be the kernel matrix of the kernel evaluated on the predictors of the training sample $\{x_i : i \in [n]\}$, that is $\mathbf{K}_{ij} = k(x_i, x_j)$. In this case, if $\{y_i : i \in [n]\} \in \operatorname{range}(\mathbf{K})$ and $\mathbf{K}$ is invertible, then the optimal solution is

$$f^*(x) = y^\intercal \mathbf{K}^{-1} k',$$

with $k' = (k(x_1, x), \ldots, k(x_n, x))^\intercal$.

Example 6 (Kernel Ridge Regression (KRR))**.** Let the loss function be

$$\ell(f(x_i), y_i) = (y_i - f(x_i))^2,$$

then the optimal solution is

$$f^*(x) = y^\intercal (\mathbf{K} + \lambda \mathbf{I})^{-1} k'.$$

Observe that the optimal function f^*, for both Example 5 and Example 6, depends on the inversion of the kernel matrix, an operation with space complexity of $O(n^2)$ and time complexity of $O(n^3)$, where n is the training sample size. Thus, the inversion becomes prohibitive for large datasets: this is the origin of the kernel methods' *scalability* problem.

Interpretability with ARD kernels

In this section, we try to build intuition on the aspect of functions in $\mathcal{H}'$, which contains the solution of the regularized minimization problem in RKHS, as discussed in section 2.3. We introduce some examples of ARD kernels and show that these kernels can control a feature's influence on the values assumed by functions belonging to the associated RKHS $\mathcal{H}'$. We postpone a formal definition of ARD kernels to Chapter 3.

Let $\lambda = (\lambda_1, \ldots, \lambda_p)$ be a vector of relevances. Core examples of ARD kernels are:

- Gaussian kernel:
$$k_\lambda(x, y) = \exp\left[-\frac{1}{2}\sum_{i=1}^{p} \lambda_i^2 (x_i - y_i)^2\right], \tag{2.24}$$
- Laplacian kernel:
$$k_\lambda(x, y) = \exp\left[-\sum_{i=1}^{p} \lambda_i |x_i - y_i|\right], \tag{2.25}$$
- Cauchy kernel:
$$k_\lambda(x, y) = \prod_{i=1}^{p} \frac{2}{1 + \lambda_i^2 (x_i - y_i)^2}. \tag{2.26}$$

Notice that by fixing $\lambda_k = 0$, we make the k-th feature of the vector x irrelevant to the determination of the value of the kernel. If we consider that functions in $\mathcal{H}'$ are linear combinations of the kernel, we are ultimately making $f \in \mathcal{H}'$ irrelevant to that feature. For instance, if $\mathcal{X} \subset \mathbb{R}$ and we set $\lambda = 0$, then any $f \in \mathcal{H}'$ will be a constant function, indicating that this single feature in $\mathcal{X}$ is not relevant to determine the predicted value. In this sense, ARD kernels allow us to control which features of X should be considered active for the prediction functions we wish to build.

In the case $\mathcal{X} \subset \mathbb{R}$, we visually explore this fact in Figure 1 for the aforementioned ARD kernels. For visualization, we constructed a function $f \in \mathcal{H}'$ as

$$f(\cdot) = \sum_{i=1}^{20} \alpha_i k_\lambda(x_i, \cdot),$$

where $\alpha_i \sim \mathsf{Uniform}(-1, 1)$ and $x_i = i/20$.

As mentioned, when the relevance parameter λ is set to $\lambda = 0$, the functions in the $\mathcal{H}'$ associated with the Gaussian, Cauchy, and Laplacian kernels, become constant, showing no dependence on the only available feature. However, as λ increases, we observe that the functions become dependent on the feature and exhibit greater nonlinearity (mainly close to the training points).

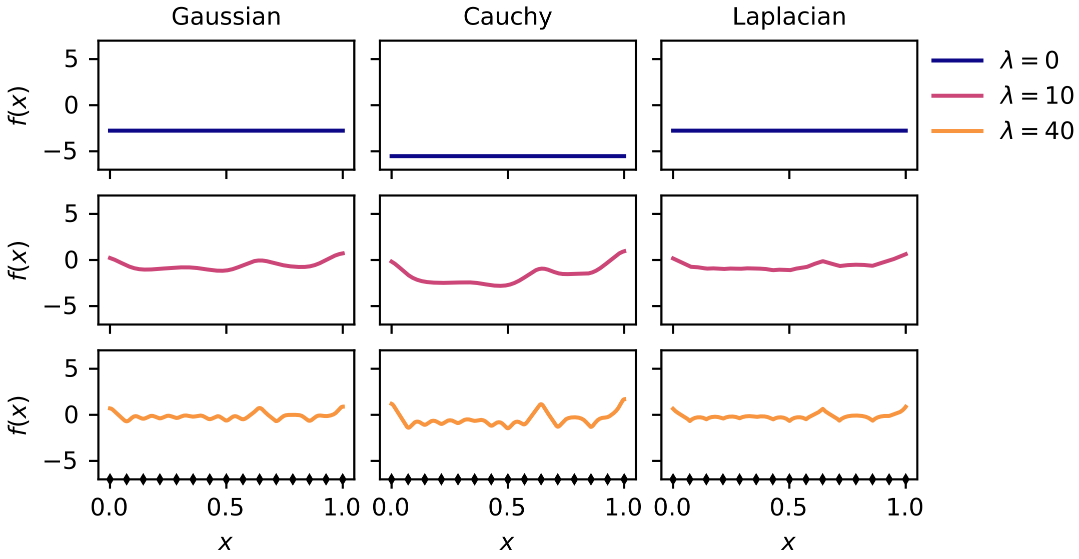


Figure 1 – Visual exploration of functions in the space $\mathcal{H}'$ elicited by one-dimensional ARD kernels. When $\lambda = 0$, the functions in $\mathcal{H}'$ exhibit no dependence on x. However, when λ increases, f becomes dependent on the feature and exhibits greater nonlinearity, changing in smaller length scales.

In that regard, adjusting the relevances of ARD kernels may eliminate the effect of non-informative features, increasing the kernel methods' interpretability. Even more, if we allow these relevances to be fitted in a data-dependent manner, then it may be possible that the relevances will be different from zero only for the features that allow for accurate prediction of the responses. This last observation guides our approach in Chapter 3.

2.4 Random Fourier features

In section 2.3, we explicitly showed that kernel methods have a scalability limitation. This was related to kernel methods relying on storage and inversion of the kernel matrix, which is prohibitive for large datasets.

In this context, many approaches for scalable kernel methods emerged (RAHIMI; RECHT, 2008a; LE; SARLOS; SMOLA, 2014; CURTó *et al.*, 2017). We delve into the Random Fourier Features (RFF) framework due to Rahimi and Recht (2008a). Roughly speaking, the idea of random Fourier features is to construct a vector feature map $z : \mathcal{X} \to \mathbb{R}^s$ whose inner products approximate a PSD kernel of interest.

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PSD kernel. We seek to construct a vector feature map $z : \mathcal{X} \to \mathbb{R}^s$, with $s \ll n$, such that

$$k(x, y) \approx z(x)^\intercal z(y).$$

In this situation, the optimal solution of regularized loss minimization problem in RKHS (2.23) could be written as

$$\begin{aligned} f^*(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot) &\approx \sum_{i=1}^{n} \alpha_i \, z(x_i)^\intercal z(\cdot) \\ &= \underbrace{\left(\sum_{i=1}^{n} \alpha_i z(x_i)\right)^\intercal}_{\beta} z(\cdot) \\ &= \sum_{j=1}^{s} \beta_j z_j(\cdot), \end{aligned} \tag{2.27}$$

where $\beta \in \mathbb{R}^s$. That is, the optimal solution involves only finding the s components of β rather than the n components of α. Consequently, the scalability problem is settled if $s \ll n$.

The construction of RFF is based on Bochner's theorem.

Theorem 5 (Bochner's)**.** Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a bounded and continuous PSD kernel. Additionally, suppose that k is shift-invariant; that is, there exists h such that $k(x, y) = h(x - y)$. Then $k(x, y)$ is the Fourier transform of a bounded positive measure.

Proof. See Rudin (2017). □

The measure in Theorem 5 is called the spectral measure of the kernel. It is usually assumed that this measure admits a density function $p(\cdot)$. In this case, from Theorem 5, the kernel can be written as

$$\begin{aligned} k(x,y) = h(x-y) &= \int_{\mathcal{W}} e^{i\omega^\intercal (x-y)} p(\omega) d\omega \\ &= \mathbb{E}_\omega \Big[\phi(x)\phi(y)^*\Big], \end{aligned} \tag{2.28}$$

where $\phi : \mathcal{X} \to \mathbb{C}$ is a complex feature map defined as $\phi(x) = e^{i\omega^\intercal x}$ and $\mathcal{W}$ is the support of the density p.

In the following theorem, we show how to construct the vector feature map that approximates PSD kernels that satisfy the conditions of Theorem 5.

Theorem 6 (Random Fourier features)**.** Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a bounded, continuous and shift-invariant kernel. Assume that the spectral measure of the kernel admits a density function $p(\cdot)$. Let $\omega_1, \ldots, \omega_s \sim p(\omega)$ and $b_1, \ldots, b_s \sim \mathsf{Uniform}(0, 2\pi)$. Define the feature map $z_i : \mathcal{X} \to \mathbb{R}$ as

$$z_i(x) = \sqrt{2}\cos(\omega_i^\intercal x + b_i) \tag{2.29}$$

and the vector feature map $z : \mathcal{X} \to \mathbb{R}^s$ as

$$z(x) = \frac{1}{\sqrt{s}} \begin{bmatrix} z_1(x) \\ \vdots \\ z_s(x) \end{bmatrix}. \tag{2.30}$$

Then

$$z(x)^\intercal z(y) \xrightarrow[s \longrightarrow \infty]{\text{a.s.}} k(x,y). \tag{2.31}$$

Proof. First, from the fact that the kernel is real-valued, that is, Im $k = 0$, we get

$$\begin{aligned} k(x,y) &= \mathbb{E}_\omega \Big[\phi(x)\phi(y)^*\Big] \\ &= \mathbb{E}_\omega \Big[\cos\left(\omega^\intercal (x-y)\right)\Big]. \end{aligned}$$

Now, observe that

$$\begin{aligned} \mathbb{E}_{\omega,b}\left[z_i(x) z_i(y)\right] &= \mathbb{E}_{\omega,b}\left[2\cos(\omega_i^\intercal x + b_i)\cos(\omega_i^\intercal y + b_i)\right] \\ &= \mathbb{E}_{\omega}\left[\cos\left(\omega_i^\intercal (x-y)\right)\right] + \mathbb{E}_{\omega,b}\left[\cos\left(\omega_i^\intercal (x+y) + 2b_i\right)\right]. \end{aligned}$$

We can rewrite the second expectation on the right-hand side as

$$\mathbb{E}_{\omega,b}\left[\cos\left(\omega_i^\intercal (x+y) + 2b_i\right)\right] = \mathbb{E}_{\omega}\left\{\mathbb{E}_b\left[\cos\left(\omega_i^\intercal (x+y) + 2b_i\right) \middle|\, \omega\right]\right\}$$

and compute the inner expectation

$$\begin{aligned} \mathbb{E}_b\left[\cos\left(\omega_i^\intercal (x+y) + 2b_i\right) \middle|\, \omega\right] &= \frac{1}{2\pi}\int_0^{2\pi} \cos\left(\omega_i^\intercal (x+y) + 2a\right) da \\ &= \frac{1}{4\pi} \sin\left(\omega_i^\intercal (x+y) + 2a\right)\Big|_0^{2\pi} \\ &= 0. \end{aligned}$$

Thus, we get

$$\mathbb{E}_\omega\left[z_i(x)z_i(y)\right] = \mathbb{E}_\omega\left[\cos\left(\omega_i^\intercal(x-y)\right)\right] = k(x,y).$$

Finally, let $\omega_1,\dots,\omega_s \sim p(\omega)$ and $b_1,\dots,b_s \sim \mathsf{Uniform}(0,2\pi)$. The random variable $\bar{k}$ defined as

$$\bar{k}(x,y) = z(x)^\intercal z(y) = \frac{1}{s}\sum_{i=1}^{s} z_i(x)z_i(y)$$

is an unbiased estimator of $k(x,y)$. From the strong law of large numbers, we conclude that

$$\langle z(x), z(y)\rangle \xrightarrow[s\longrightarrow\infty]{\text{a.s.}} k(x,y),$$

as wanted. □

Examples of kernels and the corresponding densities of their spectral measures are given in Table 1.

Table 1 – Usual shift-invariant PSD kernels and the densities of their corresponding spectral measures.

	$k(x,y)$	$p(\omega)$
Gaussian	$\exp\left[-\frac{1}{2}\|x-y\|_2^2\right]$	$(2\pi)^{-\frac{p}{2}}\exp\left[-\frac{\|\omega\|_2^2}{2}\right]$
Laplacian	$\exp\left[-\|x-y\|_1\right]$	$\prod_{i=1}^{p}\frac{1}{\pi(1+\omega_i^2)}$
Cauchy	$\prod_{i=1}^{p}\frac{2}{1+(x_i-y_i)^2}$	$\exp\left[-\|\omega\|_1\right]$

The asymptotic nature of Theorem 6 cannot guide us in choosing the dimension s of the random features to guarantee a good approximation of the kernel. In the original article (RAHIMI; RECHT, 2008a), and in follow-up papers (SUTHERLAND; SCHNEIDER, 2015), it is shown, using a uniform approximation bound, that the number of features s must scale linearly with the number of training samples; that is $s \in \Omega(n)$. This is a very pessimistic bound, as it tells that the regime $s \ll n$, where RFF can solve the scalability problem of kernel methods, is unfeasible. However, empirical evidence suggested that these bounds conflicted with the practical success of RFF in many situations.

The first point to derive sensible bounds for learning with RFF is to realize that bounding the kernel matrix approximation is not central. Instead, one should study generalization bounds for learning with functions like (2.27). First attempts to derive such bounds (RAHIMI; RECHT, 2008b; BACH, 2015) still required a pessimistic number of features. But lately, (RUDI; CARRATINO; ROSASCO, 2017; LI *et al.*, 2021) proposed refined analyses that greatly improved the number of random features needed to produce such bounds. For instance, in a worst-case scenario (concerning the spectrum of the kernel function and choice of the regularization strength), Li *et al.* (2021) showed that if

$s \geq \sqrt{n} \log n$ (which is much smaller than n), then learning with random features can attain the minimax rate of $O(1/\sqrt{n})$, matching the rate of learning with the original kernel function (CAPONNETTO; VITO, 2007).

CHAPTER 3

RFFNET

In this chapter, we describe RFFNet, our approach for scalable and interpretable kernel methods, leveraging ARD kernels and the framework of random Fourier features described in section 2.3 and section 2.4, respectively. We formally define ARD kernels and show how to assemble a random Fourier feature map that is independent of the feature relevances. We introduce the objective function for RFFNet and propose an optimization procedure to minimize it. We conclude by giving directions for extensions of RFFNet.

3.1 Random Fourier features for ARD kernels

ARD kernels are frequently used for kernel-based learning methods, such as Gaussian Processes (RASMUSSEN; WILLIAMS, 2006; DANCE; PAIGE, 2021), and support-vector machines (KEERTHI; SINDHWANI; CHAPELLE, 2006). These kernels are constructed by associating a weight or relevance to each feature in a shift-invariant kernel. Generally, ARD kernels have the form

$$k_\lambda(x, y) = h[\lambda \circ (x - y)], \tag{3.1}$$

where the vector $\lambda \in \mathbb{R}^p$ is the vector of weights or relevances, which controls a feature's influence on the kernel's value, h is a continuous function, and $\circ$ is the element-wise multiplication. The absolute value of the i-th entry in λ is used to measure the importance of the i-th feature for the predictive task.

The following propositions are the fundamental results required to construct random Fourier features for ARD kernels. First, we show that the spectral measure of an ARD kernel k_λ can be written in terms of the spectral measure of $k_{\mathbf{1}_p}$, where $\mathbf{1}_p = (1, \ldots, 1) \in \mathbb{R}^p$; that is, when all features are considered equally relevant.

Proposition 4. Let $k_\lambda : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ be a bounded, continuous, and shift-invariant kernel of the form (3.1). Then, the density p_λ of the spectral measure of k_λ satisfies

$$p_\lambda(\omega) = \frac{1}{|\lambda_1 \cdots \lambda_p|} p\left(\omega \circ \frac{1}{\lambda}\right),$$

with $p(\cdot)$ the density of the spectral measure of the kernel with $\lambda = \mathbf{1}_p$. Additionally, $p(\omega) = p(-\omega)$ for all $\omega \in \mathbb{R}^p$.

Proof. See section A.3. □

Proposition 4 also reveals that p_λ depends on λ in a very particular manner: the relevances appear as a vector scale parameter for the density p associated with the equally weighted version of the kernel. In this sense, if $\omega \sim p(\cdot)$, then $\lambda \circ \omega \sim p_\lambda(\cdot)$. That is, if we want to sample from p_λ, we can first sample from p and then scale the sample by λ[1].

Proposition 5 uses this last observation to show that the RFF map that approximates an ARD kernel does not depend on λ. In fact, we can construct the RFF associated with the ARD kernel with $\lambda = \mathbf{1}_p$ and obtain an approximation for the ARD kernel with arbitrary λ by merely introducing a data scaling by λ.

Proposition 5. Let $k_\lambda : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a bounded, continuous, and shift-invariant kernel of the form (3.1). Let $z : \mathbb{R}^p \to \mathbb{R}^s, s \geq 1$, be the RFF for $k_{\mathbf{1}_p}$. Then,

$$\bar{k}_\lambda(x, y) = z(\lambda \circ x)^\intercal z(\lambda \circ y), \tag{3.2}$$

is an unbiased estimator of $k_\lambda(x, y)$.

Proof. By Proposition 4, if we sample $\omega \sim p(\cdot)$, then $\lambda \circ \omega \sim p_\lambda(\cdot)$. Now, by Bochner's theorem, with $\omega' \sim p_\lambda(\cdot)$,

$$\begin{aligned} k_\lambda(x, y) &= \mathbb{E}_{\omega'}[\cos \omega'^\intercal (x - y)] \\ &= \mathbb{E}_\omega[\cos (\lambda \circ \omega)^\intercal (x - y)] \\ &= \mathbb{E}_\omega[\cos \omega^\intercal \left(\lambda \circ (x - y)\right)], \end{aligned}$$

where $\omega \sim p(\cdot)$. Now, from Theorem 6, if $z : \mathbb{R}^p \to \mathbb{R}^s$ is the RFF for $k_{\mathbf{1}_p}$, then

$$\mathbb{E}_\omega[\cos \omega^\intercal \left(\lambda \circ (x - y)\right)] = \mathbb{E}_\omega[z(\lambda \circ x)^\intercal z(\lambda \circ y)],$$

or equivalently,

$$k_\lambda(x, y) = \mathbb{E}_\omega[z(\lambda \circ x)^\intercal z(\lambda \circ y)],$$

which shows that

$$\bar{k}_\lambda(x, y) = z(\lambda \circ x)^\intercal z(\lambda \circ y)$$

is an unbiased estimator of $k_\lambda(x, y)$. □

[1] In the same sense that, if we want to sample $X \sim \mathsf{Normal}(0, \sigma^2)$, we first sample $Z \sim \mathsf{Normal}(0, 1)$, then multiply the realization of Z by σ.

In RFFNet, we make the assumption that

$$p(\omega) = \prod_{i=1}^{p} f_i(\omega_i), \tag{3.3}$$

where f_i are symmetric densities, i.e. $f_i(\omega_i) = f_i(-\omega_i)$ for all $\omega_i \in \mathbb{R}$. In this case, Proposition 4 implies that

$$p_\lambda(\omega) = \prod_{i=1}^{p} \frac{1}{|\lambda_i|} f_i\left(\frac{\omega_i}{\lambda_i}\right),$$

which indicates that each relevance parameter enters as a real scale parameter of the spectral measure of the kernel k_λ. Crucially, since $p_\lambda(\omega)$ is symmetric by exchanging any λ_i with $-\lambda_i$, $i = 1, \dots, p$, the positive or negative versions of λ_i encode the same relevance information of the i-th feature.

This assumption has a direct consequence on the optimization procedure. Because each λ_i may assume both positive and negative values, we don't need to introduce any constraints on λ during optimization. This is a distinctive aspect of our approach: current methods for feature selection or feature importance in kernel methods (BROUARD *et al.*, 2022; JORDAN; LIU; RUAN, 2021; RUAN; LIU; JORDAN, 2021; GREGOROVÁ *et al.*, 2018) consider that relevance variables must be strictly positive and constrain its values during optimization, which increase the computational cost.

We clarify these results in the following example.

Example 7 (Gaussian kernel)**.** Let $k_\lambda : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be the ARD gaussian kernel

$$k_\lambda(x, y) = \exp\left[-\sum_{i=1}^{p} \frac{\lambda_i^2}{2}(x_i - y_i)^2\right].$$

Then,

$$k_{\mathbf{1}_p}(x, y) = \exp\left[-\frac{1}{2}\|x - y\|_2^2\right],$$

which is the isotropic gaussian kernel shown in Table 1. According to Table 1, the spectral measure of $k_{\mathbf{1}_p}$ is

$$p(\omega) = (2\pi)^{-\frac{p}{2}} \exp\left[-\frac{\|\omega\|_2^2}{2}\right] = \prod_{i=1}^{p} (2\pi)^{-\frac{1}{2}} \exp\left[-\frac{\omega_i^2}{2}\right],$$

which is a product of standard normal densities, with the same form of (3.3). Evidently, $\omega \sim \mathsf{Normal}(0, I_p)$. Hence Proposition 4 implies that the spectral density of k_λ is

$$p_\lambda(\omega) = \prod_{i=1}^{p} (2\pi\lambda_i^2)^{-\frac{1}{2}} \exp\left[-\frac{\omega_i^2}{2\lambda_i^2}\right].$$

Now, Proposition 5 tells that, if we want to approximate k_λ via RFF, we first sample $\omega_1, \dots, \omega_s \sim \mathsf{Normal}(0, I_p)$ and $b_1, \dots, b_s \sim \mathsf{Uniform}(0, 2\pi)$ and define the RFF map $z : \mathbb{R}^p \to \mathbb{R}^s$ with $z_i(x) = \sqrt{2}\cos(\omega_i^\intercal x + b_i)$. Then $z(\lambda \circ x)^\intercal z(\lambda \circ y)$ approximates $k_\lambda(x, y)$.

Although the example is based on the ARD Gaussian kernel, since the spectral measures associated with the unweighted Laplace and Cauchy kernel also factor as (3.3) (see Table 1), the above results also holds for the ARD Laplace and ARD Cauchy kernel.

3.2 Approximate kernel machines

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PSD kernel and $z : \mathcal{X} \to \mathbb{R}^s$ be the random Fourier features that approximate k. Since z is a feature map, we can define the approximate kernel $k_z : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as

$$k_z(x, y) = z(x)^\intercal z(y) \tag{3.4}$$

which is a valid PSD kernel, according to Proposition 3. This kernel is uniquely associated with an RKHS, which we denote as $\mathcal{H}_z$. An approximate kernel machine is a solution for the regularized loss minimization in this RKHS,

$$f^* \in \arg\min_{f \in \mathcal{H}_z} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) + \mu ||f||^2_{\mathcal{H}_z}, \quad \mu > 0,$$

which admits, from Theorem 4, a representation of the form

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i k_z(x_i, \cdot) = \sum_{j=1}^{s} \beta_j z_j(\cdot). \tag{3.5}$$

Solving for the coefficients of f^* we get

$$\beta^* \in \arg\min_{\beta \in \mathbb{R}^n} \sum_{i=1}^{n} \ell(f^*(x_i), y_i) + \mu \beta^\intercal \mathbf{K} \beta, \quad \mu > 0, \tag{3.6}$$

where

$$\mathbf{K}_{ij} = z(x_i)^\intercal z(x_j).$$

Notice that finding the coefficients β^* involves the computation of the approximate kernel matrix $\mathbf{K}$, which appears in the regularization term and could potentially increase the computational cost of finding the estimates. Fortunately, we can regularize the solution avoiding kernel matrix computations. This reformulation was first proposed in Li *et al.* (2021), where the authors derived an upper bound for the regularization function of the optimization problem associated with the approximate kernel. The resulting regularization is equivalent to a ℓ_2 penalty on the space of random Fourier features, a result already present in Rahimi and Recht (2008a), although not discussed in detail.

Proposition 6. Assume that the reproducing kernel Hilbert space $\mathcal{H}_z$ with kernel K_z admits a decomposition as in (3.4). Define $\mathcal{H}'_z = \{\sum_{i=1}^{s} \alpha_i z_i(\cdot) : \alpha \in \mathbb{R}^s\}$. Then, for all $f \in \mathcal{H}'_z$ it holds that $\|f\|^2_{\mathcal{H}'_z} \leq s\|\alpha\|^2_2$.

Proof. See Li *et al.* (2021). □

With this result, instead of tackling the problem (3.6), we may find a solution to a surrogate version

$$\beta^* \in \arg\min_{\beta \in \mathbb{R}^n} \sum_{i=1}^{n} \ell(\beta^\intercal z(x_i), y_i) + \mu \|\beta\|_2^2, \quad \mu > 0, \tag{3.7}$$

which does not include the kernel matrix in the regularization term.

3.3 Overview of RFFNet

In section 3.1 and section 3.2, we saw that it is possible to:

1. Use random Fourier features that efficiently approximate ARD kernels, as in eq. (3.2);

2. Regularize the solution directly in the space of random features, as in eq. (3.7), without explicitly computing the approximate kernel matrix.

Based on these two aspects, we can now describe how RFFNet works. Given $\{(x_i, y_i)\}_{i=1}^{n}$ a training sample, RFFNet solves

$$\underset{\beta,\lambda}{\text{minimize}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, \beta^\intercal z(\lambda \circ x_i)\right) + \mu \|\beta\|_2^2, \tag{3.8}$$

where $z : \mathbb{R}^p \to \mathbb{R}^s$ is the random Fourier features map corresponding to an ARD kernel with $\lambda = \mathbf{1}_p$, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+ \cup \{+\infty\}$ is a loss function and $\mu > 0$ is the regularization strength.

RFFNet addresses the problems of scalability and interpretability, by

- **Scalability**: using a random Fourier features map $z : \mathbb{R}^p \to \mathbb{R}^s$ in the $s \ll n$ setting, where n is the training sample size;

- **Interpretability**: coupling the feature map z with the data scaling λ, which approximates an ARD kernel of interest and reduces the influence of irrelevant covariates.

3.4 Optimization algorithm

The objective function of RFFNet on eq. (3.8) is not convex due to the relevance vector λ inside the RFF map. Consequently, it is not liable to usual convex optimization procedures. Notwithstanding, depending on the loss functions, the objective has Lipschitz continuous gradients with respect to each block of coordinates. This is the case, for instance, for the squared error loss. For this reason, we used a block stochastic gradient descent algorithm (BOLTE *et al.*, 2014; POCK; SABACH, 2016; XU; YIN, 2014). For each block of coordinates, instead of a simple gradient descent iterate, we used the Adam optimizer (KINGMA; BA, 2014), which has moment estimation.

Let $H(\beta, \lambda)$ be the objective function of (3.8) without the regularization term,

$$H(\beta, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, \beta^\intercal z(\lambda \circ x_i)\right) + \mu \|\beta\|_2^2.$$

In Algorithm 1, we show a detailed description of the RFFNet algorithm to minimize H. We remark, however, that we replaced the Adam gradient iterates with simple gradient descent for clarity and conciseness.

Algorithm 1 – RFFNet training

```
procedure FIT(input: $X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n$, validation fraction $\rho$, learning rate $\eta$, regularization $\mu$, patience $K$, max epochs T, initialization $(\beta^{(0)}, \lambda^{(0)})$, density of spectral measure of unweighted kernel $p(\omega)$, number of random features $s$)

    Split data into training and validation samples (the last containing a fraction $\rho$ of all data samples).
    Generate random Fourier features map $z : \mathbb{R}^p \to \mathbb{R}^s$ sampling $s$ times from $p(\omega)$ and $\mathsf{Uniform}(0, 2\pi)$.
    for $t \in \{1, \ldots, T-1\}$ do
        $\beta^{(t+1)} \leftarrow \mathrm{prox}_{\mu\|\cdot\|_2^2,\alpha}\left(\beta^{(t)} - \eta \nabla_\beta H(\beta^{(t)}, \lambda^{(t)})\right)$
        $\lambda^{(t+1)} \leftarrow \lambda^{(t)} - \eta \nabla_\lambda H(\beta^{(t+1)}, \lambda^{(t)})$
        Apply early stopping policy (with patience $K$) based on the validation sample.
    end for
    Retrieve the model $\mathcal{M}$ that best performed on the validation sample.
    return $\mathcal{M}$
end procedure
```

3.5 Extensions

RFFNet can be seen as optimizing the parameters of a two-layer neural network, as depicted in Figure 2, where the connection between the RFF map and the output is done by a linear function. By replacing the link between the RFF map and the output with general neural network architectures, we expect to increase the method's predictive performance and, most importantly, generate a feature importance metric for neural networks based on the relevances of the ARD kernel. In the same spirit, RFFNet can be used as a layer for *any* neural network, and thus can be used in a huge variety of tasks, including unsupervised learning problems.

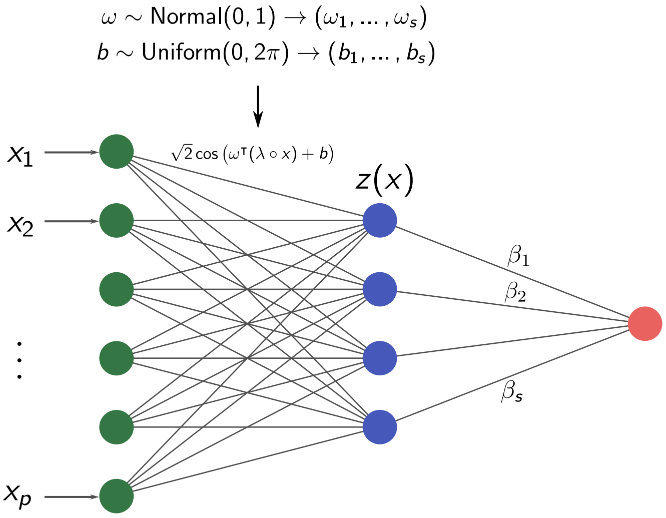

$\omega \sim \text{Normal}(0,1) \rightarrow (\omega_1, \dots, \omega_s)$
$b \sim \text{Uniform}(0, 2\pi) \rightarrow (b_1, \dots, b_s)$
$\sqrt{2}\cos(\omega^\intercal(\lambda \circ x) + b)$
$z(x)$
x_1
x_2
x_p
β_1
β_2
β_s

Figure 2 – RFFNet, with the gaussian ARD kernel, as a two-layer neural network. This interpretation is central to foresee extensions of RFFNet beyond kernel methods.

CHAPTER 4

EMPIRICAL EVALUATION

Recently, there has been a broad discussion on how to improve the empirical assessment of new machine learning (ML) methods (SCULLEY *et al.*, 2018; HENDERSON *et al.*, 2017). It is argued that the incredible practical success of machine learning has not been accompanied by the adoption of rigorous standards for the evaluation of ML methods. The implicit consensus in the field is that new methods should always defeat previous "competitors" in established benchmark datasets or tasks. Sculley *et al.* (2018), and references therein, gives many examples of situations where this culture may have led to delays, missed improvements, and false claims of outperformance.

Recent machine learning methods rely on complex models, non-convex optimization procedures, and training heuristics (early stopping and learning rate decay, for instance), which are not well understood theoretically. In this sense, the quest for successful methods should not be pursued at the expense of carefully gauging their strengths and limitations.

RFFNet, trying to improve the expressivity of kernel methods, is a complex model and solves a non-convex optimization problem during estimation. For this reason, we follow some recently suggested guidelines on empirical evaluation (SCULLEY *et al.*, 2018).

First, we describe the computational setting where RFFNet was evaluated and give some implementation details of the library. Next, we provide a description of the baseline algorithms which were compared to RFFNet and how hyperparameters were tuned during validation. Then, through a simple ablation study, we verify the impact of choosing different optimizers for minimizing RFFNet's objective function. Finally, we depict the optimization trajectories of RFFNet in the loss landscape by using UMAP (MCINNES; HEALY; MELVILLE, 2018), a recent dimension reduction technique.

4.1 Computing infrastructure

All studies described in this chapter were run with an Intel Core i7-9750H CPU with 2.60 GHz, 12 cores, and 15.6 GB of RAM. Experiments described in Chapter 5 were run with an Intel Core i7-8700 CPU with 3.20GHz, six cores, 12 threads, and 54 GB of RAM to evaluate all algorithms on the same hardware.

4.2 Implementation details

RFFNet is implemented as a TORCH model with a user interface adherent to the API of SCIKIT-LEARN. Adherence to the SCIKIT-LEARN API makes the method easily applicable and consistent with other tools available in the SCIKIT-LEARN environment, such as hyperparameter searches. The tools developed for this work, including the RFFNet model itself, are accessible in a Python library designated PYSELECT, publicly available at <https://github.com/mpotto/pyselect>. This library has few dependencies on consolidated libraries (NUMPY, TORCH and SCIKIT-LEARN). We stress that the library is still at an early stage of development, and changes in the interface or the repository can be made unexpectedly.

4.3 Baselines and tuning methodology

In this section, we describe the baseline algorithms that were compared to RFFNet and explain the tuning methodology adopted for each one. These algorithms were chosen either because they output a measure of feature importance, such as Sparse Random Fourier Features (SRFF), by Gregorová *et al.* (2018), or by their extensive use as predictive models for classification and regression problems, as is the case of XGBoost (CHEN; GUESTRIN, 2016).

As a general rule, we used data splitting into three samples: training, validation, and testing. The training sample was used to estimate the model, the validation sample was used to select all the algorithms' hyperparameters, and the test sample was used to evaluate the best-performing method, according to the validation sample. All possible combinations of the hyperparameters in the specified grids below were tested, except for XGBoost and GAMs, for which we used a random sampler (with fixed seed) to select values in predetermined grids. For regression problems, we used the mean squared error as the criterion for selecting the hyperparameters, whereas for classification problems, we used the AUC.

In what follows, we briefly describe the baselines and the adopted tuning methodologies:

1. Bayesian regression with Automatic Relevance Determination kernels (BARD): we used the implementation available in SCIKIT-LEARN (PEDREGOSA, 2016). We tuned the shape and rate parameters for the Gamma prior over the distribution of the precision of the linear model coefficients. The shape parameter λ_1 was varied in $\{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}\}$ and the rate parameter λ_2 in $\{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}\}$. We used the precisions of the distribution of weights λ as a measure of feature importance.

2. Sparse Random Fourier Features (SRFF): we used the original implementation of Gregorová *et al.* (2018). The method's internal tuning methodology for the regularization hyperparameter was used. The RFF scaling γ of SRFF was used as a measure of feature importance.

3. Kernel Ridge Regression (KRR): we used the implementation available in SCIKIT-LEARN (PEDREGOSA, 2016). We set the RBF kernel and tuned both the regularization hyperparameter α and the kernel's inverse length-scale γ. We varied α in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and γ in $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. KRR uses a single bandwidth for the kernel, therefore it does not output a measure of feature importance.

4. Logistic Regression: we used the implementation available in SCIKIT-LEARN (PEDREGOSA, 2016). An ℓ_1 regularization on the coefficients was used and we tuned the reciprocal of the regularization hyperparameter C, which was varied in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$. We did not consider any feature importance measures based on this model.

5. XGBoost: we used the PYTHON API of the original library implementation (CHEN; GUESTRIN, 2016). We tuned the maximum depth of trees (integer sampled between 3 and 20), the step size shrinkage (float sampled between 0 and 1), the minimum loss reduction required to partition further a leaf node (float sampled between 10^{-5} and 100) and the minimum child weight (integer sampled between 0 and 10). We performed 50 validation trials. We considered the number of times a feature is used to split the data across all trees as a measure of feature importance.

6. Generalized Additive Models (GAMs): we used the PYGAM library implementation (SERVÉN; BRUMMITT, 2018) of linear GAMs, trained exclusively in regression tasks. We used the default model of the library, which includes a univariate spline for each feature; thus, in this case, no interactions were modelled.

For RFFNet, in turn, we tuned the learning rate and the regularization hyperparameter. The learning rate was varied in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ and the regularization hyperparameter was taken from $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. For the number of random features, we considered $s = \lfloor n \log n \rfloor$. We adopted this as a rule of thumb for all experiments, following the discussion in section 2.4. For RFFNet, we considered the absolute value of the relevance parameter λ as a measure of feature importance.

Importantly, because the feature importances output by the aforementioned methods are in different numerical scales, we choose to scale them to the $[0, 1]$ interval.

4.4 Ablation study of optimizers

Ablation studies aim to verify the impact of specific model components on their overall performance. Initially introduced in biology to test how anatomy relates to the functioning of living beings, this technique has lately caught attention in the machine learning community (SCULLEY *et al.*, 2018; MOREAU *et al.*, 2022; BERTRAND *et al.*, 2021). In ML, an ablation study measures the impact of removing or replacing any model components – optimizers, function spaces, hyperparameter tuning heuristics *etc.*

For instance, in statistics, a researcher might be interested in testing whether adding an interaction term in a linear model can enhance the model's predictive performance compared to a simple baseline (a linear model with no interactions, for instance). Subsequently, the researcher may ask if the additional complexity introduced in the model by the interaction term, which can complicate estimation, balances well with the performance improvement.

In this section, we performed a similar experiment to test RFFNet optimization algorithm. In the previous chapter, we described the RFFNet optimization algorithm as a stochastic gradient descent (SGD) method based on the Adam optimizer (KINGMA; BA, 2014), which updates two blocks of coordinates alternately. These alternating updates with gradient descent (and proximal mappings associated with the regularization functions) share a resemblance with the PALM algorithm (BOLTE *et al.*, 2014). For this reason, we designated our algorithm as PALM Adam. Here we test whether PALM Adam performs better than a usual (or vanilla) stochastic gradient descent method with a single block of coordinates.

To perform this test, we compared four algorithms: SGD or Adam, with two blocks (PALM) or a single block of coordinates. We compared each algorithm on three simulated datasets, described in Table 2. All datasets were generated with a fixed random seed and have $n = 5 \times 10^4$ training points. To measure the impact of each optimizer, we plot the mean squared error (MSE) evaluated in the validation sample during model training. The result of this study is shown in Figure 3. First, notice that optimizing with alternating updates (PALM) and Adam finds low error solutions generally faster. Second, using solely SGD or combining SGD with PALM did not converge to low MSE solutions in any of the depicted cases. Both observations highlight the importance of combining Adam with

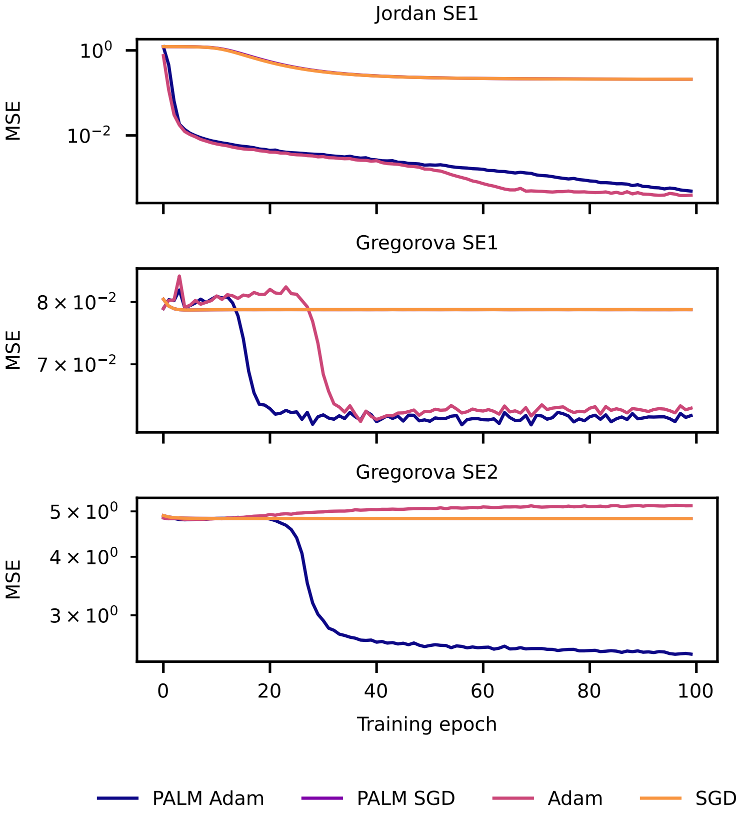


Figure 3 – Ablation study of optimizers. Combining alternating updates (PALM) with Adam performs better than other optimizers, converging faster and to low MSE error solutions.

alternating updates, as proposed in Chapter 3.

Table 2 – Description of simulated datasets used in the ablation study. In the Jordan SE1 dataset, features are sampled as $X \sim \mathsf{Normal}(0, \Sigma)$, with $\Sigma_{ij} = 0.6^{|i-j|}$. In the Gregorova SE1 and Gregorova SE2 datasets, features are sampled as $X \sim \mathsf{Normal}(0, I_p)$. In all cases, $\varepsilon \sim \mathsf{Normal}(0, \sigma = 0.1)$.

Dataset	Features (p)	Response
Jordan SE1*	10	$y = x_1 + \varepsilon$
Gregorova SE1**	18	$y = \sin[(x_1 + x_3)^2] \sin(x_7 \, x_8 \, x_9) + \varepsilon$
Gregorova SE2**	100	$y = \log[(x_{11} + x_{12} + x_{13} + x_{14} + x_{15})^2] + \varepsilon$

* Jordan, Liu and Ruan (2021).

** Gregorová *et al.* (2018).

4.5 Visualizing optimization trajectories

Training RFFNet requires minimizing a (highly) non-convex objective function, which is a challenging task computationally. In practice, however, as shown in the section 4.4, RFFNet's optimization algorithm can lead to good solutions. In this section, guided by recent work (CSILLAG *et al.*, 2022), we explore the loss landscape and optimization trajectories of RFFNet trained in a simulated dataset. We exhibit a bad and a good solution to the objective function minimization and display the feature importances learned in each case.

First, we fixed the simulated dataset as Gregorova SE1, described in Table 2, which has $p = 18$ features. We also set the number of random features (s) as $s = \lfloor \sqrt{n} \log n \rfloor$, where n is the training sample size, guided by the discussion in section 2.4. Then, we trained multiple RFFNet models with varying hyperparameters (approximately 700 different combinations). For each epoch during training, we represented the RFFNet model as a vector in $\mathbb{R}^{p+s}$, where the first p coordinates are the feature importances and the last s coordinates are the weights of the RFF expansion. Each of these vectors is a portrait of the model during training; if we observe how these vectors evolve during training, we get an optimization trajectory. Concatenating all these vectors, we then used UMAP (MCINNES; HEALY; MELVILLE, 2018) to reduce the dimensionality of each model from $\mathbb{R}^{p+s}$ to $\mathbb{R}^2$. Importantly, as done in Csillag *et al.* (2022), we employed the densMAP (NARAYAN; BERGER; CHO, 2020) option of UMAP to preserve local density information, which guarantees that similar models and trajectories are plotted close to each other in $\mathbb{R}^2$.

To generate a complete two-dimensional loss landscape, we trained the models with combinations of hyperparameters that lead to good performance or intentionally incur non-convergence and bad solutions. By doing this, we can observe how good solutions to RFFNet navigate in the loss landscape and if RFFNet can get stuck in stationary points with high error.

First, we show in Figure 4 the optimization trajectory and the feature importance pattern during a successful model training that concluded with low error. In general, note that the loss landscape for the training sample has a smaller mean squared error (visually, the surface has a darker blue tone). Also, observe that the loss landscape for training RFFNet in this dataset has many regions of small error surrounded by higher error, which can trap the optimization procedure in non-optimal stationary points. Specifically, in Figure 4, we notice that the successful training resulted in a feature importance pattern that matches the features 1, 3, 7, 8, and 9 that are active in the regression function of the dataset.

Lastly, we show in Figure 5 the optimization trajectory and the feature importance pattern for a trained model that concluded with high error. In this case, we see that the relevance pattern does not correspond to the covariates that appear in the regression function of the Gregorova SE1 dataset.

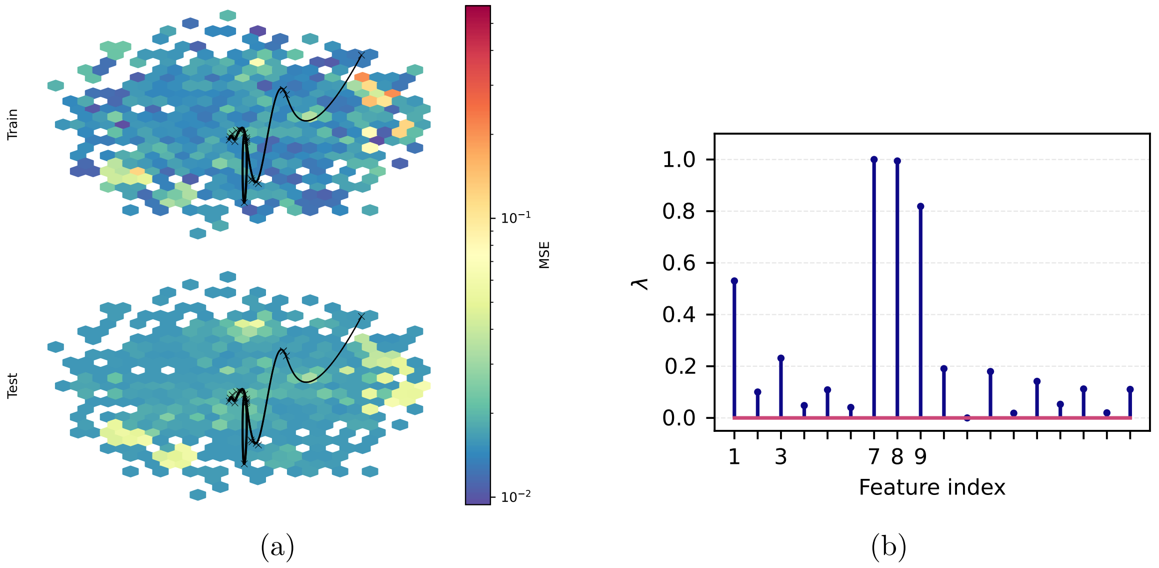


Figure 4 – (a) Trajectories of a well-trained RFFNet model in the loss landscapes of the training (top) and test (down) sample. Trajectories were ordered according to the x coordinate and do not reflect the temporal evolution of the parameters in the landscape. (b) Feature relevances scaled to $[0, 1]$ for the final model in Figure 4a. Observe that the greatest feature relevances match the features that appear in the regression function of Gregorova SE1 (see Table 2).

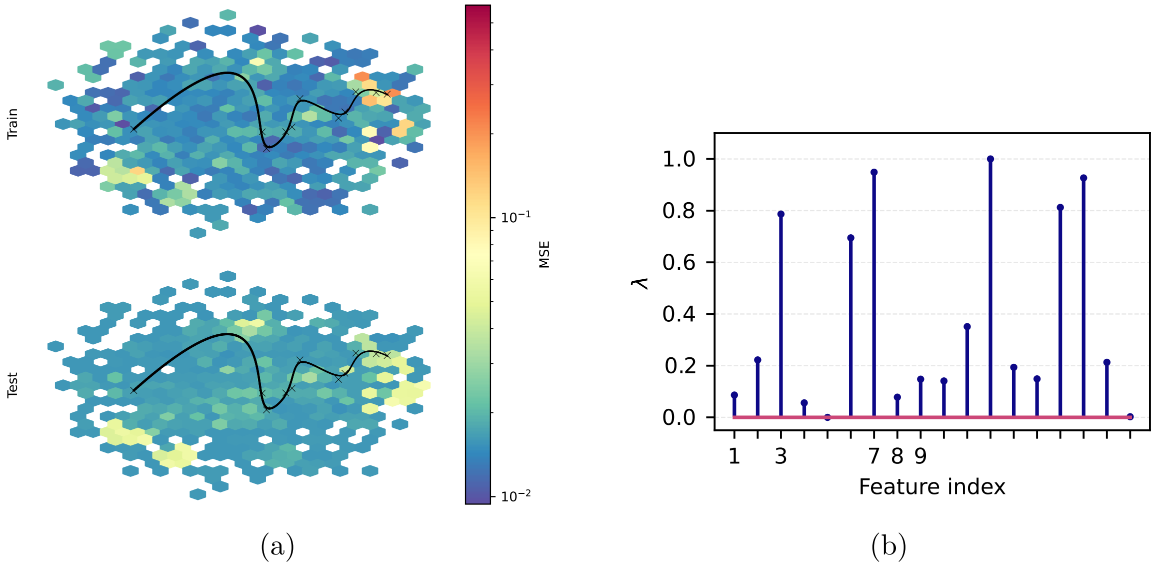


Figure 5 – (a) Trajectories of an ill-trained RFFNet model in the loss landscapes of the training (top) and test (down) sample. Trajectories were ordered according to the x coordinate and do not reflect the temporal evolution of the parameters in the landscape. (b) Feature relevances scaled to $[0, 1]$ for the final model in Figure 4a. In this case, feature relevances do not match the features that appear in the regression function of Gregorova SE1 (see Table 2).

CHAPTER 5

EXPERIMENTS

In this chapter, we describe the numerical experiments performed on simulated and real datasets to validate RFFNet. We used the ARD Gaussian kernel in all experiments. A description of the data processing steps adopted for each dataset is available on Appendix B. For each experiment, we discuss the results and, whenever possible, interpret them with domain knowledge.

Table 3 – Summary of the datasets used in simulations and real data experiments.

Dataset	Train. size	Val. size	Test size	Number of features (p)
Gregorova SE1	50 000	2 000	2 000	18
Gregorova SE2	50 000	2 000	2 000	100
Ailerons	11 000	1 000	1 000	40
Comp-Act	6 000	1 000	1 000	12
Amazon	218 502	72 835	72 834	5000
Higgs	6 600 000	2 200 000	2 200 000	28

5.1 Simulations

We use simulated datasets proposed in Gregorová *et al.* (2018) to test whether RFFNet can correctly identify the relevant features in the regression function with good predictive performance.

5.1.1 Greogorova SE1

This experiment is a regression problem with $p = 18$ features, of which only 5 are active in the regression function. The response is calculated as $y = \sin[(x_1 + x_3)^2]\sin(x_7\, x_8\, x_9) + \varepsilon$ with $\varepsilon \sim \mathsf{Normal}(0, \sigma = 0.1)$. We performed two experiments. First, to verify the influence of the sample size on the recovery of relevant features, we trained our model with $10^3, 5 \times 10^3, 10^4$ and 5×10^4 instances. Figure 6 shows that increasing the sample size enhances the identification of relevant features. Next, to compare with other algorithms, we fix the training size as $n = 5 \times 10^4$ instances and evaluate the performance on a held-out test sample. As shown in Table 4, RFFNet outperformed all baseline algorithms and exhibited low memory usage. Compared to the full Kernel Ridge Regression (KRR) with fixed bandwidths, it trained faster, with less memory, and to a lower MSE solution. Additionally, Figure 7 shows that RFFNet is the only method to output feature relevances that matches the active features of the Gregorova SE1 dataset.

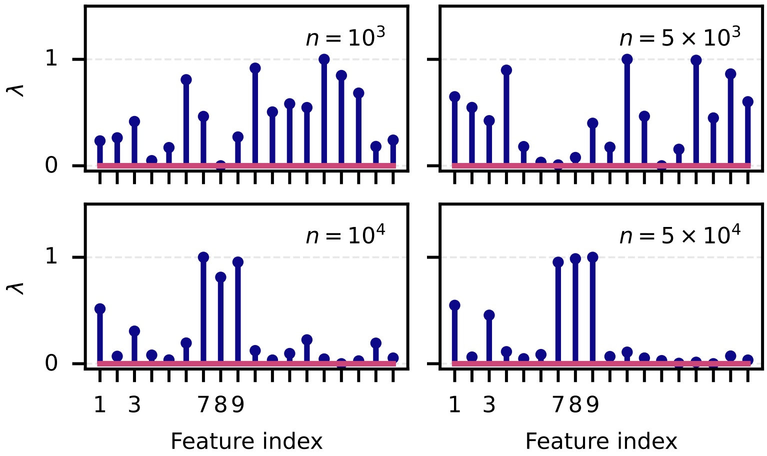


Figure 6 – Scaled relevances for the Gregorova SE1 simulation. Increasing the sample size enhances the identification of relevant features when training RFFNet. With sample sizes greater than 10^4, RFFNet consistently gives small weights to the irrelevant features.

5.1.2 Gregorova SE2

This experiment is a regression problem with $p = 100$ features of which only 5 are active in the regression function. The response is calculated as $y = \log[(x_{11}+x_{12}+x_{13}+x_{14}+x_{15})^2]+\varepsilon$ with $\varepsilon \sim \mathsf{Normal}(0, \sigma = 0.1)$. Again, we performed two experiments, one to verify the effect of sample sizes in identifying relevant features and the other, with fixed sample size, to compare with baseline algorithms. Table 4 shows that, in the fixed sample size scenario, RFFNet outperformed all baseline algorithms with low memory usage. Similarly to the previous experiment, Figure 8 depicts that increasing the sample size strengthens RFFNet's ability to identify the relevant features. Finally, Figure 9 shows that RFFNet is the only method to output feature relevances that match the active features of the Gregorova SE2 dataset.

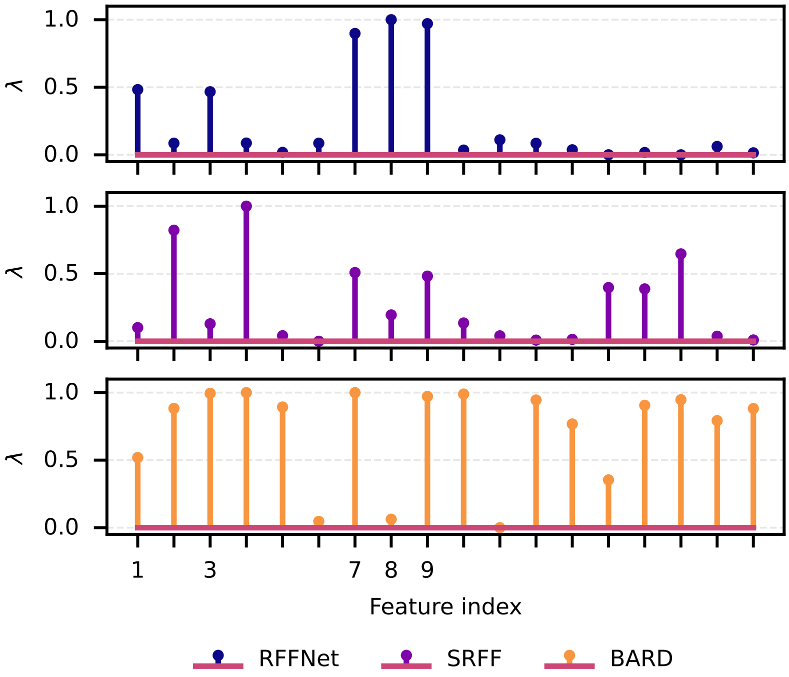


Figure 7 – Comparisom of scaled relevances for the Gregorova SE1 simulation. RFFNet is the only method that outputs a measure of the feature relevances that matches the active features in the Gregorova SE1 dataset.

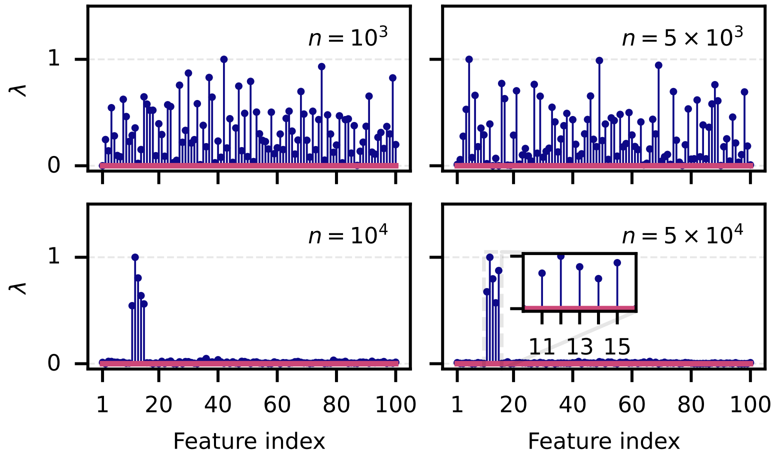


Figure 8 – Scaled relevances for the Gregorova SE2 simulation. Increasing the sample size enhances the identification of relevant features when training RFFNet. Again, with sample sizes greater than 10^4, RFFNet consistently gives small weights to the irrelevant features.

5.2 Real datasets

5.2.1 Amazon Fine Food Reviews

This dataset consists of 568,454 reviews of fine foods sold by Amazon, spanning from October 1999 to October 2012 (MCAULEY; LESKOVEC, 2013). This is a classi-

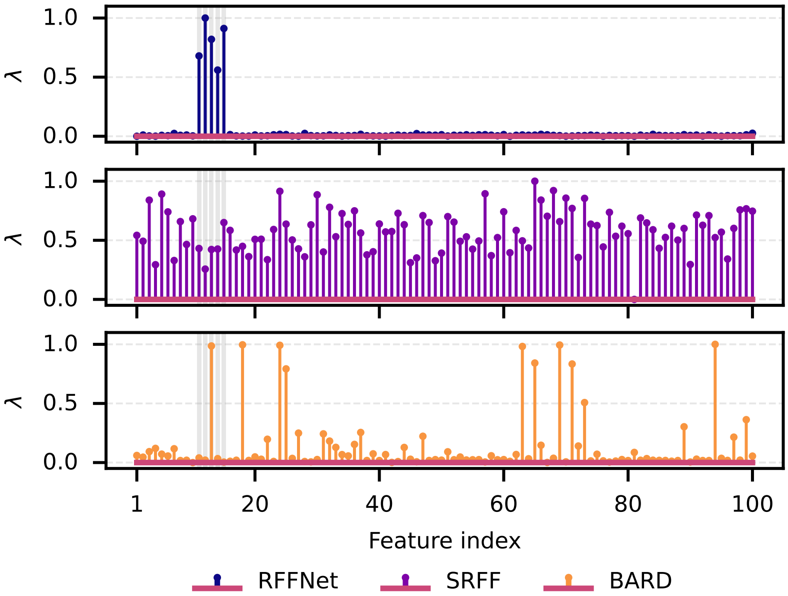


Figure 9 – Comparisom of scaled relevances for the Gregorova SE2 simulation. Again, RFFNet is the only method that outputs a measure of the feature relevances that match the active features in the Gregorova SE2 dataset.

Table 4 – Mean squared error in the test sample, peak RAM use, and elapsed time for training RFFNet and baseline algorithms in the Gregorova SE1 and SE2 simulation datasets. RFFNet outperformed all baselines in the simulation. Best performances are displayed in boldface.

	SE1			SE2		
Metric	MSE	RAM	Time	MSE	RAM	Time
KRR	0.082(0.008)	38.6 GB	345 s	3.7(0.5)	38.9 GB	337 s
BARD	0.085 (0.008)	**0.3** GB	**0.04** s	5.3 (0.5)	**0.5** GB	**0.4** s
GAM	0.086(0.009)	1.7 GB	4.3 s	4.9(0.5)	9.9 GB	50.9 s
SRFF	0.083 (0.008)	0.9 GB	20.2 s	5.4 (0.5)	1.0 GB	91 s
RFFNet	**0.070** (0.008)	0.6 GB	63 s	**1.3** (0.2)	0.6 GB	71 s

fication task, and the objective is to predict whether a review is positive (has a score between 4 and 5) or negative (has a score between 1 and 3). For this reason, we used the cross-entropy as the loss function. Table 5 shows that RFFNet performed worse than baselines, although it used significantly less memory. Additionally, Figure 10 shows that the ten features (stemmed words) with greater relevances (according to RFFNet) are indeed associated with the quality of the product (e.g., “disappoint”, “terribl”), while those with smaller relevances are not. The inferior performance may be attributed to the difficult optimization problem solved by RFFNet.

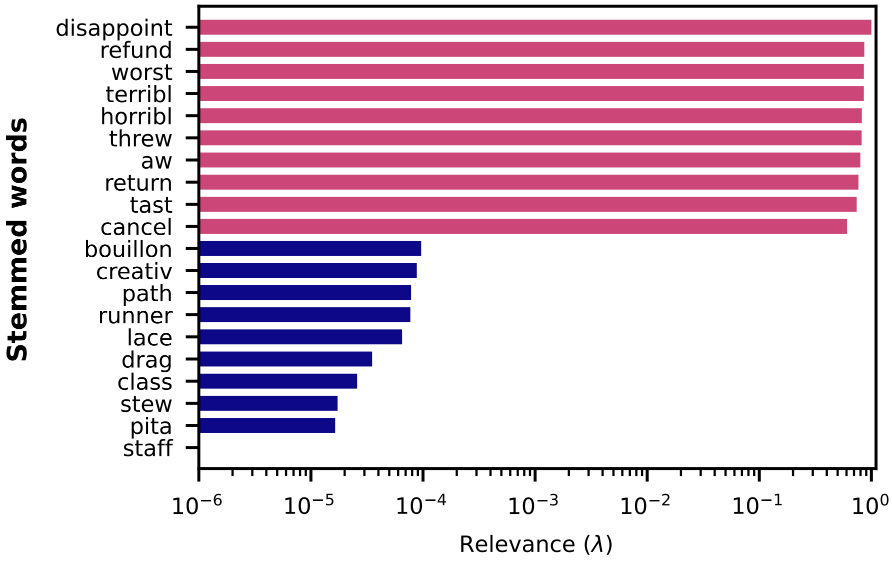


Figure 10 – Scaled relevances for the ten features with greatest (red) and lowest (blue) relevances for the Amazon Fine Food Reviews dataset. The stemmed words with the ten greatest relevances are indeed associated with product quality, showing that the feature relevance given by RFFNet is meaningful.

5.2.2 Higgs

This dataset is part of a classification problem where the objective is to distinguish between a signal process associated with the creation of Higgs bosons and a background process that generates similar decay products but with distinctive features. In this experiment, we used cross-entropy as the loss function. Table 5 shows that RFFNet performs worse than baselines in all aspects. In 11, we compare the feature importances given by RFFNet with the feature importances reported by XGBoost, scaling both to the interval $[0, 1]$. Observe that both algorithms seems to provide similar relevances to the available features, but RFFNet does not attain similar performance in the classification metrics.

Table 5 – Classification metrics in the test sample, peak RAM use, and elapsed time for training RFFNet and baseline algorithms in the real-world classification datasets. RFFNet exhibits on-par performance with the baselines but has the advantage of providing feature importances. Best performances are displayed in boldface.

	Amazon					Higgs				
Metrics	Acc.	F1	AUC	RAM	Time	Acc.	F1	AUC	RAM	Time
Logistic (ℓ_1)	**0.92**	**0.95**	**0.95**	47.8 GB	**281** s	0.64	0.69	0.68	9.6 GB	**68.4** s
XGBoost	0.92	0.95	0.94	31.4 GB	1265 s	**0.75**	**0.77**	**0.84**	**6.7** GB	1988 s
RFFNet	0.86	0.92	0.89	**24.0** GB	578 s	0.71	0.73	0.79	18.5 GB	11214 s

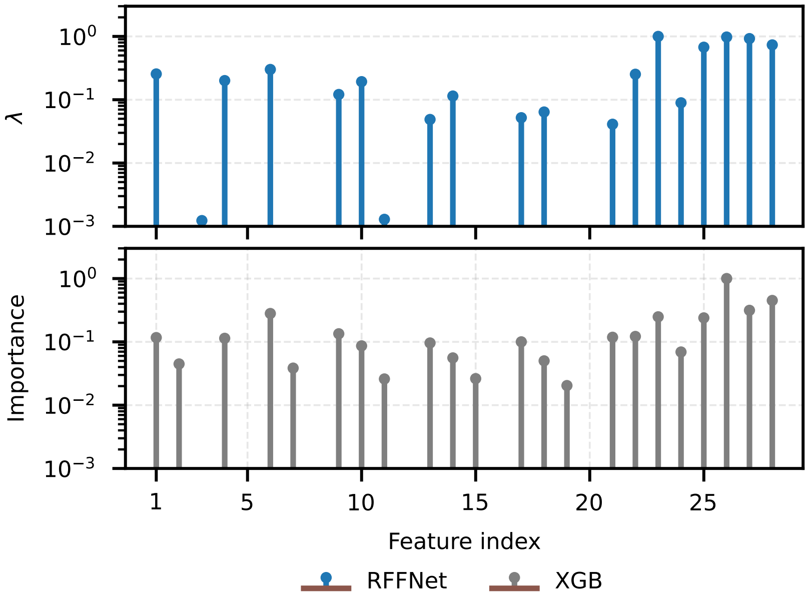


Figure 11 – Scaled relevances for RFFNet and XGBoost feature importances on the Higgs dataset. Both methods give similar relevances to the available features.

5.2.3 Ailerons

This dataset consists of a collection of sensor measurements describing the status of a jet aircraft. The goal is to predict the control action on the ailerons of the aircraft. The mean squared error reported in 6 shows that RFFNet performance is inferior to the KRR and BARD baselines. We did not plot the relevance pattern for this dataset because the inferior performance, and possible non-convergence of the optimization procedure, may induce misleading inferences about the relevant features.

5.2.4 Comp-Act

This dataset consists of a collection of computer systems activity measures. The problem is a regression task, and the objective is to predict the portion of time the CPUs run in user mode instead of system mode (which gives privileged access to hardware). Table 6 shows that RFFNet performed better than the BARD and SRFF baselines. The SRFF algorithm did not converge and produced a model with enormous MSE in the held-out dataset. Similarly to the previous dataset, because RFFNet had an inferior performance, we choose not to plot the relevance pattern.

5.3 Additional experiments

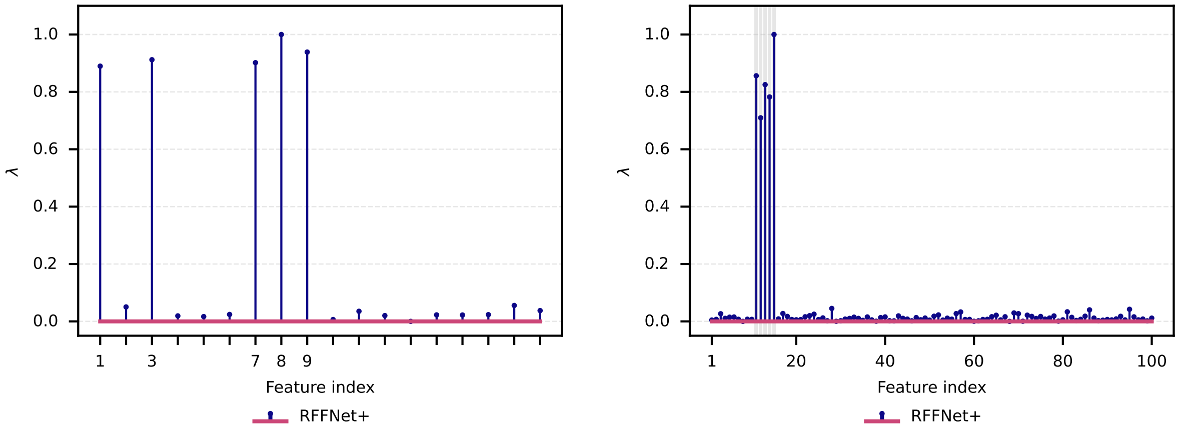
In this section, following the ideas presented in Chapter 3, we tested if using RFFNet as a first layer for neural network architectures can improve the predictive per-

Table 6 – Mean squared error in the test sample, peak RAM use, and elapsed time for training RFFNet and baseline algorithms in the real-world regression datasets. Best performances are displayed in boldface.

	Ailerons			Comp-Act		
Metric	MSE	RAM	Time	MSE	RAM	Time
KRR	$\mathbf{2.4}\,(\mathbf{0.3}) \times \mathbf{10^{-8}}$	2.3 GB	5.7 s	11(4)	0.9 GB	1.5 s
BARD	$2.7\,(0.3) \times 10^{-8}$	**0.3** GB	**0.03** s	120 (50)	**0.3** GB	**0.02** s
SRFF	$7(0.1) \times 10^{-7}$	1.1 GB	1.2 s	12.9 (1)	0.5 GB	7.2 s
GAM	$2.7\,(0.4) \times 10^{-8}$	1.5 GB	3.0 s	**10.2**(2)	0.5 GB	0.4 s
RFFNet	$3.1\,(0.3) \times 10^{-8}$	0.4 GB	7.7 s	21(3)	**0.3** GB	4.7 s

formance of our approach and deliver sensible feature importances. For the simulation datasets, Gregorova SE1 and Gregorova SE2, we considered a four-layer fully-connected neural network with 300, 20, 10, and 1 output units. As for the Comp-Act dataset, we chose a fully-connected architecture with 500, 100, 50, 10, and 1 output units. In both cases, the first layer is an RFFNet layer, the subsequent are dense layers with ReLU activation functions, and the last layer has no activation. The neural network was trained with no regularization other than early stopping with patience set to 10. We designated this method as RFFNet+.

First, we trained the neural net in the simulation datasets Gregorova SE1 and Gregorova SE2. Figure 12 shows that RFFNet+ promptly identifies the relevant features for both datasets. Moreover, Table 7 demonstrate that the predictive performance of the method was greatly increased with RFFNet+, attaining MSE errors significantly smaller than base RFFNet (see Table 4), with no impact on RAM and running time.



(a) (b)

Figure 12 – RFFNet+ scaled relevances for the simulation datasets Gregorova SE1 (a) and Gregorova SE2 (b). In both cases, the relevance pattern matches the active features of each dataset.

Finally, we also trained RFFNet+ on the Comp-Act dataset. Table 8 shows that

Table 7 – Mean squared error in the test sample, peak RAM use, and elapsed time for training RFFNet+ in the simulation datasets.

	SE1			SE2		
Metric	MSE	RAM	Time	MSE	RAM	Time
RFFNet+	0.057 (0.005)	0.4 GB	78.5 s	0.17 (0.02)	0.6 GB	50.2 s

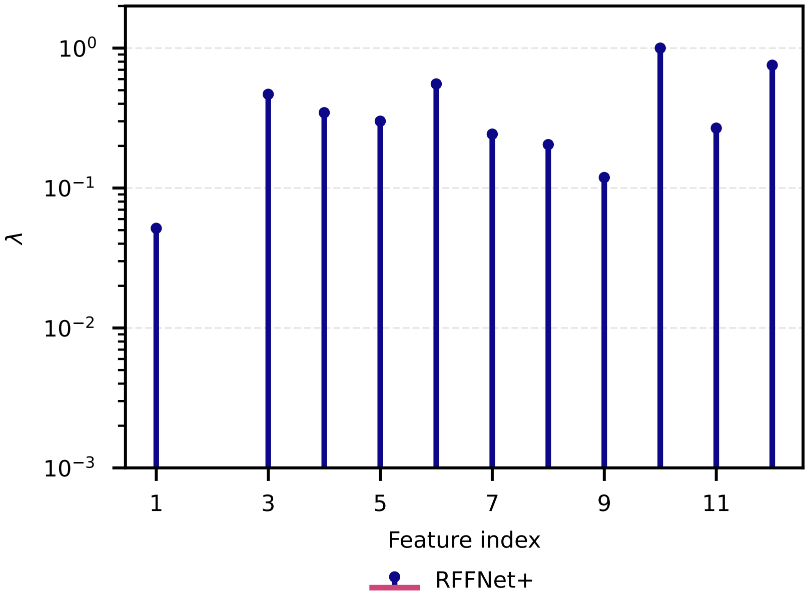


Figure 13 – RFFNet+ scaled relevances for the Comp-Act dataset. Observe that almost all features were considered relevant to predict the target response.

RFFNet+ did perform better than any other baselines in Table 6, with significant less memory use than kernel ridge regression. In Figure 13, we show the relevance pattern output by RFFNet+ in this case. The reduction in the MSE error provided by RFFNet+ may indicate that the relevance pattern found by RFFNet+ is more reliable than that of RFFNet. Figure 13 shows that almost all features of the problem are treated as important in the neural network. This, in turn, may indicate why simple RFFNet did not perform well: the absence of sparseness in data could have caused difficulties in fitting the relevances parameter.

Table 8 – Mean squared error in the test sample, peak RAM use, and elapsed time for training RFFNet+ in the Comp-Act dataset.

	Comp-Act		
Metric	MSE	RAM	Time
RFFNet+	8.6 (0.9)	0.36 GB	33 s

CHAPTER 6

CONCLUSIONS

In this work, we proposed and validated a new scalable and interpretable kernel method, designated as RFFNet, for supervised learning problems. The method is based on the framework of random Fourier features (RAHIMI; RECHT, 2008a) applied to Automatic Relevance Determination (ARD) kernels (RASMUSSEN; WILLIAMS, 2006). These kernels can be used to assemble kernel methods that are interpretable since the relevances can be used to mitigate the impact of features that do not associate with the response. Besides, the use of random Fourier features diminishes the computational cost of kernel methods (especially their large memory requirements) by reducing the number of parameters to be estimated, thus making our approach scalable.

We validated RFFNet in a series of numerical experiments. RFFNet exhibited good performance in simulation results but shows slightly inferior performance on real datasets when compared to well-established predictive inference algorithms. We also executed experiments with a new version of RFFNet, designated as RFFNet+, tailored to filter irrelevant features in arbitrary neural-network architectures. RFFNet+ displayed significantly better performance when compared to RFFNet and all baseline algorithms.

We stress that there remain many interesting problems for future work. In the theory realm, providing generalization bounds for RFFNet, testing the consistency of post-processing the relevances as a feature selection procedure, and studying connection with data-adaptive kernels in neural networks (DOU; LIANG, 2019) are some interesting research directions. From the practical point of view, we would like to extend the use of RFFNet to general neural network architectures, unsupervised problems, and non-tabular data.

BIBLIOGRAPHY

ALLEN, G. I. Automatic feature selection via weighted kernels and regularization. **https://doi.org/10.1080/10618600.2012.681213**, Taylor Francis Group, v. 22, p. 284–299, 2013. ISSN 10618600. Available: <https://www.tandfonline.com/doi/abs/10.1080/10618600.2012.681213>. Citation on page 21.

ARONSZAJN, N. Theory of reproducing kernels. **Transactions of the American Mathematical Society**, v. 68, p. 337–404, 1950. ISSN 0002-9947. Citation on page 35.

BACH, F. On the equivalence between kernel quadrature rules and random feature expansions. 2 2015. Available: <http://arxiv.org/abs/1502.06800>. Citation on page 41.

BALOG, M.; TOLSTIKHIN, I.; SCHöLKOPF, B. Differentially private database release via kernel mean embeddings. In: . PMLR, 2018. p. 414–422. ISSN 2640-3498. Available: <https://proceedings.mlr.press/v80/balog18a.html>. Citation on page 19.

BAUSCHKE, H. H.; COMBETTES, P. L. **Convex Analysis and Monotone Operator Theory in Hilbert Spaces**. [S.l.]: Springer International Publishing, 2017. ISBN 978-3-319-48310-8. Citations on pages 26 and 27.

BERTIN, K.; LECUÉ, G. Selection of variables and dimension reduction in high-dimensional non-parametric regression. **Electronic Journal of Statistics**, Institute of Mathematical Statistics and Bernoulli Society, v. 2, n. none, p. 1224 – 1241, 2008. Available: <https://doi.org/10.1214/08-EJS327>. Citation on page 19.

BERTRAND, Q.; KLOPFENSTEIN, Q.; BANNIER, P.-A.; GIDEL, G.; MASSIAS, M. Beyond l1: Faster and better sparse models with skglm. 4 2022. Available: <https://arxiv.org/abs/2204.07826>. Citation on page 27.

BERTRAND, Q.; KLOPFENSTEIN, Q.; MASSIAS, M.; BLONDEL, M.; VAITER, S.; GRAMFORT, A.; SALMON, J. Implicit differentiation for fast hyperparameter selection in non-smooth convex learning. 5 2021. Citations on pages 24 and 54.

BOLTE, J.; SABACH, S.; TEBOULLE, M.; BOLTE, J.; SABACH, S.; TEBOULLE, . M. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. **Math. Program., Ser. A**, v. 146, p. 459–494, 2014. Citations on pages 47 and 54.

BOYD, S.; VANDERBERGHE, L. **Convex optimization**. [S.l.]: Cambridge University Press, 2004. ISBN 0521833787. Citations on pages 25 and 33.

BROUARD, C.; MARIETTE, J.; FLAMARY, R.; VIALANEIX, N. Feature selection for kernel methods in systems biology. **NAR Genomics and Bioinformatics**, v. 4, n. 1, 03 2022. ISSN 2631-9268. Available: <https://doi.org/10.1093/nargab/lqac014>. Citation on page 45.

CAPONNETTO, A.; VITO, E. D. Optimal rates for the regularized least-squares algorithm. **Foundations of Computational Mathematics**, Springer, v. 7, p. 331–368, 7 2007. ISSN 16153375. Available: <https://link.springer.com/article/10.1007/s10208-006-0196-8>. Citation on page 42.

CARRATINO, L.; RUDI, A.; ROSASCO, L. Learning with sgd and random features. **Advances in Neural Information Processing Systems**, v. 31, 2018. Citation on page 27.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, ACM, 2016. Available: <http://dx.doi.org/10.1145/2939672.2939785>. Citations on pages 52 and 53.

CHEN, Y.; CHEWI, S.; SALIM, A.; WIBISONO, A. Improved analysis for a proximal algorithm for sampling. 2 2022. Available: <http://arxiv.org/abs/2202.06386>. Citation on page 25.

CHEWI, S.; GERBER, P.; LEE, H.; LU, C. Fisher information lower bounds for sampling. 10 2022. Available: <https://arxiv.org/abs/2210.02482>. Citation on page 25.

CORTES, C.; VAPNIK, V.; SAITTA, L. Support-vector networks. **Machine Learning 1995 20:3**, Springer, v. 20, p. 273–297, 9 1995. ISSN 1573-0565. Available: <https://link.springer.com/article/10.1007/BF00994018>. Citation on page 19.

CSILLAG, D.; PIAZZA, C.; RAMOS, T.; ROMANO, J. V.; OLIVEIRA, R. I.; ORENSTEIN, P. Exactboost: Directly boosting the margin in combinatorial and non-decomposable metrics. In: . PMLR, 2022. p. 9017–9049. ISSN 2640-3498. Available: <https://proceedings.mlr.press/v151/csillag22a.html>. Citation on page 56.

CURTó, J. D.; ZARZA, I. C.; YANG, F.; SMOLA, A.; TORRE, F. de la; NGO, C. W.; GOOL, L. van. Mckernel: A library for approximate kernel expansions in log-linear time. 2 2017. Available: <http://arxiv.org/abs/1702.08159>. Citations on pages 19 and 39.

DANCE, H.; PAIGE, B. Fast and scalable spike and slab variable selection in high-dimensional gaussian processes. 11 2021. Available: <http://arxiv.org/abs/2111.04558>. Citation on page 43.

DOU, X.; LIANG, T. Training neural networks as learning data-adaptive kernels: Provable representation and approximation benefits. **Journal of the American Statistical Association**, 2019. ISSN 1537274X. Available: <http://arxiv.org/abs/1901.07114http://dx.doi.org/10.1080/01621459.2020.1745812>. Citation on page 67.

GEER, S. A. van de S. A. **Applications of empirical process theory**. [S.l.]: Cambridge University Press, 2000. 286 p. ISBN 9780521650021. Citation on page 25.

GREGOROVÁ, M.; RAMAPURAM, J.; KALOUSIS, A.; MARCHAND-MAILLET, S. **Large-scale nonlinear variable selection via kernel random features**. 2018. Available: <http://arxiv.org/abs/1804.07169>. Citations on pages 21, 45, 52, 53, 55, and 59.

GUYON, I.; WESTON, J.; BARNHILL, S.; VAPNIK, V. Gene selection for cancer classification using support vector machines. **Machine Learning**, Springer, v. 46, p. 389–422, 2002. ISSN 08856125. Available: <https://link.springer.com/article/10.1023/A:1012487302797>. Citation on page 21.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning: Data mining, inference, and prediction**. [s.n.], 2009. ISSN 03436993. ISBN 9780387848570. Available: <http://www.springerlink.com/index/D7X7KX6772HQ2135.pdf>. Citation on page 31.

HE, X.; WANG, J.; LV, S. Efficient kernel-based variable selection with sparsistency. 2 2018. Citation on page 19.

HENDERSON, P.; ISLAM, R.; BACHMAN, P.; PINEAU, J.; PRECUP, D.; MEGER, D. Deep reinforcement learning that matters. 9 2017. Available: <https://arxiv.org/abs/1709.06560>. Citation on page 51.

HOFMANN, T.; SCHÖLKOPF, B.; SMOLA, A. J. Kernel methods in machine learning. **The Annals of Statistics**, v. 36, 6 2008. ISSN 0090-5364. Citation on page 31.

IZMAILOV, A.; SOLODOV, M. **Otimização - volume 2**. IMPA, 2018. 479 p. ISBN 978-85-244-0454-2. Available: <https://impa.br/page-livros/otimizacao-volume-2/>. Citation on page 28.

JITKRITTUM, W.; KANAGAWA, H.; SCHöLKOPF, B. Testing goodness of fit of conditional density models with kernels. In: . PMLR, 2020. p. 221–230. ISSN 2640-3498. Available: <https://proceedings.mlr.press/v124/jitkrittum20a.html>. Citation on page 19.

JORDAN, M. I.; LIU, K.; RUAN, F. On the self-penalization phenomenon in feature selection. p. 1–54, 2021. Available: <http://arxiv.org/abs/2110.05852>. Citations on pages 19, 21, 45, and 55.

KEERTHI, S.; SINDHWANI, V.; CHAPELLE, O. An efficient method for gradient-based adaptation of hyperparameters in svm models. In: SCHöLKOPF, B.; PLATT, J.; HOFFMAN, T. (Ed.). **Advances in Neural Information Processing Systems**. MIT Press, 2006. v. 19. Available: <https://proceedings.neurips.cc/paper/2006/file/cc431fd7ec4437de061c2577a4603995-Paper.pdf>. Citations on pages 21 and 43.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. 12 2014. Available: <http://arxiv.org/abs/1412.6980>. Citations on pages 27, 30, 48, and 54.

LAFFERTY, J.; WASSERMAN, L. Rodeo: Sparse, greedy nonparametric regression. **https://doi.org/10.1214/009053607000000811**, Institute of Mathematical Statistics, v. 36, p. 28–63, 2 2008. ISSN 0090-5364. Available: <https://projecteuclid.org/journals/annals-of-statistics/volume-36/issue-1/Rodeo-Sparse-greedy-nonparametric-regression/10.1214/009053607000000811.fullhttps://projecteuclid.org/journals/annals-of-statistics/volume-36/issue-1/Rodeo-Sparse-greedy-nonparametric-regression/10.1214/009053607000000811.short>. Citation on page 19.

LE, Q. V.; SARLOS, T.; SMOLA, A. J. Fastfood: Approximate kernel expansions in loglinear time. 8 2014. Available: <http://arxiv.org/abs/1408.3060>. Citations on pages 19 and 39.

LI, Z.; TON, J.-F.; OGLIC, D.; SEJDINOVIC, D. Towards a unified analysis of random fourier features. **Journal of Machine Learning Research**, v. 22, n. 108, p. 1–51, 2021. Available: <http://jmlr.org/papers/v22/20-1369.html>. Citations on pages 41 and 46.

LIU, F.; XU, W.; LU, J.; SUTHERLAND, D. J. Meta two-sample testing: Learning kernels for testing with limited data. 6 2021. Available: <http://arxiv.org/abs/2106.07636>. Citation on page 19.

LOUW, N.; STEEL, S. J. Variable selection in kernel fisher discriminant analysis by means of recursive feature elimination. **Computational Statistics Data Analysis**, v. 51, p. 2043–2055, 2006. ISSN 0167-9473. Available: <https://www.sciencedirect.com/science/article/pii/S0167947305003294>. Citation on page 21.

MCAULEY, J. J.; LESKOVEC, J. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In: **Proceedings of the 22nd International Conference on World Wide Web**. New York, NY, USA: Association for Computing Machinery, 2013. (WWW '13), p. 897–908. ISBN 9781450320351. Available: <https://doi.org/10.1145/2488388.2488466>. Citation on page 61.

MCINNES, L.; HEALY, J.; MELVILLE, J. Umap: Uniform manifold approximation and projection for dimension reduction. 2 2018. Available: <http://arxiv.org/abs/1802.03426>. Citations on pages 51 and 56.

MERCER, J. Xvi. functions of positive and negative type, and their connection the theory of integral equations. **Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character**, The Royal Society London, v. 209, p. 415–446, 1 1909. ISSN 0264-3952. Available: <https://royalsocietypublishing.org/doi/10.1098/rsta.1909.0016>. Citation on page 35.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of Machine Learning, second edition**. [S.l.]: MIT Press, 2018. (Adaptive Computation and Machine Learning series). ISBN 9780262039406. Citations on pages 23, 24, and 33.

MOREAU, T.; MASSIAS, M.; GRAMFORT, A.; ABLIN, P.; BANNIER, P.-A.; CHARLIER, B.; DAGRéOU, M.; TOUR, T. D. la; DURIF, G.; DANTAS, C. F.; KLOPFENSTEIN, Q.; LARSSON, J.; LAI, E.; LEFORT, T.; MALéZIEUX, B.; MOUFAD, B.; NGUYEN, B. T.; RAKOTOMAMONJY, A.; RAMZI, Z.; SALMON, J.; VAITER, S. Benchopt: Reproducible, efficient and collaborative optimization benchmarks. 6 2022. Available: <http://arxiv.org/abs/2206.13424>. Citation on page 54.

NARAYAN, A.; BERGER, B.; CHO, H. Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability. **bioRxiv**, Cold Spring Harbor Laboratory, p. 2020.05.12.077776, 5 2020. Available: <https://www.biorxiv.org/content/10.1101/2020.05.12.077776v1https://www.biorxiv.org/content/10.1101/2020.05.12.077776v1.abstract>. Citation on page 56.

NITANDA, A. Stochastic proximal gradient descent with acceleration techniques. **Advances in Neural Information Processing Systems**, v. 27, 2014. Citation on page 31.

PARIKH, N.; BOYD, S. **Proximal algorithms**. [S.l.: s.n.], 2013. 130 p. ISBN 1601987161. Citation on page 29.

PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L.; DESMAISON, A.; KöPF, A.; YANG, E.; DEVITO, Z.; RAISON, M.; TEJANI, A.; CHILAMKURTHY, S.; STEINER, B.; FANG, L.; BAI, J.; CHINTALA, S. Pytorch: An imperative style, high-performance deep learning library. 12 2019. Available: <https://arxiv.org/abs/1912.01703>. Citation on page 20.

PEDREGOSA, F. Hyperparameter optimization with approximate gradient. 2 2016. Available: <https://arxiv.org/abs/1602.02355>. Citations on pages 24 and 53.

POCK, T.; SABACH, S. Inertial proximal alternating linearized minimization (iPALM) for nonconvex and nonsmooth problems. **SIAM Journal on Imaging Sciences**, Society for Industrial & Applied Mathematics (SIAM), v. 9, n. 4, p. 1756–1787, jan 2016. Citation on page 47.

POWELL, M. J. On search directions for minimization algorithms. **Mathematical Programming 1973 4:1**, Springer, v. 4, p. 193–201, 12 1973. ISSN 1436-4646. Available: <https://link.springer.com/article/10.1007/BF01584660>. Citation on page 21.

RAHIMI, A.; RECHT, B. Random features for large-scale kernel machines. In: . [S.l.: s.n.], 2008. Citations on pages 19, 39, 41, 46, and 67.

______. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. **Advances in Neural Information Processing Systems**, v. 21, 2008. Citation on page 41.

RASMUSSEN, C. E.; WILLIAMS, C. K. I. **Gaussian processes for machine learning**. [S.l.: s.n.], 2006. ISSN 10236090. ISBN 026218253X. Citations on pages 20, 43, and 67.

ROCKAFELLAR, R. T. **Convex Analysis**. [S.l.]: Princeton University Press, 1970. 451 p. ISBN 0691015864. Citations on pages 25 and 26.

RUAN, F.; LIU, K.; JORDAN, M. I. Taming nonconvexity in kernel feature selection – favorable properties of the laplace kernel. 6 2021. Available: <http://arxiv.org/abs/2106.09387>. Citation on page 45.

RUDI, A.; CARRATINO, L.; ROSASCO, L. Falkon: An optimal large scale kernel method. 5 2017. Available: <https://arxiv.org/abs/1705.10958>. Citations on pages 19 and 41.

RUDIN, W. **Fourier analysis on groups**. Dover ed. [S.l.]: Dover, 2017. ISBN 047152364X. Citation on page 39.

SCETBON, M.; MEUNIER, L.; ROMANO, Y. An asymptotic test for conditional independence using analytic kernel embeddings. In: . PMLR, 2022. p. 19328–19346. ISSN 2640-3498. Available: <https://proceedings.mlr.press/v162/scetbon22a.html>. Citation on page 19.

SCHÖLKOPF, B.; SMOLA, A. J. **Learning with kernels: Support vector machines, regularization, optimization, and beyond**. [S.l.]: The MIT Press, 2002. ISBN 0-262-19475-9. Citations on pages 31 and 36.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. 7 2017. Available: <https://arxiv.org/abs/1707.06347>. Citation on page 27.

SCULLEY, D.; SNOEK, J.; RAHIMI, A.; WILTSCHKO, A. Winner's curse? on pace, progress, and empirical rigor. In: . [S.l.: s.n.], 2018. p. 1–4. Citations on pages 51 and 54.

SERVÉN, D.; BRUMMITT, C. **pyGAM: Generalized Additive Models in Python**. 2018. Available: <https://doi.org/10.5281/zenodo.1208723>. Citation on page 53.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: From theory to algorithms**. Cambridge University Press, 2014. ISBN 9781107298019. Available: <http://ebooks.cambridge.org/ref/id/CBO9781107298019>. Citations on pages 23 and 24.

SHEKHAR, S.; KIM, I.; RAMDAS, A. A permutation-free kernel two-sample test. 11 2022. Available: <http://arxiv.org/abs/2211.14908>. Citation on page 19.

SUTHERLAND, D. J.; SCHNEIDER, J. **On the error of Random Fourier Features**. 2015. Citation on page 41.

WAINWRIGHT, M. J. **High-dimensional statistics**. Cambridge University Press, 2019. ISBN 9781108627771. Available: <https://www.cambridge.org/core/product/identifier/9781108627771/type/book>. Citations on pages 24, 35, and 36.

XU, Y.; YIN, W. Block stochastic gradient iteration for convex and nonconvex optimization. 8 2014. Available: <http://arxiv.org/abs/1408.2597>. Citation on page 47.

ZHANG, R.; MUANDET, K.; SCHöLKOPF, B.; IMAIZUMI, M. Instrument space selection for kernel maximum moment restriction. 6 2021. Available: <https://arxiv.org/abs/2106.03340>. Citation on page 19.

APPENDIX A

PROOFS

A.1 Proof of Theorem 2

Proof. Since f is L-smooth, for all $x, y \in \mathbb{R}^n$ it holds that

$$f(y) \leq f(x) + \nabla f(x)^\intercal (y - x) + \frac{L}{2}\|y - x\|_2^2.$$

In particular, setting $y = x - t\nabla f(x)$, we get

$$f(y) \leq f(x) - \left(1 - \frac{Lt}{2}\right) t\|\nabla f(x)\|_2^2.$$

Using the first-order characterization of convexity and taking $t \leq 1/L$, it follows that

$$f(y) \leq f(x^*) + \nabla f(x)^\intercal (x - x^*) - \frac{t}{2}\|\nabla f(x)\|_2^2.$$

Now, from the definition of y,

$$\nabla f(x) = \frac{x - y}{t},$$

we get

$$\begin{aligned} f(y) &\leq f(x^*) + \frac{1}{t}(x - y)^\intercal (x - x^*) - \frac{1}{2t}\|y - x\|_2^2 \\ &= f(x^*) + \frac{1}{2t}\left(2(x - y)^\intercal (x - x^*) - \|y - x\|_2^2\right) \\ &= f(x^*) + \frac{1}{2t}\left(2(y - x)^\intercal (x - x^*) - \|y - x^*\|_2^2 + \|y - x^*\|_2^2 - \|y - x\|_2^2\right) \\ &= f(x^*) + \frac{1}{2t}\left(\|x - x^*\|_2^2 - \|y - x^*\|_2^2\right). \end{aligned}$$

Rewrite the previous equation as

$$f(y) - f(x^*) \leq \frac{1}{2t}\left(\|x - x^*\|_2^2 - \|y - x^*\|_2^2\right).$$

Considering y to be the gradient descent iterates of the form $x^{(i)} = x^{(i-1)} - t\nabla f(x^{(k)})$ and summing all indexes $i \in [k]$,

$$\begin{aligned}\sum_{i=1}^{k}\left[f(x^{(i)}) - f(x^*)\right] &= \frac{1}{2t}\sum_{i=1}^{k}\left[\|x^{(i-1)} - x^*\|_2^2 - \|x^{(i)} - x^*\|_2^2\right] \\ &= \frac{1}{2t}\|x^{(0)} - x^*\|_2^2 - \|x^{(k)} - x^*\|_2^2 \\ &\leq \frac{1}{2t}\|x^{(0)} - x^*\|_2^2.\end{aligned}$$

Since $f(x^{(k)}) \leq \cdots \leq f(x^{(1)})$, we have

$$k(f(x^{(k)} - f(x^*)) \leq \sum_{i=1}^{k}\left[f(x^{(i)}) - f(x^*)\right] \leq \frac{1}{2t}\|x^{(0)} - x^*\|_2^2,$$

which results in

$$f(x^{(k)}) - f(x^*) \leq \frac{1}{2tk}\|x^{(0)} - x^*\|_2^2.$$

□

A.2 Proof of Theorem 3

Proof. Since f is L-smooth for all $x, y \in \mathbb{R}^n$ it holds that

$$f(y) \leq f(x) + \nabla f(x)^\intercal(y - x) + \frac{L}{2}\|y - x\|_2^2.$$

In particular, setting $y = x - t\nabla f(x)$, we get

$$f(y) \leq f(x) - \left(1 - \frac{Lt}{2}\right) t\|\nabla f(x)\|_2^2.$$

In the setting $t \leq 1/L$, we have

$$f(y) \leq f(x) - \frac{t}{2}\|\nabla f(x)\|_2^2.$$

Considering the descent iterations of the form $x^{(i+1)} = x^{(i)} - t\nabla f(x^{(i)})$, previous equation reads

$$f(x^{(i+1)}) \leq f(x^{(i)}) - \frac{t}{2}\|\nabla f(x^{(i)})\|_2^2.$$

Now, using that the minimum is always less than the average, we get

$$\begin{aligned}\min_{0\leq i\leq k}\|\nabla f(x^{(i)})\|_2^2 &\leq \frac{1}{k+1}\sum_{i=0}^{k}\|f^{(i)}\|_2^2 \\ &\leq \frac{2}{t(k+1)}\sum_{i=0}^{k}\left[f(x^{(i)}) - f(x^{(i+1)})\right] \\ &= \frac{2}{t(k+1)}\left[f(x^{(0)}) - f(x^{(k+1)})\right] \\ &\leq \frac{2}{t(k+1)}\left[f(x^{(0)}) - f(x^*)\right].\end{aligned}$$

□

A.3 Proof of Proposition 4

Proof. Let us denote the p-dimensional vector of ones as $\mathbf{1}_p$. The density of the spectral measure of k_λ is, by Bochner's Theorem,

$$\begin{aligned} p_\lambda(\omega) &= \frac{1}{2\pi}\int e^{-i\omega^\intercal(x-y)} k_\lambda(x,y)d\delta \\ &= \frac{1}{2\pi}\int e^{-i\omega^\intercal\delta} h(\lambda\circ\delta)d\delta, \end{aligned}$$

where $\delta = x - y$. Define the new variable $b = \lambda \circ \delta = (\lambda_1\delta_1, \dots, \lambda_p\delta_p)$, then, by the multivariate change of variables theorem, we get

$$\begin{aligned} p_\lambda(\omega) &= \frac{1}{2\pi}\int e^{-i\left(\omega\circ\frac{1}{\lambda}\right)^\intercal b} h(b)\frac{1}{|\lambda_1\cdots\lambda_p|}db \\ &= \frac{1}{|\lambda_1\cdots\lambda_p|}\frac{1}{2\pi}\int e^{-i\left(\omega\circ\frac{1}{\lambda}\right)^\intercal b} h(b)db \\ &= \frac{1}{|\lambda_1\cdots\lambda_p|}p\left(\omega\circ\frac{1}{\lambda}\right), \end{aligned} \tag{A.1}$$

with $p(\cdot)$ the spectral measure of the kernel with $\lambda = \mathbf{1}_p$.

Additionally, since $k_\lambda(x,y) = k_\lambda(y,x)$, we have that $h(\lambda\circ\delta) = h(-\lambda\circ\delta)$ for all $\delta \in \mathbb{R}^p$. In particular, for $\lambda = \mathbf{1}_p$, we have $h(\delta) = h(-\delta)$. Now, the density of the spectral measure of k_λ with $\lambda = \mathbf{1}_p$ reads

$$\begin{aligned} p(\omega) &= \frac{1}{2\pi}\int e^{-i\omega^\intercal(x-y)} k_{\mathbf{1}_p}(x,y)d\delta \\ &= \frac{1}{2\pi}\int e^{-i\omega^\intercal\delta} h(\delta)d\delta \\ &= \frac{1}{2\pi}\int e^{-i\omega^\intercal\delta} h(-\delta)d\delta. \end{aligned}$$

Let $\delta' = -\delta$, then

$$\begin{aligned} p(\omega) &= \frac{1}{2\pi}\int e^{-i(-\omega)^\intercal\delta'} h(\delta')d\delta' \\ &= p(-\omega). \end{aligned}$$

□

APPENDIX

B

DATA PROCESSING

B.1 SE1, SE2, Comp-Act, and Ailerons

We first split the datasets into training, validation, and testing parts. We then normalized and centered the original features in the training sample. Next, we centered and normalized the validation and testing samples using the sample mean and the sample variance from the training set.

B.2 Amazon Fine Food Reviews

We first converted the review rating (ranging from 1 to 5) to a binary response, setting $y = 0$ to reviews with ratings 1 and 2 and $y = 1$ to reviews with ratings 4 and 5. We then dropped duplicate entries and created a matrix containing only user reviews. Next, we applied the Snowball (Porter) stemmer from the library NLTK and removed punctuation, HTML tags, marks, and stopwords. Subsequently, we used the term frequency-inverse document frequency vectorization (TF-IDF), keeping the 5000 most frequent unigrams. We then split the dataset into training, validation, and testing parts. Finally, we centered and normalized the TF-IDF matrix of the training split. We centered and normalized the TF-IDF matrix of validation and testing split using the sample mean and the sample variance from the training set.

B.3 Higgs

We split the dataset into training, validation, and testing parts. Then, we centered and normalized the features in training, validation, and testing splits as done for the previous experiments.

ICMC USP
SÃO CARLOS