Skolkovo Institute of Science and Technology

GAUSSIAN PROCESS MODELS FOR LARGE-SCALE PROBLEMS

Doctoral Thesis

by

Yermek Kapushev

DOCTORAL PROGRAM IN COMPUTATIONAL AND DATA SCIENCE AND ENGINEERING

Supervisor
Associate Professor Evgeny Burnaev

Moscow, 2021

I hereby declare that the work presented in this thesis was carried out by myself at Skolkovo Institute of Science and Technology, Moscow, except where due acknowledgement is made, and has not been submitted for any other degree.

Yermek Kapushev
Associate Professor Evgeny Burnaev

Abstract

This thesis is devoted to the problem of building large-scale models based on Gaussian Processes (GP). We consider two cases: (1) the data sets in which input points lie on a multi-dimension grid, and (2) general data sets without any specific structure. For the first case, we develop a technique for calculating an exact inference for the GP regression model by applying tensor arithmetic that efficiently handles the structure of the data set. The proposed approach can also deal with missing values, which are often a problem in practical applications. For the second case of unstructured data sets, we developed a kernel approximation technique based on an integral representation of the kernel function and special quadrature rules. We show that our approach is a generalization of several prominent papers in this area. The experimental section demonstrates superiority of the proposed technique compared to other methods. Finally, we develop several methods based on the proposed large-scale models for three different problems: tensor completion, density estimate and simultaneous localization and mapping. This very diverse set of problems demonstrate how our models can be built into different pipelines and show some advantages of this approach.

Publications

1. Belyaev, M., Burnaev, E., Kapushev, Y. (2016). Computationally efficient algorithm for gaussian process regression in case of structured samples. Computational Mathematics and Mathematical Physics, 56(4), 499-513.

2. Belyaev, M., Burnaev, E., Kapushev, Y., Panov, M., Prikhodko, P., Vetrov, D., Yarotsky, D. GTApprox: surrogate modeling for industrial design. Advances in Engineering Software 102 (2016) 29–39 (Q1)

3. Munkhoeva, M., Kapushev, Y., Burnaev, E., Oseledets, I. (2018). Quadrature-based features for kernel approximation. In Advances in Neural Information Processing Systems (pp. 9147-9156).

4. Kapushev, Y., Oseledets, I., & Burnaev, E. (2020). Tensor completion via Gaussian process based initialization. Accepted by SIAM Journal on Scientific Computing.

5. Tsimboy, O., Kapushev, Y., Burnaev, E., Oseledets, I. Denoising score matching with random Fourier features. Submitted to Neurocomputing.

6. Kapushev, Y., Kishkun, A., Ferrer, G., Burnaev, E. Random Fourier features based SLAM. Submitted to 2021 IEEE International Conference on Robotics and Automation.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

Scalar numbers are denoted by unbolded letters (x), bold $\mathbf{x}$ represents a vector and x_i is an i-th element of the vector, bold capital letter $\mathbf{X}$ represents a matrix, while calligaphic capital $\mathcal{X}$ is typically used to denote a tensor.

Symbol	Meaning
$\lvert\mathbf{K}\rvert$	determinant of $\mathbf{K}$ matrix
$\langle f, g\rangle_{\mathcal{H}}$	RKHS inner product
$\lVert f\rVert_{\mathcal{H}}$	RKHS norm
∇ or $\nabla_{\mathbf{x}}$	partial derivatives w.r.t $\mathbf{x}$
Δ	trace of the (Hessian) matrix of second derivatives
$\partial_i^k f(\mathbf{x})$	k-th order partial derivative of $f(\mathbf{x})$ w.r.t. i-th input variable x_i
$\partial^{\mathbf{p}} f(\mathbf{x})$	higher order partial derivative $\partial^{\mathbf{p}} f(\mathbf{x}) = \frac{\partial^{p_1+\cdots+p_d}}{\partial x_1^{p_1}\cdots\partial x_d^{p_d}} f(\mathbf{x})$, where $\mathbf{p}$ is a multi-index
$\partial^{\mathbf{p},\mathbf{q}} k(\mathbf{x}, \mathbf{x}')$	higher order partial derivative of the kernel function, where multi-index $\mathbf{p}$ corresponds to the derivatives w.r.t. the first argument of the kernel function and $\mathbf{q}$ corresponds to the second argument
$\mathcal{D}$	Data set $\mathcal{D} = (\mathbf{X}, \mathbf{y}) = \{\mathbf{x}_i, y_i\}_{i=1}^N$
d	dimension of input vectors $\mathbf{x}_i$
$\mathbb{E}$ or $\mathbb{E}_{p(x)}[f(x)]$	expectation of $f(x)$ when $x \sim p(x)$
$\mathbf{1}$ or $\mathbf{1}_n$	vector of all 1's (of length n)
$\mathbf{I}$ or $\mathbf{I}_n$	identity matrix (of length n)
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	Gaussian distribution with mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ
$\mathcal{GP}$	Gaussian Process
$k(\mathbf{x}, \mathbf{x}')$	covariance or kernel function evaluated at points $\mathbf{x}$ and $\mathbf{x}'$

$\mathbf{K}$ or $k(\mathbf{X}, \mathbf{X})$	$N \times N$ covariance matrix with elements $k(\mathbf{x}_i, \mathbf{x}_j), i, j = 1, \ldots, N$
$\mathbf{K_y}$	covariance or kernel matrix for noisy observations $\mathbf{y}$; for independent homoscedastic noise $\mathbf{K}_y = \mathbf{K} + \sigma^2_{noise}\mathbf{I}$
$\mathbf{k}$ or $k(\mathbf{X}, \mathbf{x}_*)$	vector of kernel values between test point $\mathbf{x}_*$ and the training points $\mathbf{X}$
N	number of training points
D	number of random features
$\mathcal{O}(\cdot)$	big O notation
$\phi(\mathbf{x}_i)$	feature map of input $\mathbf{x}_i$
$\mathbf{\Phi}$	matrix where each row is a feature map vector of input variable $\mathbf{x}_i$, $i = 1, \ldots, N$
$\partial\mathbf{\Phi}$ and $\partial^2\mathbf{\Phi}$	matrix of the first and second derivatives of the feature map vectors correspondingly
GP and GPR	Gaussian Process and Gaussian Process Regression
MSE and RMSE	Mean squared error and root mean squared error
MLE	Maximum likelihood estimation
DoE	Design of Experiment
RFF	Random Fourier Features
RKHS	Reproducing Kernel Hilbert Space
RBF	Radial basis functions
TT	Tensor train
SLAM	Simultaneous localization and mapping

Chapter 1

Introduction

Learning dependencies from data with neural networks has become ubiquitous. However, when it comes to problems without any structure in inputs (which is often the case for tabular data) other techniques can provide accurate results. Bayesian approaches are ones of the most appealing. First, they allow to avoid over-fitting by imposing prior distribution on the parameters of the model and then marginalizing them out which acts as a regularization. Second, Bayesian models are probabilistic models, so they can provide uncertainty estimate of the predictions. Gaussian Process (GP) based models is one of the most popular bayesian tools, especially for regression tasks, which is often applied to model physical characteristics, time series forecasting, object tracking, and many others.

Technology and availability of computational resources allow generating a lot of data for a given problem. So, to succeed in solving the problem the most crucial role plays accurate analysis of the obtained data. Generally, in data-driven approaches the more data we have, the better model we can construct. The standard approach to build GP models heavily suffers from the computational complexity that grows cubically with the data set's size. Therefore, developing large-scale GP approaches is an important research direction.

When we need to analyze some object of phenomenon, the data generation process is one of the initial stages in analysis. The properly generated data set allows capturing all the peculiarities of the characteristics we are interested in. In many engineering applications the data is often generated on a grid. This can be encouraged by the experimental setup. For example, designing some assembly part engineer can run a computer simulation to measure its characteristics or can create the detail in full-scale and conduct live experiments. In the former case there are some parameters that can be easily changed, like some external conditions, so it is natural to choose their values

on a grid. The parameters of the detail itself are much harder or expensive to change. In this case, the data set has a particular structure. Sometimes, there can be missing points in this structure due to some simulation errors or the infeasibility of some set of parameters. Nevertheless, having the structure in the data set, even with missing points, can be utilized to build more efficient algorithms. In many cases, however, the data set is unstructured. To cover such cases approximate models should be developed.

Gaussian Processes is a simple yet powerful model that can be used as a part of other complex systems, e.g., Bayesian optimization, multi-fidelity modeling, etc. Developing accurate large-scale GP models allows us to apply such models in other systems. For example, GP models can be incorporated into deep neural networks, density estimate pipeline or simultaneous localization and mapping problems in robotics. Using GP models not only may result in better accuracy, but can also provide an analytical solution, regularize the final model, etc.

To sum up, the development of large-scale Gaussian Process models for structured and unstructured data sets is an actual research direction. Providing examples on how to use different properties of such GP models in different data-driven systems is essential and has a lot of practical interest.

The topic of this thesis is large-scale Gaussian Process models and its applications. The subject of the research is methods to build GP models in case of large structured and unstructured data sets and approaches to incorporate such models into different data-driven systems. The main goal of this work is to develop computationally efficient methods to fit large-scale GP models for structured and unstructured data sets and to provide examples of building such models into different systems. To achieve this goal, the following tasks should be considered:

1. Development of the computationally efficient method for GP inference that takes into account the special structure of the data set.
2. Development of the computationally efficient approximation for the GP inference in case of unstructured data sets.
3. Implement the proposed approaches, demonstrate ways to build large-scale GP methods into different models.

The scientific novelty of this work includes:

- Computationally efficient approach for exact GP inference in case of structured data sets with possible missing points.

- A new technique for approximation of the kernel function that enables computationally efficient GP inference.
- Theoretical analysis of the developed kernel function approximation.
- We developed several approaches based on the proposed methods and achieve state-of-the-art performance. In particular, we designed algorithms for tensor completion, probability density estimate and simultaneous localization and mapping.

The practical significance of the developed methods was demonstrated on a set of real-world engineering problems. The developed approaches for tensor completion problems, probability density estimate, simultaneous localization and mapping all based on the proposed methods justifies the broad applicability of the obtained results.

The reliability of the presented results presented is supported by double-blind reviews of the results at top international conferences; by presentations and seminars and various academic venues; by conducted numerical experiments.

The structure of the thesis is the following. The rest of this chapter gives an introduction to Gaussian Process models. Chapter 2 describes efficient construction for the case of full factorial or incomplete factorial Design of Experiments. In Chapter 3, we develop general approach for unstructured data sets based on random features. Chapter 4 is dedicated to applications of the proposed approaches on several problems, namely, density estimation, tensor completion and simultaneous localization and mapping. In this chapter, we give the necessary background on the kernel methods and Gaussian Processes.

1.1 Gaussian Process Regression

We will consider mainly a regression task. Let $f(\mathbf{x})$ be some unknown smooth function. Suppose that we are given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^N$ of N pairs of inputs $\mathbf{x}_i$ and outputs y_i. This set we will also call *training set*. We would like to construct an approximation $\hat{f}(\mathbf{x})$ of the function $f(\mathbf{x})$ where outputs y_i are assumed to be noisy with additive i.i.d. Gaussian noise:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2_{noise}). \tag{1.1}$$

The noise level is usually not known.

GP regression is a Bayesian approach where a prior distribution over continuous functions is assumed to be a Gaussian Process. This means that any set of function values

$f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)$ have a joint Gaussian distribution (Rasmussen & Williams, 2006)

$$\mathbf{f} \,|\, \mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}_f), \tag{1.2}$$

where $\mathbf{f} = \begin{pmatrix} f(\mathbf{x}_1) & \ldots & f(\mathbf{x}_N) \end{pmatrix}^\top$ is a vector of function values, $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_N \end{pmatrix}^\top$ is a matrix of inputs, $\boldsymbol{\mu} = \begin{pmatrix} \mu(\mathbf{x}_1) & \ldots & \mu(\mathbf{x}_N) \end{pmatrix}^\top$ is a mean vector for some function $\mu(\mathbf{x})$, $\mathbf{K}_f = \{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^N$ is a covariance matrix. The value of the function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - \boldsymbol{\mu}(\mathbf{x}))(f(\mathbf{x}') - \boldsymbol{\mu}(\mathbf{x})')\right]$ is the covariance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ and is called the *covariance function* or the *kernel function*. We will also write Gaussian process as

$$f(\mathbf{x}) \sim \mathcal{GP}(\boldsymbol{\mu}(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

In GP regression we assume that $f(\mathbf{x})$ from (1.2) is a Gaussian Process. Following a Bayesian approach, we would like to derive the posterior distribution of a value y^* at some arbitrary point $\mathbf{x}^*$ given the observed data $(\mathbf{X}, \mathbf{y})$. As the joint distribution of $(y^*, \mathbf{y})$ is Gaussian, the posterior distribution being conditional is also Gaussian

$$\begin{aligned} y^* | \mathbf{X}, \mathbf{y}, \mathbf{x}^* &\sim \mathcal{N}(\hat{\mu}(\mathbf{x}^*), \hat{\sigma}^2(\mathbf{x}^*)), \\ \hat{\mu}(\mathbf{x}^*) &= \mu(\mathbf{x}^*) + \mathbf{k}(\mathbf{x}^*)^\top \mathbf{K}_y^{-1}(\mathbf{y} - \boldsymbol{\mu}), \\ \hat{\sigma}^2(\mathbf{x}^*) &= k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^\top \mathbf{K}_y^{-1} \mathbf{k}(\mathbf{x}^*), \end{aligned} \tag{1.3}$$

where $\mathbf{k}(\mathbf{x}^*) = \begin{pmatrix} k(\mathbf{x}^*, \mathbf{x}_1) & \ldots k(\mathbf{x}^*, \mathbf{x}_N) \end{pmatrix}^\top$ and $\mathbf{K}_y = \mathbf{K}_f + \sigma^2_{noise}\mathbf{I}$. The posterior mean we use as a prediction and the posterior variance can be used as an uncertainty estimate of the prediction. It is common practice to use zero-mean function as it can be accounted in the kernel function.

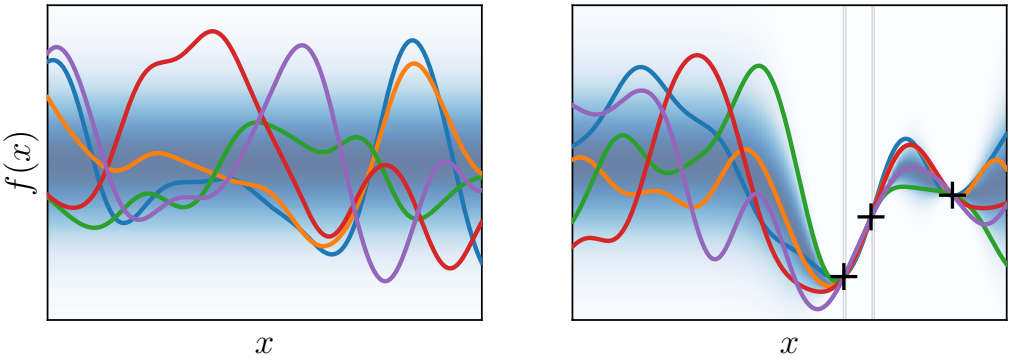


FIGURE 1.1: Illustration of Gaussian Process Regression in a one-dimensional case. The shades of blue represent the distribution of function values at each input point x. Color lines are random functions sampled from the GP. *Left*: figure depicts prior distribution over functions with several random functions drawn from it. *Right*: posterior distribution conditioned on several data observations. Here the noise in observations is very small, therefore, all posterior samples pass through the given points.

The kernel function is central and the most crucial part of GP models. The type of kernel specifies which class of functions can be approximated by GP and what peculiarities the model will possess. By specifying a kernel we can end up with models that are equivalent to other well known models, like linear regression or splines. However, in practice usually a squared exponential (it is also often called Radial Basis Function (RBF) kernel) covariance function is used

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\sum_{i=1}^{d} \theta_i^2 (\mathbf{x}_p^{(i)} - \mathbf{x}_q^{(i)})^2\right). \tag{1.4}$$

This kernel is infinitely smooth and is shift-invariant, i.e. $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ which has its own advantages and disadvantages. Nevertheless, this kernel is highly popular and is a universal choice that works quite well in many applications.

1.1.1 Hyperparameters tuning

One of the appealing property of GP models is that they allow to carefully tune their hyper-parameters without over-fitting to the observed data.

Let us denote hyper-parameters of the kernel function and noise variance by $\boldsymbol{\theta}$. To choose good $\boldsymbol{\theta}$ we consider the log likelihood

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \sigma_f, \sigma_{noise}) = \underbrace{-\frac{1}{2}\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}}_{\text{data fit term}} - \underbrace{\frac{1}{2}\log |\mathbf{K}_y|}_{\text{complexity term}} - \frac{N}{2}\log 2\pi \tag{1.5}$$

and maximize it over the hyper-parameters (Rasmussen & Williams, 2006). The objective is also called *marginal log-likelihood* as it implicitly marginalizes out all function values at other input locations that are not present in the training set. There are essentially two terms in the objective function. One term encourages the data fit and the other one controls the model's complexity. It allows comparing different models and balance between data interpolation and the model's capacity.

1.2 Features of Gaussian Processes

GP models have some peculiarities, some of them makes it more attractive, some — limit their applicability.

1. **Analytical solution**. The predictive posterior distribution is given by a simple analytical expression. This simplifies analysis of the model and can make it easier to build on top of GP model. And it is always nice to have a closed-form solution.

2. **Marginalization**. Marginalization over all function values at locations that are not in the training set is actually averaging over a wide range of functions, which is actually a strong regularization that makes over-fitting less possible.

3. **Flexibility**. Different peculiarities of the data set and underlying unknown function can be taken into account by considering specific kernel functions.

4. **Uncertainty estimation**. Predictive distribution allows us to estimate the uncertainty of the prediction. It can be required by the application itself as well as used to solve other problems, e.g., adaptive Design of Experiments, Bayesian Optimization and others.

5. **Well developed theory**. GP models have a simple structure and, therefore, have already been well studied and many properties have been discovered. This makes the models attractive because their behavior can be explained theoretically. It is also easier to analyze the models that were built on top of GP models. Theoretical analysis of the models is essential as it helps to understand in what cases the model will or will not work, how to pre-process the data or modify the overall approach to make it work and so on.

6. **Computational complexity**. As it can be seen from (1.5) we need to calculate the determinant and the inverse of the kernel matrix of size $N \times N$. Therefore, inference requires $\mathcal{O}(N^3)$ operations, which makes it prohibitive to use GP models in case of large data sets. Test time evaluation complexity is $\mathcal{O}(N)$ and in some applications it can also be too high. In Chapter 2 we show a way to reduce the computational complexity drastically for structured data sets, and in Chapter 3 we consider general unstructured data sets and develop an efficient approximation to the kernel function.

7. **Kernel choice**. Kernel is the most important part of the GP model. It specifies the behavior of the model and a class of functions that can be well approximated. However, in most cases, we do not know in advance what type of kernel best suits the data set at hand and either stick to some universal kernel (like RBF) or construct an appropriate kernel manually. Nevertheless, there are some works that tries to select the best kernel automatically, for example, (Duvenaud, 2014; Abdessalem et al., 2017; Teng et al., 2020). There are works that bypass the kernel design by building Deep Gaussian Processes (Damianou & Lawrence, 2013) or combine deep networks with Gaussian Processes (Wilson et al., 2016).

8. **Gaussian distribution**. In GP the distribution of output values (i.e., likelihood) is Gaussian. Sometimes the actual likelihood is non-Gaussian, for example, when we deal with classification tasks. In this case we need to use approximate solutions.

This direction is well developed and there are several frameworks that allows us to work with non-Gaussian likelihoods (De G. Matthews et al., 2017; Gardner et al., 2018) easily.

In the thesis we approach the computational complexity issue. There is considerable amount of work in this direction (for example, (Quiñonero-Candela & Rasmussen, 2005; Rahimi & Recht, 2008; Titsias, 2009; Cutajar et al., 2017)). Nevertheless, we would like to point that generally exact Gaussian Process models work better than the existing approximations (Cutajar et al., 2016; Wang et al., 2019). Also, typically large-scale approximate methods have hyper-parameter that controls the accuracy of the approximation (for example, number of inducing points in (Quiñonero-Candela & Rasmussen, 2005; Titsias, 2009) and number of features in (Rahimi & Recht, 2008; Felix et al., 2016) and their follow-ups). The analysis of such approaches show that optimal learning rates requires this parameter to be at least $\mathcal{O}(\sqrt{N}\log N)$ (Rudi & Rosasco, 2017; Rudi et al., 2017). Therefore, the computational complexity becomes $\mathcal{O}(N\sqrt{N})$ in this case. In the thesis we prefer to do exact inference when it is possible. In case of large-scale data sets it can be done if there is a way to exploit the structure of the problem. Such approach we develop in Chapter 2 for data sets on multi-dimensional grids and show that its computational complexity is lower than the complexity of the large-scale approximations.

In case of unstructured data sets the way to build large-scale Gaussian Process model is to do approximations. Current state-of-the-art scalable approaches show great effectiveness in real-world problems. Better performance is usually achieved by *data-dependent* approaches, i.e. the approaches that rely on the given data set. On the other hand, in some applications *data-independent* techniques are more attractive as they do not require the data set to construct an approximation. For example, applications where we data changes frequently, so it is more costly or less accurate to use data-dependent techniques. Simultaneous localization and mapping is an example of such problems (we consider it Chapter 4). For this reasons in the thesis we focus on data-independent approach.

Chapter 2

Gaussian Process Models on Multi-dimensional Grids

Let us consider the case when the training set has a special structure called *factorial Design of Experiments* (DoE) (Montgomery, 2006). In this structure, all the input variables are grouped into several *factors*, and the training set consists of the Cartesian product of the factors. Such experimental designs arise naturally in many applications, especially in engineering. Imagine we want to model some aerodynamic characteristic of an airfoil depending on the shape of the airfoil and external conditions, like Mach number and angle of attack. There are two ways to do it. The first one is conducting computational fluid dynamic simulation to estimate the desired characteristics. Another way is building an airfoil and conducting wind tunnel experiments. In the latter case, though it is expensive to build airfoils, it is easy to change the external conditions so that we can select the external parameters on a regular grid for each airfoil. As a result, we have two sets of parameters. One set describes the geometry of the airfoil, the other one - external conditions. And in our measurements, those parameters have the Cartesian product structure. In some cases, we can have missing values in such designs of the experiments. It can either be some errors in measurements, computationally unstable procedures, or just decrease the data set size. In this case, we have *incomplete factorial Design of Experiments*.

The problem with such designs is that the training set size grows exponentially with the number of factors. The complexity of standard GP models does not allow to work with such training sets. There are several methods for large-scale GP modeling; although they are not exact and approximate, either the kernel function or the output of the GP model itself. These approaches are general and can be applied to any data set. More details can be found in Chapter 3.

However, in case of full or incomplete factorial DoE, the special structure of the data set can be exploited to derive computationally efficient and the exact inference procedure. There aren't many regression methods that consider the design of experiments. There are several methods based on splines, which consider this special structure of the given data (Stone et al., 1997). A disadvantage of these methods is that they work only with one-dimensional factors and cannot be applied to a more general case when the factors are multidimensional. Another shortcoming is that such approaches do not have approximation accuracy evaluation procedure. The missing values cannot be handled as well.

There is another problem that we are likely to encounter. Factor sizes can vary significantly. Engineers usually use large factors sizes if the corresponding input variables have a significant impact on function values. Otherwise, the factors sizes are likely to be small, i.e., the factor sizes are often selected using the knowledge from a subject domain (Rendall & Allen, 2008). For example, if it is known that dependency on some variable is quadratic, then the size of this factor will be three, as a larger size is redundant. The difference between factor sizes can lead to the degeneracy of the GP model. We will refer to this property of data set as *anisotropy*.

In this chapter, we develop an approach for fast, exact inference of the GP regression model by considering the factorial nature of the design of experiments in a general case of multidimensional factors. Both full factorial and incomplete factorial designs are covered. We also discuss how to choose the initial values of parameters for the GP model and regularization in order to consider possible anisotropy of the training data set.

The paper (Wilson et al., 2014) considers the same problem statement. To efficiently train and evaluate a GP model, one needs a fast procedure to calculate $\mathbf{K}^{-1}\mathbf{y}$, where $\mathbf{K}$ is a kernel matrix, and an efficient procedure to evaluate the determinant of the kernel matrix. The authors of (Wilson et al., 2014) exploit the structure in the data set to efficiently calculate both components in the case of full factorial design of experiments. In the case of incomplete factorial design of experiments, they apply the preconditioned conjugate gradient (PCG) method to calculate $\mathbf{K}^{-1}\mathbf{y}$. It is based on an efficient approach for the matrix-vector product $\mathbf{K}\mathbf{y}$, which, however, gives an exact answer only in the limit when the parameter of the method goes to infinity. Empirically, the PCG in this case requires a small number of iterations (much smaller than the dataset size). In contrast, this chapter of the dissertation shows an approach that calculates $\mathbf{K}^{-1}\mathbf{y}$ in case of incomplete factorial deign of experiments exactly. We also derive the exact number of iterations required for conjugate gradients approach to converge. We show that the method is efficient when the number of missing points is small or very large. Another difference is that we calculate the determinant of the kernel matrix exactly but in about $\mathcal{O}(N^2)$ operations, whereas

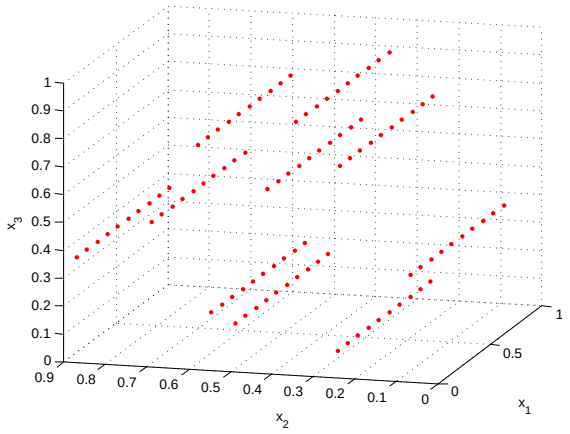


FIGURE 2.1: Example of a multidimensional factor. In the figure, x_1 is a usual one-dimensional factor and (x_2, x_3) is a two-dimensional factor.

(Wilson et al., 2014) make an approximation to the determinant but in $\mathcal{O}(N)$ iterations. We also introduce a special regularization that allows to reduce degeneration effect.

The main contributions of this chapter are as follows

- We develop a computationally efficient approach for the case of the multi-dimensional data set.
- For the case of multi-dimensional grid with missing points, we derive a conjugate gradient-based approach for matrix inversion, which provably converges in $\mathcal{O}(\min(R, N))$ iterations, where R is the number of missing points.
- We propose a special regularization for the data sets on multi-dimensional grids that makes the GP model less prone to degeneration.

2.1 Factorial design of experiments

Let us refer to sets of points $\omega_k = \{x_{i_k}^k \in \mathrm{X}_k\}_{i_k=1}^{n_k}$, $\mathrm{X}_k \subset \mathbb{R}^{d_k}$, $k = \overline{1, K}$ as *factors*. A set of points Ω is referred to as a factorial design of experiments if it is a Cartesian product of factors

$$\Omega = \omega_1 \times \omega_2 \times \cdots \times \omega_k = \{[x_{i_1}^1, \ldots, x_{i_K}^K], \{i_k = 1, \ldots, n_k\}_{k=1}^K\}. \tag{2.1}$$

The elements of Ω are vectors of a dimension $d = \sum_{i=1}^K d_i$ and the sample size is a product of sizes of all factors $N = \prod_{i=1}^K n_i$. If all the factors are one-dimensional, Ω is a full factorial design. But in a more general case, factors are multidimensional (see example in Figure 2.1).

2.2 Tensor and related operations

Let us introduce tensor notation and some related operations that will be used later.

A *tensor* $\mathcal{Y}$ is a K-dimensional matrix of size $n_1 \times n_2 \times \cdots \times n_K$ (Kolda & Bader, 2009a):

$$\mathcal{Y} = \{y_{i_1,i_2,\ldots,i_K}, \{i_k = 1, \ldots, n_k\}_{k=1}^K\}. \tag{2.2}$$

By $\mathcal{Y}^{(j)}$, we will denote a matrix consisting of elements of the tensor $\mathcal{Y}$ whose rows are $1 \times n_j$ slices of $\mathcal{Y}$ with fixed indices $i_{j+1}, \ldots, i_K, i_1, \ldots, i_{j-1}$ and altering index $i_j = 1, \ldots, n_j$. In case of a 2-dimensional tensor, it holds that $\mathcal{Y}^{(1)} = \mathcal{Y}^T$ and $\mathcal{Y}^{(2)} = \mathcal{Y}$.

Now let us introduce the multiplication of a tensor by a matrix along the direction i. Let B be some matrix of size $n_i \times n_i'$. Then the product of the tensor $\mathcal{Y}$ and the matrix B along the direction i is a tensor $\mathcal{Z}$ of size $n_1 \times \cdots \times n_{i-1} \times n_i' \times n_{i+1} \times \cdots \times n_K$ such that $\mathcal{Z}^{(i)} = \mathcal{Y}^{(i)}B$. We will denote this operation by $\mathcal{Z} = \mathcal{Y} \otimes_i B$. For a 2-dimensional tensor $\mathcal{Y}$, multiplication along the first direction is a left multiplication by matrix $\mathcal{Y} \otimes_1 B = B^T\mathcal{Y}$, and along the second direction — is a right multiplication $\mathcal{Y} \otimes_2 B = \mathcal{Y}B$.

Multiplication of a tensor by a matrix along some direction is closely related to the Kronecker product. Let's consider an operation *vec*, which, for every multidimensional matrix $\mathcal{Y}$ returns a vector containing all elements of $\mathcal{Y}$. An inner product of tensors $\mathcal{Y}$ and $\mathcal{Z}$ is the inner product of vectors $\mathrm{vec}(\mathcal{Y})$ and $\mathrm{vec}(\mathcal{Z})$

$$\langle \mathcal{Y}, \mathcal{Z} \rangle = \langle \mathrm{vec}(\mathcal{Y}), \mathrm{vec}(\mathcal{Z}) \rangle .$$

For every multidimensional matrix $\mathcal{Y}$ of size $n_1 \times n_2 \times \cdots \times n_K$ and $n_i \times p_i$ size matrices B_i, $i = 1, \ldots, K$ the following identity holds (Loan, 2000)

$$(B_1 \otimes B_2 \cdots \otimes B_K)\mathrm{vec}(\mathcal{Y}) = \mathrm{vec}(\mathcal{Y} \otimes_1 B_1^T \cdots \otimes_K B_K^T), \tag{2.3}$$

where symbol $\otimes$ denotes the Kronecker product.

Let's compare the complexity of the right and the left hand sides of (2.3). For simplicity, we assume that all the matrices B_i are quadratic of size $n_i \times n_i$ and $N = \prod n_i$. Then computation of the left-hand side of (2.3) requires N^2 operations (of additions and multiplications) not considering the complexity of the Kronecker product while the right hand side requires $N \sum_i n_i$ operations.

2.2.1 Efficient log-likelihood calculation

Now, let us see how the introduced operations can be used for efficient computation of the predictive distribution and the marginal log-likelihood.

Covariance function (1.4) can be represented as a product of covariance functions each depending only on the variables from one factor

$$k(\mathbf{x}_p, \mathbf{x}_q) = \prod_{i=1}^{K} k_i(x_p^i, x_q^i), \tag{2.4}$$

where $x_p^i, x_q^i \in \mathbb{R}^{d_i}$ belong to the same factor ω_i. For the squared exponential function, we have $k_i(x_p^i, x_q^i) = \omega_{f,i}^2 \exp\left(-\sum_j^{d_i} \left(\theta_i^{(j)}\right)^2 \left(x_p^{(j),i} - x_q^{(j),i}\right)^2\right)$, where $x_p^{(j),i}$ is a j-th component of x_p^i. Note that in general case covariance functions k_i are not necessarily squared exponential, they can be of different types for different factors. It allows to consider the special features of factors (knowledge from a subject domain) if they are known. In such a case, the function defined by (2.4) is still a valid covariance function being the product of separate covariance functions. From now on, we will denote by $\theta_i = (\theta_i^{(1)}, \ldots, \theta_i^{(d_i)})$ the set of hyperparameters for covariance function of the i-th factor and let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$.

Such form of the covariance function and the factorial DoE allows us to represent the covariance matrix as the Kronecker product

$$\mathbf{K}_f = \bigotimes_{i=1}^{K} \mathbf{K}_i, \tag{2.5}$$

where $\mathbf{K}_i$ is a covariance matrix defined by the k_i covariance function computed at points from the i-th factor ω_i.

The Kronecker product of matrices can be efficiently inverted due to the following property

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$$

if A and B are invertible matrices. If A has size $n_a \times n_a$ and B has size $n_b \times n_b$ then the left side of the above equation requires $\mathcal{O}(n_a^3 n_b^3)$ operations while the right hand side requires only $\mathcal{O}(n_a^3 + n_b^3)$ operations and this is much less. However, we have to invert the matrix $\mathbf{K}_y = \mathbf{K}_f + \sigma_{noise}^2 \mathbf{I}$. For this, we use the Singular Value Decomposition (SVD)

$$\mathbf{K}_i = \mathbf{U}_i \mathbf{D}_i \mathbf{U}_i^T,$$

where $\mathbf{U}_i$ is an orthogonal matrix of eigenvectors of matrix $\mathbf{K}_i$ and $\mathbf{D}_i$ is a diagonal matrix of eigenvalues. Using the properties of the Kronecker product and representing

an identity matrix as $\mathbf{I}_{d_i} = \mathbf{U}_i \mathbf{U}_i^T$ we obtain

$$\mathbf{K}_y^{-1} = \left(\bigotimes_{i=1}^{K} \mathbf{U}_i \right) \left(\left[\bigotimes_{i=1}^{K} \mathbf{D}_i \right] + \sigma_{noise}^2 \mathbf{I} \right)^{-1} \left(\bigotimes_{i=1}^{K} \mathbf{U}_i^T \right). \tag{2.6}$$

Computing SVD for all $\mathbf{K}_i$ requires $\mathcal{O}(\sum_k n_k^3)$ operations. Calculation of the Kronecker product in (2.6) has complexity $\mathcal{O}(N^2)$. So, this gives us overall complexity $\mathcal{O}(N^2)$ for calculation of expressions for the log-likelihood, the predictive mean, and the covariance. It is faster than the straightforward calculations; however, it can be improved.

Equations (1.3) and (1.5) for GP regression do not require explicit inversion of $\mathbf{K}_y$. In each equation, it is multiplied by vector $\mathbf{y}$ (or $\mathbf{k}_*$). So, we will compute $\mathbf{K}_y^{-1}\mathbf{y}$ instead of explicitly inverting $\mathbf{K}_y$ and then multiplying it by the vector $\mathbf{y}$.

Let $\mathcal{Y}$ be a tensor containing values of the vector $\mathbf{y}$ such that $\text{vec}(\mathcal{Y}) = \mathbf{y}$. Now using identities (2.3) and (2.6) we can write $\mathbf{K}_y^{-1}\mathbf{y}$ as

$$\begin{aligned} \mathbf{K}_y^{-1}\mathbf{y} = \left(\bigotimes_{i=1}^{K} \mathbf{U}_i \right) \left(\left[\bigotimes_{i=1}^{K} \mathbf{D}_i \right] + \sigma_{noise}^2 \mathbf{I} \right)^{-1} \times \text{vec}(\mathcal{Y} \otimes_1 \mathbf{U}_1 \cdots \otimes_K \mathbf{U}_K) = \\ = \text{vec}\left[\left((\mathcal{Y} \otimes_1 \mathbf{U}_1 \cdots \otimes_K \mathbf{U}_K) * \mathcal{D}^{-1} \right) \otimes_1 \mathbf{U}_1^T \cdots \otimes_K \mathbf{U}_K^T \right], \end{aligned} \tag{2.7}$$

where $\mathcal{D}$ is a tensor constructed by transforming the diagonal of matrix $\left[\bigotimes_k \mathbf{D}_k \right] + \sigma_{noise}^2 \mathbf{I}$ into a tensor.

The elements of the tensor $\mathcal{D}$ are eigenvalues of the matrix $\mathbf{K}_y$, therefore, its determinant can be calculated as

$$|\mathbf{K}_y| = \prod_{i_1,\ldots,i_K} \mathcal{D}_{i_1,\ldots,i_K}. \tag{2.8}$$

Proposition 2.1. *The computational complexity of the log likelihood (1.5), where $\mathbf{K}_y^{-1}y$ and $|\mathbf{K}_y|$ are calculated using (2.7) and (2.8), is*

$$\mathcal{O}\left(\sum_{i=1}^{K} n_i^3 + N \sum_{i=1}^{K} n_i \right). \tag{2.9}$$

Proof. Let's calculate the complexity of computing $\mathbf{K}_y^{-1}\mathbf{y}$ using (2.7). Computation of the matrices $\mathbf{U}_i$ and $\mathbf{D}_i$ requires $\mathcal{O}(\sum_i n_i^3)$ operations. Multiplication of the tensor $\mathcal{Y}$ by the matrices $\mathbf{U}_i$ requires $\mathcal{O}(N \sum_i n_i)$ operations. Further, a component-wise product of the obtained tensor and the tensor $\mathcal{D}^{-1}$ requires $\mathcal{O}(N)$ operations. And the complexity of multiplication of the result by the matrices $\mathbf{U}_i$ is again $\mathcal{O}(N \sum_i n_i)$. The determinant, calculated by equation (2.8), requires $\mathcal{O}(N)$ operations. Thus, the overall complexity of computing (2.7) is $\mathcal{O}(\sum_{i=1}^{K} n_i^3 + N \sum_{i=1}^{K} n_i)$. □

For more illustrative estimate of the computational complexity, suppose that $n_i \ll N$ (number of factors is large and their sizes are close). In this case, it holds that $\mathcal{O}(N\sum_i n_i) = \mathcal{O}(N^{1+\frac{1}{K}})$ and this is much less than $\mathcal{O}(N^3)$.

To optimize the log-likelihood over the hyperparameters, we use a gradient-based method. The derivatives of the log likelihood with respect to the hyperparameters take the form

$$\frac{\partial}{\partial\theta}\left(\log p(\mathbf{y}|\mathbf{X},\sigma_f,\sigma_{noise})\right) = -\frac{1}{2}\mathrm{Tr}(\mathbf{K}_y^{-1}\mathbf{K}') + \frac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{K}'\mathbf{K}_y^{-1}\mathbf{y}, \tag{2.10}$$

where θ is one of the hyperparameters θ_i, σ_{noise} or $\omega_{f,i}, i = 1,\ldots,d$ and $\mathbf{K}' = \dfrac{\partial\mathbf{K}}{\partial\theta}$. $\mathbf{K}'$ is also the Kronecker product

$$\mathbf{K}' = \mathbf{K}_1 \otimes \cdots \otimes \mathbf{K}_{i-1} \otimes \frac{\partial\mathbf{K}_i}{\partial\theta} \otimes \mathbf{K}_{i+1} \otimes \cdots \otimes \mathbf{K}_K,$$

where θ is a parameter of the i-th covariance function. Denoting by $\mathcal{A}$, a tensor such that $\mathrm{vec}(\mathcal{A}) = \mathbf{K}_y^{-1}\mathbf{y}$ the second term in equation (2.10) can be efficiently computed using the same technique as in (2.7):

$$\begin{aligned}\frac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{K}'\mathbf{K}_y^{-1}\mathbf{y} = \Bigg\langle \mathcal{A}, \mathcal{A} \otimes_1 \mathbf{K}_1^T \otimes_2 \cdots \otimes_{i-1} \mathbf{K}_{i-1}^T \otimes_i \frac{\partial\mathbf{K}_i^T}{\partial\theta} \otimes_{i+1} \\ \otimes_{i+1} \mathbf{K}_{i+1}^T \otimes_{i+2} \cdots \otimes_K \mathbf{K}_K^T \Bigg\rangle.\end{aligned} \tag{2.11}$$

The complexity of calculating this term of derivative is the same as the complexity of equation (2.7).

Now, let us compute the first term

$$\begin{aligned}\mathrm{Tr}(\mathbf{K}_y^{-1}\mathbf{K}') &= \mathrm{Tr}\left(\left(\bigotimes_{i=1}^K \mathbf{U}_i\right)\mathbf{D}^{-1}\left(\bigotimes_{i=1}^K \mathbf{U}_i^T\right)\mathbf{K}'\right) = \\ &= \mathrm{Tr}\left(\mathbf{D}^{-1}\left(\bigotimes_{i=1}^K \mathbf{U}_i^T\mathbf{K}_i'\mathbf{U}_i\right)\right) = \\ &= \left\langle \mathrm{diag}\left(\mathbf{D}^{-1}\right), \mathrm{diag}\left(\bigotimes_{i=1}^K \mathbf{U}_i\mathbf{K}_i'\mathbf{U}_i\right)\right\rangle = \\ &= \left\langle \mathrm{diag}\left(\mathbf{D}^{-1}\right), \bigotimes_{i=1}^K \mathrm{diag}\left(\mathbf{U}_i\mathbf{K}_i'\mathbf{U}_i\right)\right\rangle,\end{aligned} \tag{2.12}$$

where $\mathrm{diag}(A)$ is a vector of diagonal elements of a matrix A, $\mathbf{D} = \bigotimes_i \mathbf{D}_i + \sigma_{noise}^2\mathbf{I}$.

The computational complexity of this derivative term is the same as the computational complexity of equation (2.11).

Thus, we obtain

TABLE 2.1: Runtime (in seconds) of tensored GP and original GP algorithms.

	ORIGINAL GP	TENSORED GP
64	0.8	0.16
160	2.69	0.16
432	14.31	0.74
1000	120.38	1.02
2000	970.21	1.11
10240	—	33.18
64000	—	74.9
160000	—	175.15
400000	—	480.14

Proposition 2.2. *The computational complexity of calculating derivatives of the log likelihood is* $\mathcal{O}\left(\sum_{i=1}^{K} n_i^3 + N \sum_{i=1}^{K} n_i\right)$.

Table 2.1 contains training times for original GP and proposed GP regression for different sample sizes. The experiments were conducted on a PC with Intel i7 2.8 GHz processor and 4 GB RAM. For the original GP, we used GPML Matlab code (Rasmussen & Nickisch, 2010). We also adopted the GPML code to use tensor operations. The results illustrate that the proposed approach is much faster than the original GP and allows making approximations using extremely large data sets.

2.2.2 Anisotropy

In this section, we will consider an anisotropy problem. As it was mentioned in an engineering practice, factorial designs are often anisotropic, i.e., the sizes of factors differ significantly. It is a common case for the GP regression to become degenerate in such a situation. Suppose the given DoE consists of two one-dimensional factors with sizes n_1 and n_2. Assume that $n_1 \ll n_2$. Then one could expect the length-scale for the first factor to be much greater than the length-scale for the second factor (or $\theta_1 \ll \theta_2$). However, in practice, we often observe the opposite $\theta_1 \gg \theta_2$. This happens because the optimization algorithm stacks in a local maximum during maximization over the hyperparameters as the objective function (the log-likelihood) is non-convex with lots of local maxima. We get an undesired effect of degeneracy: in the region without training points, the approximation is constant, and it has sharp peaks at training points. This situation is illustrated in Figure 2.3a (compare with the true function in Figure 2.3c).

Let us denote length-scales as $l_k^{(i)} = \left[\theta_k^{(i)}\right]^{-1}$. To incorporate our prior knowledge about factor sizes into regression model we introduce prior distribution on the hyperparameters $\boldsymbol{\theta}$:

$$\frac{\theta_k^{(i)} - a_k^{(i)}}{b_k^{(i)} - a_k^{(i)}} \sim \mathcal{B}e(\alpha, \beta),\ \{i = 1, \ldots, d_k\}_{k=1}^{K}, \tag{2.13}$$

i.e., prior on hyperparameter $\theta_k^{(i)}$ is a beta distribution with parameters α and β scaled to some interval $\left[a_k^{(i)}, b_k^{(i)}\right]$.

The log likelihood then has the form

$$\begin{aligned} \log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \sigma_f, \sigma_{noise}) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \\ -\frac{N}{2}\log 2\pi + \sum_{k,i}\Big((\alpha-1)\log(\theta_k^{(i)}) + \\ +(\beta-1)\log(1-\theta_k^{(i)})\Big) - d\log(\mathrm{B}(\alpha,\beta)), \end{aligned} \tag{2.14}$$

where $\mathrm{B}(\alpha, \beta)$ is a beta function.

By introducing such prior, we restrict parameters $\theta_k^{(i)}$ to belong to some interval $\left[a_k^{(i)}, b_k^{(i)}\right]$ (or length-scales $l_k^{(i)}$ to belong to the interval $\left[\left(b_k^{(i)}\right)^{-1}, \left(a_k^{(i)}\right)^{-1}\right]$). It seems reasonable that for an approximation to fit the training points the length-scale is not needed to be much less than the distance between points. That's why we choose the lower bound for the length-scale $l_k^{(i)}$ to be $c_k * \min\limits_{x,y\in\omega_k, x^{(i)}\neq y^{(i)}} ||x^{(i)} - y^{(i)}||$ and the upper bound for the length-scale to be $C_k * \max\limits_{x,y\in\omega_k} ||x^{(i)} - y^{(i)}||$. The value c_k should be close to 1. If it is too small, we are taking risks to overfit the data by allowing small length-scales. If c_k is too large, we will underfit the data by allowing only large length-scales and forbidding small ones. Constants C_k must be much greater than c_k to permit large length-scales and preserve flexibility. In this work, we used $c_k = 0.5$ and $C_k = 100$. Such values of c_k and C_k worked rather well in our test cases.

Parameters of beta distribution was set to $\alpha = \beta = 2$ to get symmetrical probability distribution function (see Figure 2.2) because we do not know a priori if the values of GP parameters should be large or small. The chosen prior distribution penalizes too large and too small values of parameters θ_k as they are undesirable.

Figure 2.3b illustrates the approximation of the GP regression with introduced prior distribution (and initialization described in Section 2.2.3). The hyperparameters were chosen such that the approximation is nondegenerate.

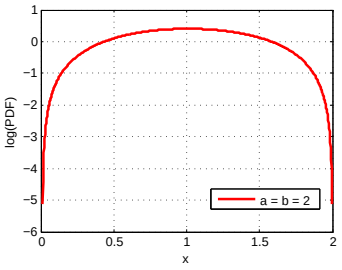


FIGURE 2.2: Logarithm of Beta distribution probability density function, rescaled to $[0.01, 2]$ interval, with parameters $\alpha = \beta = 2$.

2.2.3 Initialization

It is also important to choose reasonable initial values of hyperparameters in order to converge to a good solution during parameter optimization. The kernel-widths for different factors should have different scales because corresponding factor sizes have different numbers of levels. Hence, it seems reasonable to use the average distance between points in a factor as an initial value

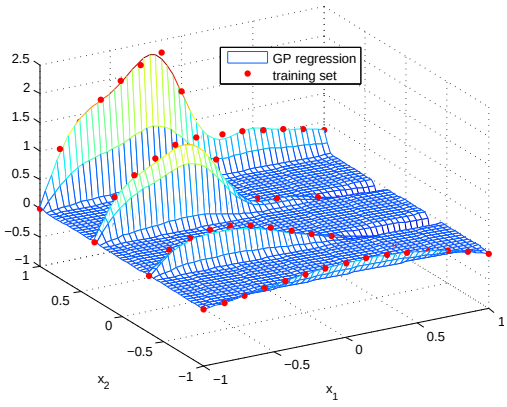
$$\theta_k^{(i)} = \left[\frac{1}{n_k} \left(\max_{x \in \omega_k}(x^{(i)}) - \min_{x \in \omega_k}(x^{(i)}) \right) \right]^{-1}. \tag{2.15}$$

2.3 Experiments

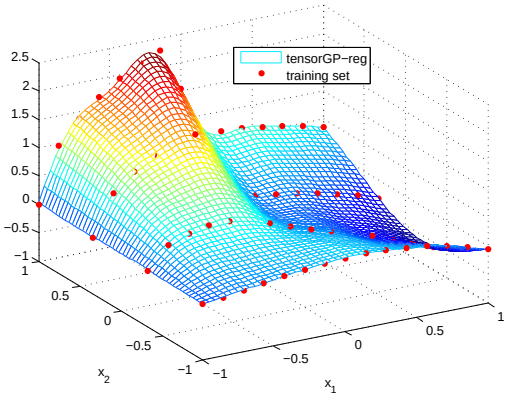
The proposed algorithm was tested on a set of test functions (Evoluationary computation pages — the function testbed; System optimization — testroblems). The functions have different input dimensions from 2 to 6 and the sample sizes N varied from 100 to about 200000. For each function, several factorial anisotropic training sets were generated. We will test the following algorithms: GP with tensor computations (tensorGP), GP with tensor computations and prior distribution (tensorGP-reg), the sparse pseudo-point input GP (FITC) (Snelson & Ghahramani, 2005). For the FITC method, we used GPML Matlab code (Rasmussen & Nickisch, 2010). The number of inducing points of FITC algorithm varied from $M = 500$ for small samples (up to 5000 points) to $M = 70$ for large samples (about 10^5 points) in order to obtain approximation in reasonable time (complexity of FITC algorithm is $\mathcal{O}(M^2 N)$). For tensorGP and tensorGP-reg, we adopted GPML code to use tensor operations, proposed prior distribution and initialization.

To assess the quality of approximation, a mean squared error was used

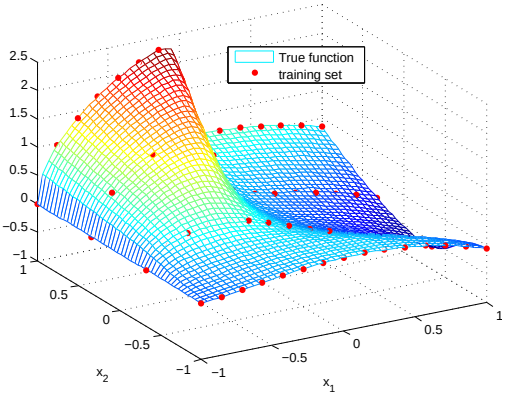
$$\text{MSE} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))^2. \tag{2.16}$$



(A) Degeneracy of the GP model in case of anisotropic data set. The length-scales for this model are $l_1 = 0.286, l_2 = 0.033$ whereas factor sizes are $n_1 = 15, n_2 = 4$.



(B) The GP regression with proposed prior distribution and initialization.



(C) True function.

To compare different algorithms, Dolan-Moré curves are used (Dolan & Moré, 2002). The idea of Dolan-Moré curves is as follows. Let $t_{p,a}$ be an error of an a-th algorithm on a p-th problem and $r_{p,a}$ be a performance ratio

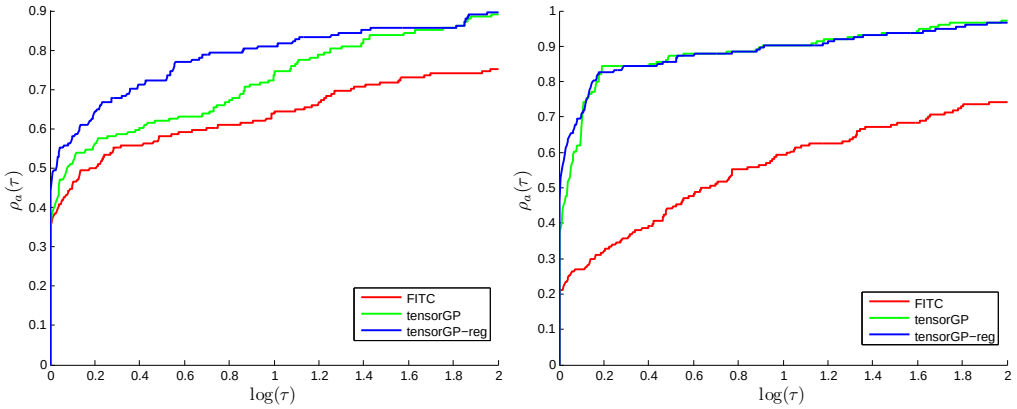
$$r_{p,a} = \frac{t_{p,a}}{\min\limits_{s}(t_{p,s})}.$$

Then Dolan-Moré curve is a graph of $\rho_a(\tau)$ function where

$$\rho_a(\tau) = \frac{1}{n_p}\text{size}\{p : r_{p,a} \leq \tau\},$$

which can be thought of as a probability for the a-th algorithm to have performance ratio within factor $\tau \in \mathbb{R}_+$. The higher the curve $\rho_a(\tau)$ is located, the better the corresponding algorithm works. $\rho_a(1)$ is the number of problems on which the a-th algorithm showed the best performance.

As expected, tensorGP performs better than FITC, as it uses all the information contained in the training sample. The introduced prior distribution is more suited for



(A) Approximations accuracy. (B) Run-times.

FIGURE 2.4: Dolan-Moré curves for tensorGP, tensorGP-reg and FITC algorithms in logarithmic scale. The higher the curve lies, the better the corresponding algorithm performs.

anisotropic data and hence, GP with such prior (tensorGP-reg) performs even better (see Figure 2.4a).

To compare the run-time performances of the algorithms, we plotted Dolan-Mor'e curves where instead of approximation error, the training time was used (see Figure 2.4b). Here, we see that tensorGP and tensorGP-reg outperform FITC algorithm.

2.3.1 Rotating disc problem

In this section, we will consider a real-world problem of rotating disc shape design. Such kind of problems often arise during aircraft engine design and in turbomachinery (Armand, 1995).

In this problem, a disc of an impeller is considered. It is rotated around the shaft. The geometrical shape of the disc considered here is parameterized by 6 variables $\mathbf{x} = (h_1, h_2, h_3, h_4, r_2, r_3)$ (r_1 and r_4 are fixed), see Figures 2.5 and 2.6. The task of an engineer is to find such geometrical shape of the disc that minimizes the disc's weight and the contact pressure p_1 between the disc and the shaft while constraining the maximum radial stress Sr_{max} to be less than some threshold. The physical model of a rotating disc is described in (Armand, 1995) and it was adopted to the disc shape presented in Figures 2.5, 2.6 in order to calculate the contact pressure p_1 and the maximum radial stress Sr_{max}.

It is a common practice to build approximations of objective functions in order to analyze them and perform optimization (Forrester et al., 2008). So, we applied the tensorGP-reg algorithm and FITC developed in this work to this problem. The DoE was full factorial; the number of points in each dimension was $[1, 8, 8, 3, 15, 5]$, i.e., x_1 was fixed. The

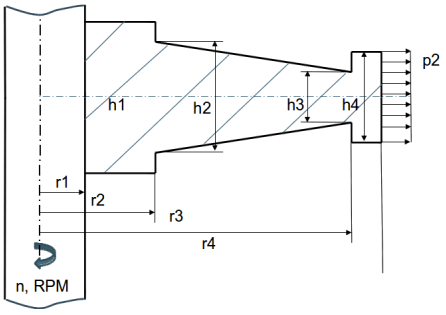


FIGURE 2.5: Rotating disc parametrization.

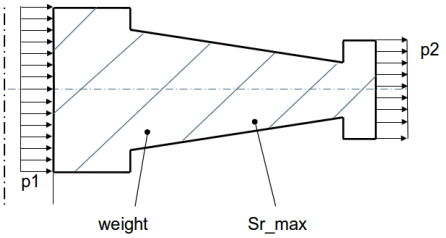
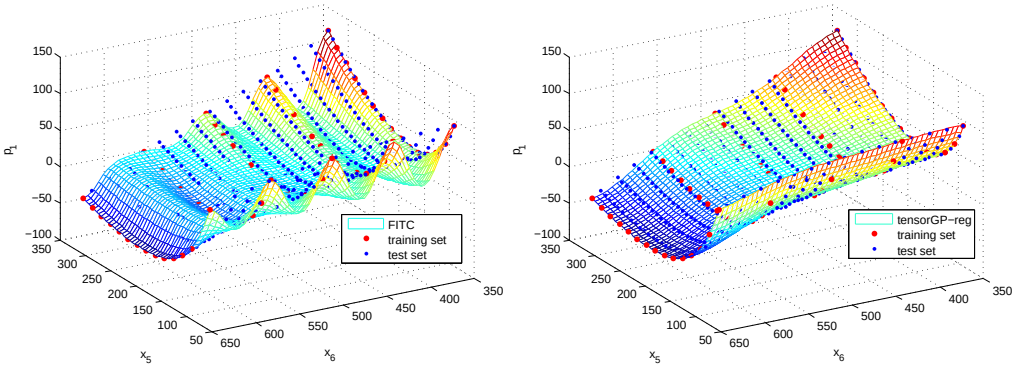


FIGURE 2.6: Rotating disc objectives.



(A) FITC. (B) The proposed approach.

FIGURE 2.7: 2D slice along x_5 and x_6 variables (other variables are fixed) of tensorGP-reg approximation.

number of points in factors differs significantly and the generated data set is anisotropic. The overall number of points in the training sample was 14 400.

Figures 2.7a and 2.7b depict 2D slices of contact pressure approximations along x_5, x_6 variables. As you can see, FITC model degenerates while tensorGP-reg provides a smooth and accurate approximation.

2.4 Incomplete Factorial Design of Experiments

A subset $\widetilde{\mathbf{\Omega}} \subseteq \mathbf{\Omega}$ of size $\widetilde{N}$ of the full factorial DoE is referred to as *incomplete factorial design of experiments*. In general, taking a subset of Ω breaks the structure of the dataset,

so we cannot apply the same techniques described in the previous section. However, the partial structure that is preserved can be of use, though at a higher computational cost. We use the same notation for the training set in this section $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\widetilde{N}}$.

2.4.1 Log-likelihood and posterior distribution calculation

To calculate the log-likelihood and the posterior mean, we need to calculate $\mathbf{K}_y^{-1}\mathbf{y}$ and the determinant $|\mathbf{K}_y|$.

Let $\mathbf{W}$ be a diagonal matrix of size $N \times N$, where N is the size of the full factorial DoE $\widetilde{\mathbf{\Omega}} \supseteq \mathbf{\Omega}$. Denote $R = N - \widetilde{N}$. Let $\mathbf{X} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbf{\Omega}\}_{i=1}^{N}$. Then

$$\mathbf{W}_{ii} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in \mathbf{\Omega}, \\ 0, & \text{if } \mathbf{x}_i \notin \mathbf{\Omega}. \end{cases}$$

We construct the vector $\widetilde{\mathbf{y}}$ as follows: if $\mathbf{x}_i \in \widetilde{\mathbf{\Omega}}$, then $\widetilde{\mathbf{y}}_i$ is filled with elements from $\mathbf{y}$ corresponding to $\mathbf{x}_i$. We fill the rest of the elements of $\widetilde{\mathbf{y}}$ arbitrarily. Now the matrix $\widetilde{\mathbf{K}}$ is the covariance matrix between points from the set $\mathbf{\Omega}$. Let us consider the following system of equations

$$\left(\mathbf{W}\widetilde{\mathbf{K}} + \sigma_{noise}^2\mathbf{I}\right)\mathbf{z} = \mathbf{W}\widetilde{\mathbf{y}}. \tag{2.17}$$

Notice that equations that correspond to missing values have the form $\sigma_{noise}^2\mathbf{z}_j = 0$. Therefore, the elements of $\mathbf{z}$ that correspond to missing values are equal to zero. The other elements are solutions of the equation $\mathbf{K}_y\mathbf{z}' = \mathbf{y}$, which gives us $\mathbf{K}_y^{-1}\mathbf{y}$. To solve (2.17), we use conjugate gradients (CG) approach. For CG, the matrix should be symmetric, so let's multiply the left hand side and the right hand side by $\widetilde{\mathbf{K}}$:

$$\left(\widetilde{\mathbf{K}}\mathbf{W}\widetilde{\mathbf{K}} + \sigma_{noise}^2\widetilde{\mathbf{K}}\right)\mathbf{z} = \widetilde{\mathbf{K}}\mathbf{W}\widetilde{\mathbf{y}}.$$

By substituting $\widetilde{\mathbf{z}} = \sqrt{\widetilde{\mathbf{K}}}\mathbf{z}$ we obtain a system with lower condition number:

$$\left(\sqrt{\widetilde{\mathbf{K}}}\mathbf{W}\sqrt{\widetilde{\mathbf{K}}} + \sigma_{noise}^2\mathbf{I}\right)\widetilde{\mathbf{z}} = \sqrt{\widetilde{\mathbf{K}}}\mathbf{W}\widetilde{\mathbf{y}},$$

where $\sqrt{\widetilde{\mathbf{K}}} = \mathbf{U}\sqrt{\mathbf{D}}\mathbf{U}^T$, $\mathbf{U}$ — is a matrix of eigenvectors of $\widetilde{\mathbf{K}}$, $\mathbf{D}$ — is a diagonal matrix with eigenvalues of $\widetilde{\mathbf{K}}$. Note that fast matrix-vector products can be calculated using the results of the section 2.2.1.

It is well known that CG converges at most in r iterations, where r is a number of different eigenvalues of the system matrix Nocedal & Wright (2006). In the system that we obtained, we can do such changes of variables, so that the number of different eigenvalues is equal to either $R + 1$ or $\widetilde{N} + 1$:

1. Change of variables 1:

$$\mathbf{z}_1 = \sqrt{\widehat{\mathbf{D}}}\mathbf{U}^T\widetilde{\mathbf{z}},$$

where $\widehat{\mathbf{D}} = \mathbf{D} + \sigma^2_{noise}\mathbf{I}$. The system of equations in this case looks like

$$\left(\sqrt{\widehat{\mathbf{D}}^{-1}\mathbf{D}}\mathbf{U}^T(\mathbf{W}-\mathbf{I})\mathbf{U}\sqrt{\mathbf{D}\widehat{\mathbf{D}}^{-1}} + \mathbf{I}\right)\mathbf{z}_1 = \sqrt{\widehat{\mathbf{D}}^{-1}\mathbf{D}}\mathbf{U}^T\mathbf{W}\widetilde{\mathbf{y}}. \tag{2.18}$$

2. Change of variables 2:

$$\mathbf{z}_2 = \mathbf{U}^T\widetilde{\mathbf{z}}.$$

In this case we obtain,

$$\left(\sqrt{\mathbf{D}}\mathbf{U}^T\mathbf{W}\mathbf{U}\sqrt{\mathbf{D}} + \sigma^2_{noise}\mathbf{I}\right)\mathbf{z}_2 = \sqrt{\mathbf{D}}\mathbf{U}^T\mathbf{W}\widetilde{\mathbf{y}}. \tag{2.19}$$

Proposition 2.3. *The matrix of the system of equations* (2.18) *has no more than* $R+1$ *different eigenvalues, while the matrix* (2.19) *has no more than* $\widetilde{N}+1$.

Proof. Let us show the proof for (2.19). The statement for (2.18) can be proved similarly. The rank of matrix $\mathbf{W}$ is equal to $\widetilde{N}$. As the rank of the product of matrices is not greater than the rank of the factors, the rank of the matrix $\mathbf{A} = \sqrt{\mathbf{D}}\mathbf{U}^T\mathbf{W}\mathbf{U}\sqrt{\mathbf{D}}$ is not greater than $\widetilde{N}$. Now, the matrix $\mathbf{A}$ is symmetric and has N real eigenvalues $\lambda_i, i = 1, \ldots, N$. Moreover, as the rank is not greater than $\widetilde{N}$, the number of different eigenvalues is not greater than $\widetilde{N}$ either. The eigenvalues of the matrix in (2.19) are equal to $\lambda_i + \sigma^2_{noise}$. Hence, the matrix has no more than $\widetilde{N}+1$ different eigenvalues. □

The computational complexity of one iteration of CG is the same as the complexity of computing $\widetilde{\mathbf{K}}_y\mathbf{x}$, therefore, the overall complexity of computing $\mathbf{K}_y^{-1}\mathbf{y}$ is

$$\mathcal{O}\left(\min\{R+1, \widetilde{N}+1\}N\sum_{i=1}^{K} n_i\right).$$

Assuming that $n_i \ll N$ (i.e., the number of factors is large, and their sizes are equal), we obtain $\mathcal{O}(\min\{R+1, \widetilde{N}+1\}N^{1+\frac{1}{K}})$. So, if less than half of the points are missing, the complexity is linear in the number of missing points. $\mathcal{O}((R+1)N^{1+\frac{1}{K}})$. In the limit, when there are no missing points, the complexity is the same as the complexity for full factorial DoE.

2.4.2 Calculation of the determinant

Let $\mathbf{A}$ be a covariance matrix of size $N \times R$ between training points and missing points, and $\mathbf{B}$ be a covariance matrix of size $R \times R$ between missing points. Then matrix $\widetilde{\mathbf{K}}_y$ can be represented as a block matrix

$$\widetilde{\mathbf{K}}_{\mathbf{y}} = \begin{pmatrix} \mathbf{K}_y & \mathbf{A} \\ \mathbf{A}^T & \mathbf{B} \end{pmatrix}.$$

The determinant of the block matrix can be calculated as follows $|\widetilde{\mathbf{K}}_y| = |\mathbf{K}_y||\mathbf{B} - \mathbf{A}^T\mathbf{K}_y^{-1}\mathbf{A}|$, therefore, the determinant of interest is given by

$$|\mathbf{K}_y| = \frac{|\widetilde{\mathbf{K}}_y|}{|\mathbf{B} - \mathbf{A}^T\mathbf{K}_y^{-1}\mathbf{A}|}. \tag{2.20}$$

The computational complexity of this expression is $\mathcal{O}(\min\{R+1, \widetilde{N}+1\}RN\sum_{i=1}^{K} n_i)$, because we need to calculate matrix-vector multiplication $\mathbf{K}_y^{-1}\mathbf{A}_i$ R times, where $\mathbf{A}_i$ is an i-th column of matrix $\mathbf{A}$. The memory complexity equals $\mathcal{O}(R\widetilde{N} + N + \sum_{i=1}^{K} n_i^2)$.

However, we can reduce memory to $\mathcal{O}(\widetilde{N})$ and improve the constant in the complexity of (2.20). For this, notice that the approach to calculate $\mathbf{K}_y^{-1}\mathbf{y}$ also allows to calculate $\mathbf{C}^{-1}\mathbf{v}$, where $\mathbf{C}$ is an arbitrary principal submatrix of size $m \times m$ of the matrix $\widetilde{\mathbf{K}}_y$, and $\mathbf{v}$ is any vector of length m. The computational complexity of the operation in this case is $\mathcal{O}\left(\min\{m+1, \widetilde{N}-m+1\}\widetilde{N}\sum_{i=1}^{K} n_i\right)$.

Denote

$$\widetilde{\mathbf{K}}_y = \begin{pmatrix} \mathbf{K}_1 & \mathbf{a}_1 \\ \mathbf{a}_1^T & b_1 \end{pmatrix},$$

where $\mathbf{K}_1$ is of size $(\widetilde{N}-1) \times (\widetilde{N}-1)$, $\mathbf{a}_1$ is a vector of length $\widetilde{N}-1$, b_1 is a scalar. Then

$$|\widetilde{\mathbf{K}}_y| = |\mathbf{K}_1|(b_1 - \mathbf{a}_1^T\mathbf{K}_1^{-1}\mathbf{a}_1).$$

Similarly, splitting $\mathbf{K}_1$ into blocks $\begin{pmatrix} \mathbf{K}_2 & \mathbf{a}_2 \\ \mathbf{a}_2 & b_2 \end{pmatrix}$, and splitting matrix $\mathbf{K}_2$ into $\begin{pmatrix} \mathbf{K}_3 & \mathbf{a}_3 \\ \mathbf{a}_3 & b_3 \end{pmatrix}$, and so on, we obtain

$$|\widetilde{\mathbf{K}}_y| = |\mathbf{K}_y| \prod_{i=1}^{R} (b_i - \mathbf{a}_i^T\mathbf{K}_i^{-1}\mathbf{a}_i),$$

or

$$|\mathbf{K}_y| = \frac{|\widetilde{\mathbf{K}}_y|}{\prod_{i=1}^{R}(b_i - \mathbf{a}_i^T\mathbf{K}_i^{-1}\mathbf{a}_i)}. \tag{2.21}$$

Here, to calculate the right hand side, we need to store only $\mathbf{K}_i^{-1}\mathbf{a}_i$, therefore, the total memory complexity is $\mathcal{O}(N + \sum_{i=1}^{K} n_i^2)$.

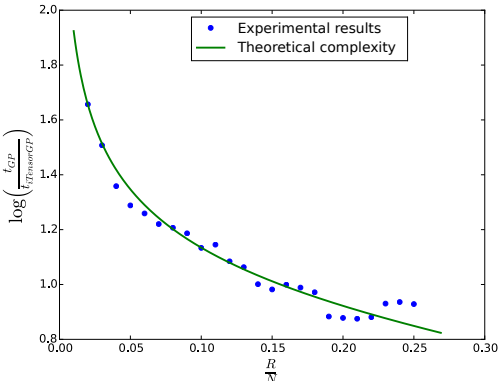


FIGURE 2.8: Comparison of the training time of the proposed approach (iTensorGP) and standard approach (GP).

The complexity of the calculation of each $\mathbf{K}_i^{-1}\mathbf{a}_i$ is lower than the complexity of the calculation of $\mathbf{K}_y^{-1}\mathbf{A}_i$, thus, the expression (2.21) is more efficient, although asymptotically, it is the same.

2.4.3 Experiments

To compare the classical approach for GP regression against the proposed approach, we generated several datasets with incomplete factorial DoE. The dimensionality of input is 5, factor sizes are equal to $[5, 4, 4, 4, 4]$, the number of missing points varies from 100 to 320. In Figure 2.8, you can see the ratio of the computation time of the standard approach (GP) and the training time of the proposed approach (iTensorGP) for different number of missing values. Theory shows that the ratio should be $\frac{(N-R)^3}{R^2 N(\sum_{i=1}^{K} n_i)}$. Then the number of missing points increases, and so does the training time of iTensorGP. At some points, iTensorGP becomes less efficient than the standard GP approach. However, in case of a small amount of missing points, iTensorGP is much faster.

Chapter 3

Large-Scale Kernel Methods for Unstructured Datasets

In the vast majority of real-world problems data sets do not have the factorial structure. In this case, there are several approaches to scale up the GP model. One is to aggregate several smaller models into a big one, e.g., Mixture GPs or Bayesian machine committee (Rasmussen & Williams, 2006; Rasmussen & Ghahramani, 2001). The idea here is to split data set into several small subsets $\mathcal{D}_i, i = 1, \ldots, M$, build on each data set GP model and then combine them. For instance, in the Bayesian committee machine, the final distribution is given by

$$p(y^*|\mathcal{D}) \sim \frac{\prod_{i=1}^{M} p(y^*|\mathcal{D}_i)}{p(y^*)},$$

assuming that the correlation between the subsets is small.

Another class of methods builds a lightweight approximation of the kernel function. There are also data-dependent approaches that follow Nyström's method for kernel approximation (Rasmussen & Williams, 2006). The idea of approximation is based on the numerical approximation of the *eigenfunctions* and *eigenvalues* of the kernel function. A function $\phi(\cdot)$ that satisfies the following equation

$$\int k(\mathbf{x}, \mathbf{x}')\phi(\mathbf{x})d\mu(\mathbf{x}) = \lambda\phi(\mathbf{x}')$$

is called the eigenfunction of kernel k, where λ is a corresponding eigenvalue with respect to measure μ. If we let consider $d\mu(\mathbf{x}) = p(\mathbf{x})d\mathbf{x}$, we can write the numerical

approximation to the equation

$$\lambda_i \phi_i(\mathbf{x}) \simeq \frac{1}{M} \sum_{k=1}^{M} k(\mathbf{x}_k, \mathbf{x}) \phi_i(\mathbf{x}_k), \tag{3.1}$$

where points $\mathbf{x}_k$, generated from distribution $p(\mathbf{x})$, are called *inducing points*. By plugging $\mathbf{x}_i$ from the training set into equation above we obtain

$$\mathbf{K}_f \mathbf{u}_i = \hat{\lambda}_i \mathbf{u}_i, \quad \phi_i(\mathbf{x}_j) = \sqrt{\hat{\lambda}_i}\,(\mathbf{u}_i)_j\,.$$

Then from (3.1), we obtain the *Nyström* approximation of the i-th eigenfunction $\phi(\mathbf{x}) = \frac{M}{\hat{\lambda}_i} \mathbf{k}^\top \mathbf{u}_i$, and, therefore, the approximation of the kernel matrix:

$$\mathbf{K}_f \simeq \widehat{\mathbf{K}}_f = \mathbf{K}_{NM} \mathbf{K}_{MM}^{-1} \mathbf{K}_{MN}, \tag{3.2}$$

where $\mathbf{K}_{MN}$ is a covariance matrix between training points and inducing points while $\mathbf{K}_{MM}$ is a covariance matrix between inducing points. More details can be found in (Rasmussen & Williams, 2006).

In (Williams & Seeger, 2001), the authors first proposed to use the (3.2) approximation in the posterior distribution of the GP regression model. The computational complexity is reduced to $\mathcal{O}(NM^2 + M^3)$ operations, which is beneficial if $M \ll N$. There are several works that build on the Nyström approximation, trying to refine the approximation of the covariance matrix and thus improve the quality of the model while preserving low computational complexity (Quiñonero-Candela & Rasmussen, 2005; Rossi et al., 2020).

In this thesis, we consider different types of approximations based on so-called random features. The idea of the approach is to construct an approximation of the kernel function that enables a low-rank approximation of the kernel matrix. To construct such an approximation, we need to consider the kernel function from a different perspective. It turns out that every positive definite kernel k uniquely defines some space of functions and vice versa. Moreover, it can be seen as an inner product in an appropriate Hilbert space.

Definition 3.1 (Positive definite kernels)**.** Let $\mathcal{X}$ be a nonempty set. A symmetric function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel, if for any set $\mathbf{x}_1, \ldots, \mathbf{x}_N$, $\forall N \in \mathbb{N}$ and for any $(c_1, \ldots, c_N) \subset \mathbb{R}$ it holds

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

Note that the covariance functions that we use in GP are also positive definite kernels.

Definition 3.2 (RKHS, (Aronszajn, 1950))**.** Let $\mathcal{X}$ be a nonempty set and k be a positive definite kernel on $\mathcal{X}$. A Hilbert space $\mathcal{H}$ of function on $\mathcal{X}$ with an inner-product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is called a reproducing kernel Hilbert space (RKHS) with reproducing kernel k, if the following is satisfied:

1. For all $\mathbf{x} \in \mathcal{X}$ we have $k(\cdot, \mathbf{x}) \in \mathcal{H}$;
2. for all $\mathbf{x} \in \mathcal{X}$ and for all $f \in \mathcal{H}$

$$f(\mathbf{x}) = \langle f, k(\cdot, \mathbf{x}) \rangle. \qquad \text{Reproducing property}$$

In the reproducing property, we used $k(\cdot, \mathbf{x})$, which is actually a real-valued function such that $\mathbf{y} \to k(\mathbf{y}, \mathbf{x})$ for $\forall \mathbf{y} \in \mathcal{X}$. So, the kernel function satisfies

$$k(\mathbf{x}, \mathbf{y}) = \langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{y}) \rangle, \quad \mathbf{x}, \mathbf{y} \in \mathcal{X}.$$

where $k(\cdot, \mathbf{x})$ is the *canonical feature map* of $\mathbf{x}$. There can be a lot of different features maps $\psi \colon \mathcal{X} \to \mathcal{H}$, such that their inner product is defined by the kernel function $k(\mathbf{x}, \mathbf{y}) = \langle \psi(\mathbf{x}), \psi(\mathbf{y}) \rangle$.

The idea for kernel approximation using randomized feature maps is based on finding finite-dimensional maps that approximate the inner product associated with the kernel function.

The main contributions of this chapter are as follows:

- Introducing a technique for kernel approximation based on randomized feature maps and special quadrature rules.
- Deriving error bounds for the developed approach.
- Showing that (Rahimi & Recht, 2008; Felix et al., 2016) are special cases of the proposed method.

3.1 Quadrature-based Features for Kernel Approximation

3.1.1 Random Fourier Features

One of the most well-known approach is called Random Fourier Features (RFF). It was first proposed by (Rahimi & Recht, 2008), and it is based on Bochner's theorem.

Theorem 3.3 (Bochner). *A continuous kernel* $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{r}), \mathbf{r} = \mathbf{x} - \mathbf{x}'$ *on* $\mathbb{R}^d$ *is positive definite if and only if* $k(\mathbf{r})$ *is a Fourier transform of a non-negative measure* $p(\mathbf{w})$

$$k(\mathbf{r}) = \int_{\Omega} p(\mathbf{w}) e^{j\mathbf{w}^\top(\mathbf{x}-\mathbf{x}')} d\mathbf{w}. \tag{3.3}$$

By applying Monte-Carlo sampling to approximate integral in (3.3) RFF introduces a low-dimensional randomized approximation to feature maps:

$$k(\mathbf{x}, \mathbf{y}) \approx \hat{\mathbf{\Psi}}(\mathbf{x})^\top \hat{\mathbf{\Psi}}(\mathbf{y}), \tag{3.4}$$

$$\hat{\mathbf{\Psi}}(\mathbf{x}) = \frac{1}{\sqrt{D}} \begin{bmatrix} \cos(\mathbf{w}_1^\top \mathbf{x}) \\ \sin(\mathbf{w}_1^\top \mathbf{x}) \\ \cdots \\ \cos(\mathbf{w}_D^\top \mathbf{x}) \\ \sin(\mathbf{w}_D^\top \mathbf{x}) \end{bmatrix}, \quad \mathbf{w}_i \sim p(\mathbf{w}),$$

where D is a number of generated samples. Exploiting the idea with an integral representation of the dot product

$$k(\mathbf{x}, \mathbf{y}) = \int_{\Omega} \psi(\mathbf{w}, \mathbf{x}) \psi(\mathbf{w}, \mathbf{y}) p(\mathbf{w}) d\mathbf{w},$$

we can construct low-rank approximations to a wider class of kernel functions (not only shift-invariant kernels). In this case the feature map $\hat{\mathbf{\Psi}}(\mathbf{x})$ looks like

$$\hat{\mathbf{\Psi}}(\mathbf{x}) = \frac{1}{\sqrt{D}} \begin{bmatrix} \psi(\mathbf{w}_1, \mathbf{x}) \\ \cdots \\ \psi(\mathbf{w}_D, \mathbf{x}) \end{bmatrix}, \quad \mathbf{w} \sim p(\mathbf{w}).$$

A randomized D-dimensional mapping $\hat{\mathbf{\Psi}}(\cdot)$ applied to the original data input allows employing standard linear methods, i.e. reverting the kernel trick. In doing so one reduces the complexity to that of linear methods, e.g. D-dimensional approximation admits $\mathcal{O}(ND^2)$ training time, $\mathcal{O}(ND)$ memory and $\mathcal{O}(N)$ prediction time.

It is well known from the theory on Monte Carlo based estimates that as $D \to \infty$, the randomized feature maps based approximations converge to the exact kernel $k(\mathbf{x}, \mathbf{y})$. Recent research (Yang et al., 2014; Felix et al., 2016; Choromanski & Sindhwani, 2016) aims to improve the convergence of approximation so that a smaller D can be used to obtain the same quality of approximation.

Here we will consider the class of kernels admitting the following integral representation

$$k(\mathbf{x},\mathbf{y}) = \mathbb{E}_{p(\mathbf{w})} f_{\mathbf{xy}}(\mathbf{w}) = I(f_{\mathbf{xy}}), \quad p(\mathbf{w}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{\|\mathbf{w}\|^2}{2}}, \\ f_{\mathbf{xy}} = \phi(\mathbf{w}^\top \mathbf{x})\phi(\mathbf{w}^\top \mathbf{y}). \tag{3.5}$$

It includes the class of shift-invariant kernels, e.g. the popular Gaussian kernel with $f_{\mathbf{xy}}(\mathbf{w}) = \phi(\mathbf{w}^\top \mathbf{x})^\top \phi(\mathbf{w}^\top \mathbf{y})$, where $\phi(\cdot) = \begin{bmatrix}\cos(\cdot) & \sin(\cdot)\end{bmatrix}^\top$. It also contains Pointwise Nonlinear Gaussian (PNG) kernels. They are widely used in practice and have interesting connections with neural networks (Cho & Saul, 2009; Williams, 1997).

The main challenge for the construction of low-dimensional feature maps is the approximation of the expectation in (3.5) which is d-dimensional integral with Gaussian weight. To improve the approximation we introduce quarature rules to approximate the integral and then show that they generalize several prominent papers in this topic.

3.2 Quadrature Rules

We start with rewriting the expectation in Equation (3.5) as integral of $f_{\mathbf{xy}}$ with respect to $p(\mathbf{w})$:

$$I(f_{\mathbf{xy}}) = (2\pi)^{-\frac{d}{2}} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2}} f_{\mathbf{xy}}(\mathbf{w}) d\mathbf{w}.$$

Integration can be performed by means of quadrature rules. The rules usually take a form of interpolating function that is easy to integrate. Given such a rule, one may sample points from the domain of integration and calculate the value of the rule at these points. Then, the sample average of the rule values would yield the approximation of the integral.

The connection between integral approximation and mapping ψ is straightforward. In what follows we show a brief derivation of the quadrature rules that allow for an explicit mapping of the form: $\psi(\mathbf{x}) = [\ a_0\phi(0)\ \ a_1\phi(\mathbf{w}_1^\top \mathbf{x})\ \ \ldots\ \ a_D\phi(\mathbf{w}_D^\top \mathbf{x})\ ]$, where the choice of the weights a_i and the points $\mathbf{w}_i$ is dictated by the quadrature.

We use the average of sampled quadrature rules developed by (Genz & Monahan, 1998) to yield unbiased estimates of $I(f_{\mathbf{xy}})$. A change of coordinates is the first step to facilitate stochastic spherical-radial rules. Now, let $\mathbf{w} = r\mathbf{z}$, with $\mathbf{z}^\top \mathbf{z} = 1$, so that $\mathbf{w}^\top \mathbf{w} = r^2$ for $r \in [0, \infty]$, leaving us with (to ease the notation we substitute $f_{\mathbf{xy}}$ with f)

$$I(f) = (2\pi)^{-\frac{d}{2}} \int_{U_d} \int_0^{\infty} e^{-\frac{r^2}{2}} r^{d-1} f(r\mathbf{z}) dr d\mathbf{z} = \frac{(2\pi)^{-\frac{d}{2}}}{2} \int_{U_d} \int_{-\infty}^{\infty} e^{-\frac{r^2}{2}} |r|^{d-1} f(r\mathbf{z}) dr d\mathbf{z}, \tag{3.6}$$

$I(f)$ is now a double integral over the unit d-sphere $U_d = \{\mathbf{z} : \mathbf{z}^\top \mathbf{z} = 1, \mathbf{z} \in \mathbb{R}^d\}$ and over the radius. To account for both integration regions we apply a combination of spherical (S) and radial (R) rules known as spherical-radial (SR) rules. To provide an intuition how the rules work, here we briefly state and discuss their form.

Stochastic radial rules of degree $2l+1$ $R(h) = \sum_{i=0}^{l} \hat{w}_i \frac{h(\rho_i)+h(-\rho_i)}{2}$ have the form of the weighted symmetric sums and approximate the infinite range integral $T(h) = \int_{-\infty}^{\infty} e^{-\frac{r^2}{2}} |r|^{d-1} h(r) dr$. Note that when h is set to the function f of interest, $T(f)$ corresponds to the inner integral in (3.6). To get an unbiased estimate for $T(h)$, points ρ_i are sampled from specific distributions. The weights $\hat{w}_i$ are derived so that the rule is exact for polynomials of degree $2l+1$ and give unbiased estimate for other functions.

Stochastic spherical rules $S_{\mathbf{Q}}(s) = \sum_{j=1}^{p} \widetilde{w}_j s(\mathbf{Q}\mathbf{z}_j)$, where $\mathbf{Q}$ is a random orthogonal matrix, approximate an integral of a function $s(\mathbf{z})$ over the surface of unit d-sphere U_d, where $\mathbf{z}_j$ are points on U_d, i.e. $\mathbf{z}_j^\top \mathbf{z}_j = 1$. Remember that the outer integral in (3.6) has U_d as its integration region. The weights $\widetilde{w}_j$ are stochastic with distribution such that the rule is exact for polynomials of degree p and gives unbiased estimate for other functions.

Stochastic spherical-radial rules SR of degree $(2l+1, p)$ are given by the following expression[1]

$$SR_{\mathbf{Q},\rho}^{(2l+2,p)} = \sum_{j=1}^{p} \widetilde{w}_j \sum_{i=1}^{l} \hat{w}_i \frac{f(\rho\mathbf{Q}\mathbf{z}_i) + f(-\rho\mathbf{Q}\mathbf{z}_i)}{2},$$

where the distributions of weights are such that if degrees of radial rules and spherical rules coincide, i.e. $2l+1 = p$, then the rule is exact for polynomials of degree $2l+1$ and gives unbiased estimate of the integral for other functions.

3.2.1 Spherical-radial rules of degree (1, 1)

Proposition 3.4. *Random Fourier Features for RBF kernel are SR rules of degree* $(1, 1)$.

Proof. If we take radial rule of degree 1 and spherical rule of degree 1, we obtain the following rule $SR_{\mathbf{Q},\rho}^{(1,1)} = \frac{f(\rho\mathbf{Q}\mathbf{z})+f(-\rho\mathbf{Q}\mathbf{z})}{2}$, where $\rho \sim \chi(d)$. It is easy to see that $\rho\mathbf{Q}\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, and for shift invariant kernel $f(\mathbf{w}) = f(-\mathbf{w})$, thus, the rule reduces to $SR_{\mathbf{Q},\rho}^{(1,1)} = f(\mathbf{w})$, where $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$. Now, RFF (Rahimi & Recht, 2008) makes approximation of the RBF kernel in exactly the same way: it generates random vector from Gaussian distribution and calculates the corresponding feature map. □

[1] Please see (Genz & Monahan, 1998) for detailed derivation of SR rules.

3.2.2 Spherical-radial rules of degree (1, 3)

Proposition 3.5. *Orthogonal Random Features for RBF kernel are SR rules of degree* $(1, 3)$.

Proof. Let us take radial rule of degree 1 and spherical rule of degree 3. In this case we get the following spherical-radial rule $SR^{1,3}_{\mathbf{Q},\rho} = \sum_{i=1}^{d} \frac{f(\rho\mathbf{Q}\mathbf{e}_i)+f(-\rho\mathbf{Q}\mathbf{e}_i)}{2}$, where $\rho \sim \chi(d)$, $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)^\top$ is an i-th column of the identity matrix.

Let us compare $\mathrm{SR}^{1,3}$ rules with Orthogonal Random Features (Felix et al., 2016) for the RBF kernel. In the ORF approach, the weight matrix $\mathbf{W} = \mathbf{S}\mathbf{Q}$ is generated, where $\mathbf{S}$ is a diagonal matrix with the entries drawn independently from $\chi(d)$ distribution and $\mathbf{Q}$ is a random orthogonal matrix. The approximation of the kernel is then given by $k_{\mathrm{ORF}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} f(\mathbf{w}_i)$, where $\mathbf{w}_i$ is the i-th row of the matrix $\mathbf{W}$. As the rows of $\mathbf{Q}$ are orthonormal, they can be represented as $\mathbf{Q}\mathbf{e}_i$. □

3.2.3 Spherical-radial rules of degree (3, 3)

We go further and take both spherical and radial rules of degree 3, where we use original and reflected vertices $\mathbf{v}_j$ of randomly rotated unit vertex regular d-simplex $\mathbf{V}$ as the points on the unit sphere

$$SR^{3,3}_{\mathbf{Q},\rho}(f) = \left(1 - \frac{d}{\rho^2}\right) f(\mathbf{0}) + \frac{d}{d+1} \sum_{j=1}^{d+1} \left[\frac{f(-\rho\mathbf{Q}\mathbf{v}_j) + f(\rho\mathbf{Q}\mathbf{v}_j)}{2\rho^2}\right], \tag{3.7}$$

where $\rho \sim \chi(d+2)$. We apply (3.7) to the approximation of (3.6) by averaging the samples of $SR^{3,3}_{\mathbf{Q},\rho}$:

$$I(f) = \mathbb{E}_{\mathbf{Q},\rho}[SR^{3,3}_{\mathbf{Q},\rho}(f)] \approx \hat{I}(f) = \frac{1}{n} \sum_{i=1}^{n} SR^{3,3}_{\mathbf{Q}_i,\rho_i}(f), \tag{3.8}$$

where n is the number of sampled SR rules. Speaking in terms of the approximate feature maps, the new feature dimension D in case of the quadrature based approximation equals $2n(d+1)+1$ as we sample n rules and evaluate each of them at $2(d+1)$ random points and 1 zero point.

We propose to modify the quadrature rule by generating $\rho_j \sim \chi(d+2)$ for each $\mathbf{v}_j$, i.e. $SR^{3,3}_{\mathbf{Q},\rho}(f) = \left(1 - \sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2}\right) f(\mathbf{0}) + \frac{d}{d+1} \sum_{j=1}^{d+1} \left[\frac{f(-\rho_j\mathbf{Q}\mathbf{v}_j)+f(\rho_j\mathbf{Q}\mathbf{v}_j)}{2\rho_j^2}\right]$. It doesn't affect the quality of approximation while simplifies an analysis of the quadrature-based random features.

Explicit mapping We finally arrive at the map $\psi(\mathbf{x}) = \begin{bmatrix} a_0\phi(0) & a_1\phi(\mathbf{w}_1^\top \mathbf{x}) & \dots & a_D\phi(\mathbf{w}_D^\top \mathbf{x}) \end{bmatrix}$, where $a_0 = \sqrt{1 - \sum_{d+1}^{j=1} \frac{d}{\rho^2}}$ [2], $a_j = \frac{1}{\rho_j}\sqrt{\frac{d}{2(d+1)}}$, $\mathbf{w}_j$ is the j-th row in the matrix $\mathbf{W} = \boldsymbol{\rho} \otimes \begin{bmatrix} (\mathbf{QV})^\top \\ -(\mathbf{QV})^\top \end{bmatrix}$, $\boldsymbol{\rho} = [\rho_1 \dots \rho_D]^\top$. To get D features one simply stacks $n = \frac{D}{2(d+1)+1}$ such matrices $\mathbf{W}^k = \boldsymbol{\rho}^k \begin{bmatrix} (\mathbf{Q}^k\mathbf{V})^\top \\ -(\mathbf{Q}^k\mathbf{V})^\top \end{bmatrix}$ so that $\mathbf{W} \in \mathbb{R}^{D\times d}$, where only $\mathbf{Q}^k \in \mathbb{R}^{d\times d}$ and $\boldsymbol{\rho}^k$ are generated randomly $(k = 1, \dots, n)$. For Gaussian kernel, $\phi(\cdot) = \begin{bmatrix} \cos(\cdot) & \sin(\cdot) \end{bmatrix}^\top$. For the 0-order arc-cosine kernel, $\phi(\cdot) = \Theta(\cdot)$, where $\Theta(\cdot)$ is the Heaviside function. For the 1-order arc-cosine kernel, $\phi(\cdot) = \max(0, \cdot)$.

3.2.4 Generating uniformly random orthogonal matrices

The SR rules require a random orthogonal matrix $\mathbf{Q}$. If $\mathbf{Q}$ follows Haar distribution, the averaged samples of $SR^{3,3}_{\mathbf{Q},\rho}$ rules provide an unbiased estimate for (3.6). Essentially, Haar distribution means that all orthogonal matrices in the group are equiprobable, i.e. uniformly random. Methods for sampling such matrices vary in their complexity of generation and multiplication.

We test two algorithms for obtaining $\mathbf{Q}$. The first uses a QR decomposition of a random matrix to obtain a product of a sequence of reflectors/rotators $\mathbf{Q} = \mathbf{H}_1 \dots \mathbf{H}_{n-1}\mathbf{D}$, where $\mathbf{H}_i$ is a random Householder/Givens matrix and a diagonal matrix $\mathbf{D}$ has entries such that $\mathbb{P}(d_{ii} = \pm 1) = \frac{1}{2}$. It implicates no fast matrix multiplication. We test both methods for random orthogonal matrix generation and, since their performance coincides, we leave this one out for cleaner figures in the Experiments section.

The other choice for $\mathbf{Q}$ are so-called butterfly matrices (Genz, 1998). For $d = 4$

$$\mathbf{B}^{(4)} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_3 & -s_3 \\ 0 & 0 & s_3 & c_3 \end{bmatrix} \begin{bmatrix} c_2 & 0 & -s_2 & 0 \\ 0 & c_2 & 0 & -s_2 \\ s_2 & 0 & c_2 & 0 \\ 0 & s_2 & 0 & c_2 \end{bmatrix} = \begin{bmatrix} c_1c_2 & -s_1c_2 & -c_1s_2 & s_1s_2 \\ s_1c_2 & c_1c_2 & -s_1s_2 & -c_1s_2 \\ c_3s_2 & -s_3s_2 & c_3c_2 & -s_3c_2 \\ s_3s_2 & c_3s_2 & s_3c_2 & c_3c_2 \end{bmatrix},$$

where s_i, c_i is sine and cosine of some angle θ_i, $i = 1, \dots, d-1$. For definition and discussion please see Appendix. The factors of $\mathbf{B}^{(d)}$ are structured and allow fast matrix multiplication. The method using butterfly matrices is denoted by $\mathbf{B}$ in the Experiments section.

[2] To get $a_0^2 \geq 0$, you need to sample ρ_j two times on average (see Appendix for details).

3.3 Error bounds

Proposition 3.6. *Let l be a diameter of the compact set $\mathcal{X}$ and $p(\mathbf{w}) = \mathcal{N}(0, \sigma_p^2 \mathbf{I})$ be the probability density corresponding to the kernel. Let us suppose that $|\phi(\mathbf{w}^\top \mathbf{x})| \leq \kappa$, $|\phi'(\mathbf{w}^\top \mathbf{x})| \leq \mu$ for all $\mathbf{w} \in \Omega$, $\mathbf{x} \in \mathcal{X}$ and $\left|\frac{1 - f_{\mathbf{xy}}(\rho \mathbf{z})}{\rho^2}\right| \leq M$ for all $\rho \in [0, \infty)$, where $\mathbf{z}^\top \mathbf{z} = 1$. Then for Quadrature-based Features approximation $\hat{k}(\mathbf{x}, \mathbf{y})$ of the kernel function $k(\mathbf{x}, \mathbf{y})$ and any $\varepsilon > 0$ it holds*

$$\mathbb{P}\left(\sup_{\mathbf{x},\mathbf{y} \in \mathcal{X}} |\hat{k}(\mathbf{x}, \mathbf{y}) - k(\mathbf{x}, \mathbf{y})| \geq \varepsilon\right) \leq \beta_d \left(\frac{\sigma_p l \kappa \mu}{\varepsilon}\right)^{\frac{2d}{d+1}} \exp\left(-\frac{D\varepsilon^2}{8M^2(d+1)}\right),$$

where $\beta_d = \left(d^{\frac{-d}{d+1}} + d^{\frac{1}{d+1}}\right) 2^{\frac{6d+1}{d+1}} \left(\frac{d}{d+1}\right)^{\frac{d}{d+1}}$. Thus we can construct approximation with error no more than ε with probability at least $1 - \delta$ as long as

$$D \geq \frac{8M^2(d+1)}{\varepsilon^2} \left[\frac{2}{1 + \frac{1}{d}} \log \frac{\sigma_p l \kappa \mu}{\varepsilon} + \log \frac{\beta_d}{\delta}\right].$$

The proof of this proposition closely follows (Sutherland & Schneider, 2015), details can be found in the Appendix.

Term β_d depends on dimension d, its maximum is $\beta_{86} \approx 64.7 < 65$, and $\lim_{d \to \infty} \beta_d = 64$, though it is lower for small d. Let us compare this probability bound with the similar result for RFF in (Sutherland & Schneider, 2015). Under the same conditions the required number of samples to achieve error no more than ε with probability at least $1 - \delta$ for RFF is the following

$$D \geq \frac{8(d+1)}{\varepsilon^2} \left[\frac{2}{1 + \frac{1}{d}} \log \frac{\sigma_p l}{\varepsilon} + \log \frac{\beta_d}{\delta} + \frac{d}{d+1} \log \frac{3d+3}{2d}\right].$$

For Quadrature-based Features for RBF kernel $M = \frac{1}{2}, \kappa = \mu = 1$, therefore, we obtain

$$D \geq \frac{2(d+1)}{\varepsilon^2} \left[\frac{2}{1 + \frac{1}{d}} \log \frac{\sigma_p l}{\varepsilon} + \log \frac{\beta_d}{\delta}\right].$$

The asymptotics is the same, however, the constants are smaller for our approach. See Section 3.4 for empirical justification of the obtained result.

Proposition 3.7 ((Sutherland & Schneider, 2015))**.** *Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, let $h(\mathbf{x})$ denote the result of kernel ridge regression using the positive semi-definite training kernel matrix $\mathbf{K}$, test kernel values $\mathbf{k}_\mathbf{x}$ and regularization parameter λ. Let $\hat{h}(\mathbf{x})$ be the same using a PSD approximation to the training kernel matrix $\widehat{\mathbf{K}}$ and test kernel values $\hat{\mathbf{k}}_\mathbf{x}$. Further, assume that the training labels are centered,*

$\sum_{i=1}^n y_i = 0$, *and let* $\sigma_y^2 = \frac{1}{n}\sum_{i=1}^n y_i^2$. *Also suppose* $\|\mathbf{k}_\mathbf{x}\|_\infty \leq \kappa$. *Then*

$$|\hat{h}(\mathbf{x}) - h(\mathbf{x})| \leq \frac{\sigma_y\sqrt{n}}{\lambda}\|\hat{\mathbf{k}}_\mathbf{x} - \mathbf{k}_\mathbf{x}\|_2 + \frac{\kappa\sigma_y n}{\lambda^2}\|\widehat{\mathbf{K}} - \mathbf{K}\|_2.$$

Suppose that $\sup|k(\mathbf{x},\mathbf{x}') - \hat{k}(\mathbf{x},\mathbf{x}')| \leq \varepsilon$ for all $\mathbf{x},\mathbf{x}' \in \mathbb{R}^d$. Then $\|\hat{\mathbf{k}}_\mathbf{x} - \mathbf{k}_\mathbf{x}\|_2 \leq \sqrt{n}\varepsilon$ and $\|\widehat{\mathbf{K}} - \mathbf{K}\|_2 \leq \|\widehat{\mathbf{K}} - \mathbf{K}\|_F \leq n\varepsilon$. By denoting $\lambda = n\lambda_0$ we obtain $|\hat{h}(\mathbf{x}) - h(\mathbf{x})| \leq \frac{\lambda_0+1}{\lambda_0^2}\sigma_y\varepsilon$. Therefore,

$$\mathbb{P}\left(\ |\hat{h}(\mathbf{x}) - h(\mathbf{x})| \geq \varepsilon\right) \leq \mathbb{P}\left(\|\hat{k}(\mathbf{x},\mathbf{x}') - k(\mathbf{x},\mathbf{x}')\|_\infty \geq \frac{\lambda_0^2\varepsilon}{\sigma_y(\lambda_0+1)}\right).$$

So, for the quadrature rules we can guarantee $|\hat{h}(\mathbf{x}) - h(\mathbf{x})| \leq \varepsilon$ with probability at least $1-\delta$ as long as

$$D \geq 8M^2(d+1)\sigma_y^2\left(\frac{\lambda_0+1}{\lambda_0^2\varepsilon}\right)^2\left[\frac{2}{1+\frac{1}{d}}\log\frac{\sigma_y\sigma_p l\kappa\mu(\lambda_0+1)}{\lambda_0^2\varepsilon} + \log\frac{\beta_d}{\delta}\right].$$

3.4 Experiments

We extensively study the proposed method on several established benchmarking datasets: Powerplant, LETTER, USPS, MNIST, CIFAR100 (Krizhevsky & Hinton, 2009), LEUKEMIA (Golub et al., 1999). We compare kernel approximation error across different kernels and number of features with several other approaches. We also estimate the quality of SVM models with approximate kernels on the same data sets.

3.4.1 Methods

We present a comparison of our method (**B**) with estimators based on a simple Monte Carlo, quasi-Monte Carlo (Yang et al., 2014) and Gaussian quadratures (Dao et al., 2017). The Monte Carlo approach has a variety of ways to generate samples: unstructured Gaussian (Rahimi & Recht, 2008), structured Gaussian (Felix et al., 2016), random orthogonal matrices (ROM) (Choromanski et al., 2017).

Monte Carlo integration (G, Gort, ROM). The kernel is estimated as $\hat{k}(\mathbf{x},\mathbf{y}) = \frac{1}{D}\phi(\mathbf{Mx})\phi(\mathbf{My})$, where $\mathbf{M} \in \mathbb{R}^{D\times d}$ is a random weight matrix. For unstructured Gaussian based approximation $\mathbf{M} = \mathbf{G}$, where $\mathbf{G}_{ij} \sim \mathcal{N}(0,1)$. Structured Gaussian has $\mathbf{M} = \mathbf{G}_{\text{ort}}$, where $\mathbf{G}_{\text{ort}} = \mathbf{DQ}$, $\mathbf{Q}$ is obtained from QR decomposition of $\mathbf{G}$, $\mathbf{D}$ is a diagonal matrix with diagonal elements sampled from the $\chi(d)$ distribution. In

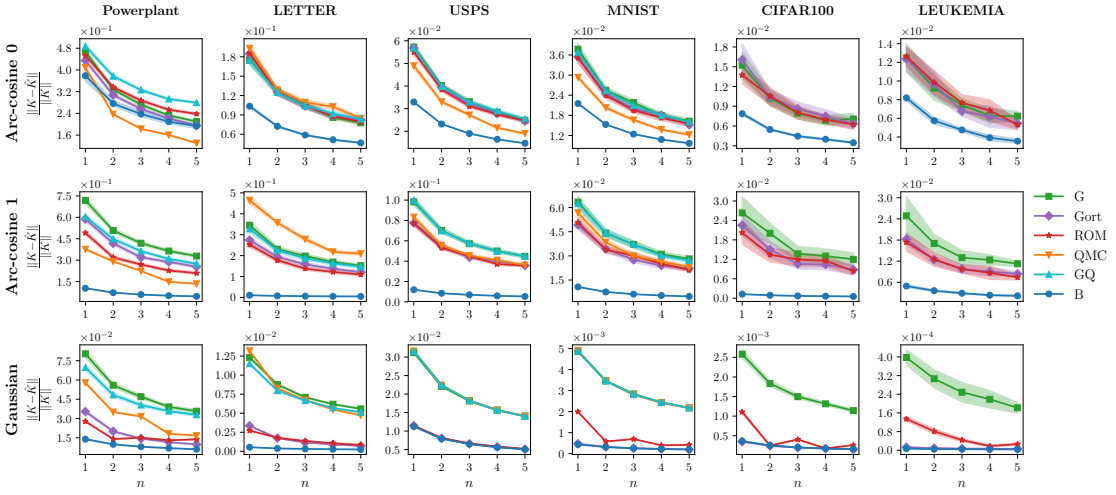


FIGURE 3.1: Kernel approximation error across three kernels and 6 datasets. Lower is better. The x-axis represents the factor to which we extend the original feature space, $n = \frac{D}{2(d+1)+1}$, where d is the dimensionality of the original feature space, D is the dimensionality of the new feature space.

compliance with the previous work on ROM we use **S**-Rademacher with three blocks: $\mathbf{M} = \sqrt{d}\prod_{i=1}^{3}\mathbf{SD}_i$, where $\mathbf{S}$ is a normalized Hadamard matrix and $\mathbb{P}(\mathbf{D}_{ii} = \pm 1) = 1/2$.

Quasi-Monte Carlo integration (QMC). Quasi-Monte Carlo integration boasts improved rate of convergence $1/D$ compared to $1/\sqrt{D}$ of Monte Carlo, however, as empirical results illustrate its performance is poorer than that of orthogonal random features (Felix et al., 2016). It has larger constant factor hidden under $\mathcal{O}$ notation in computational complexity. For QMC the weight matrix $\mathbf{M}$ is generated as a transformation of quasi-random sequences. We run our experiments with Halton sequences in compliance with the previous work.

Gaussian quadratures (GQ). We included subsampled dense grid method from (Dao et al., 2017) into our comparison as it is the only data-independent approach from the paper that is shown to work well. We reimplemented code for the paper to the best of our knowledge as it is not open sourced.

3.4.2 Kernel approximation

To measure kernel approximation quality we use relative error in Frobenius norm $\frac{\|\mathbf{K}-\hat{\mathbf{K}}\|_F}{\|\mathbf{K}\|_F}$, where $\mathbf{K}$ and $\hat{\mathbf{K}}$ denote exact kernel matrix and its approximation. In line with other works we run experiments for the kernel approximation on a random subset of a dataset. Table 3.2 displays the settings for the experiments across the datasets.

Approximation was constructed for different number of SR samples $n = \frac{D}{2(d+1)+1}$, where d is an original feature space dimensionality and D is the new one. For the Gaussian

TABLE 3.1: Space and time complexity.

Method	**Space**	**Time**
ORF	$\mathcal{O}(Dd)$	$\mathcal{O}(Dd)$
QMC	$\mathcal{O}(Dd)$	$\mathcal{O}(Dd)$
ROM	$\mathcal{O}(d)$	$\mathcal{O}(d \log d)$
Quadrature based	$\mathcal{O}(d)$	$\mathcal{O}(d \log d)$

TABLE 3.2: Experimental settings for the datasets.

Dataset	N	d	**#samples**	**#runs**
Powerplant	9568	4	550	500
LETTER	20000	16	550	500
USPS	9298	256	550	500
MNIST	70000	784	550	100
CIFAR100	60000	3072	50	50
LEUKEMIA	72	7129	10	10

kernel we set hyperparameter $\gamma = \frac{1}{2\sigma^2}$ to the default value of $\frac{1}{d}$ for all the approximants, while the arc-cosine kernels (see definition of arc-cosine kernel in the Appendix) have no hyperparameters.

We run experiments for each [kernel, dataset, n] tuple and plot 95% confidence interval around the mean value line. Figure 3.1 shows the results for kernel approximation error on LETTER, MNIST, CIFAR100 and LEUKEMIA datasets.

QMC method almost always coincides with RFF except for arc-cosine 0 kernel. It particularly enjoys Powerplant dataset with $d = 4$, i.e. small number of features. Possible explanation for such behaviour can be due to the connection with QMC quadratures. The worst case error for QMC quadratures scales with $n^{-1}(\log n)^d$, where d is the dimensionality and n is the number of sample points (Owen, 1998). It is worth mentioning that for large d it is also a problem to construct a proper QMC point set. Thus, in higher dimensions QMC may bring little practical advantage over MC. While recent randomized QMC techniques indeed in some cases have no dependence on d, our approach is still computationally more efficient thanks to the structured matrices. GQ method as well matches the performance of RFF. We omit both QMC and GQ from experiments on datasets with large $d = [3072, 7129]$ (CIFAR100, LEUKEMIA).

The empirical results in Figure 3.1 support our hypothesis about the advantages of **SR** quadratures applied to kernel approximation compared to SOTA methods. With an exception of a couple of cases: (Arc-cosine 0, Powerplant) and (Gaussian, USPS), our method displays clear exceeding performance.

We also study the derived error bounds, Proposition 3.6, empirically. While we do not know the values of the constants in the proposition, the asymptotic for the number

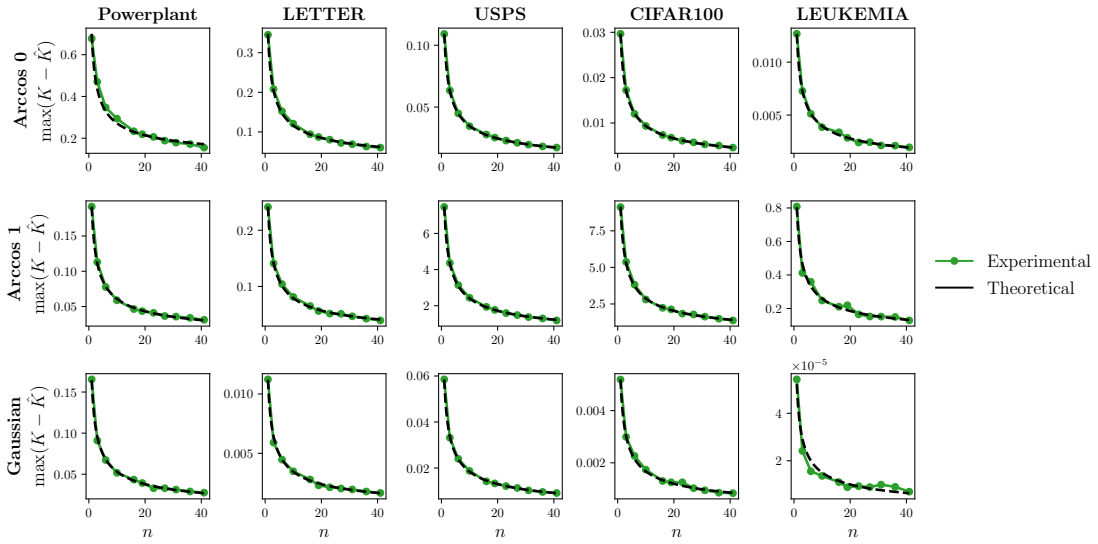


FIGURE 3.2: Empirical and theoretical kernel approximation error for different number of features.

of features required to obtain ε-accuracy of the kernel approximation is $\mathcal{O}(1/\varepsilon^2)$. To establish this rate of convergence empirically, we calculated the approximation error $\hat{\varepsilon} = \max(\mathbf{K} - \widehat{\mathbf{K}})$ for different number of features D. Then, we fitted the linear model to the obtained data using $\frac{1}{\sqrt{D}}$ as input features and $\hat{\varepsilon}$ as target values. This experiment was conducted on different datasets and different kernels. The results are given in Figure 3.2. As it can be seen, the linear model accurately explains the data, therefore, confirming the derived error bounds.

3.4.3 Classification/regression with new features

We estimate accuracy and R^2 scores for the classification/regression tasks on some of the datasets (Figure 3.3). We examine the performance with the same setting as in experiments for kernel approximation error, except now we map the whole dataset. We use Support Vector Machines to obtain predictions.

Kernel approximation error does not fully define the final prediction accuracy – the best performing kernel matrix approximant not necessarily yields the best accuracy or R^2 score. However, the empirical results illustrate that our method delivers comparable and often superior quality on the downstream tasks.

3.4.4 Walltime experiment

We measure time spent on explicit mapping of features by running each experiment 50 times and averaging the measurements. Indeed, Figure 3.4 demonstrates that the method

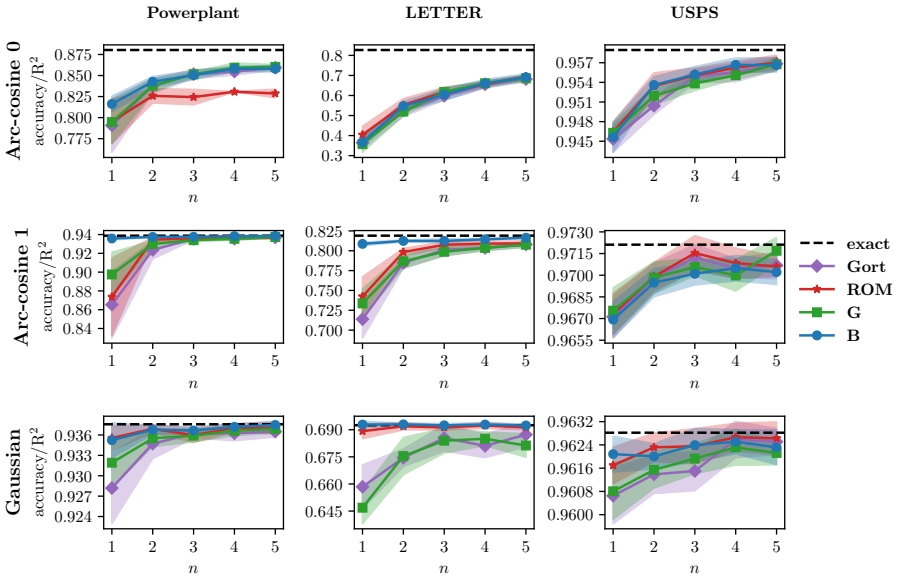


FIGURE 3.3: Accuracy/R^2 score using embeddings with three kernels on 3 datasets. Higher is better. The x-axis represents the factor to which we extend the original feature space, $n = \frac{D}{2(d+1)+1}$.

scales as theoretically predicted with larger dimensions thanks to the structured nature of the mapping.

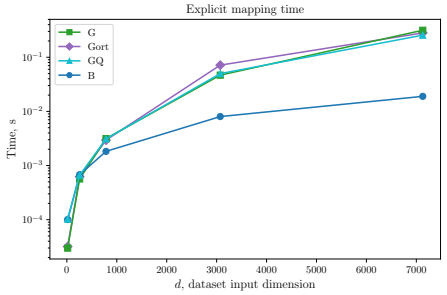


FIGURE 3.4: Time spent on explicit mapping. The x-axis represents the 5 datasets with increasing input number of features: LETTER, USPS, MNIST, CIFAR100 and LEUKEMIA.

3.5 Related work

In this section we provide brief review of the existing approaches connected to the low-rank approximation of the kernel function.

The most popular methods for scaling up kernel methods are based on a low-rank approximation of the kernel using either data-dependent or independent basis functions. The first one includes Nyström method (Drineas & Mahoney, 2005), greedy basis selection techniques (Smola & Schölkopf, 2000), incomplete Cholesky decomposition (Fine & Scheinberg, 2001).

The construction of basis functions in these techniques utilizes the given training set making them more attractive for some problems compared to Random Fourier Features approach. In general, data-dependent approaches perform better than data-independent approaches when there is a gap in the eigen-spectrum of the kernel matrix. The rigorous study of generalization performance of both approaches can be found in (Yang et al., 2012).

In data-independent techniques, the kernel function is approximated directly. Most of the methods (including the proposed approach) that follow this idea are based on Random Fourier Features (Rahimi & Recht, 2008). They require so-called weight matrix that can be generated in a number of ways. (Le et al., 2013) form the weight matrix as a product of structured matrices. It enables fast computation of matrix-vector products and speeds up generation of random features.

Another work (Felix et al., 2016) orthogonalizes the features by means of orthogonal weight matrix. This leads to less correlated and more informative features increasing the quality of approximation. They support this result both analytically and empirically. The authors also introduce matrices with some special structure for fast computations. (Choromanski et al., 2017) propose a generalization of the ideas from (Le et al., 2013) and (Felix et al., 2016), delivering an analytical estimate for the mean squared error (MSE) of approximation.

All these works use simple Monte Carlo sampling. However, the convergence can be improved by changing Monte Carlo sampling to Quasi-Monte Carlo sampling. Following this idea (Yang et al., 2014) apply quasi-Monte Carlo to Random Fourier Features. In (Yu et al., 2015) the authors make attempt to improve quality of the approximation of Random Fourier Features by optimizing sequences conditioning on a given dataset.

Among the recent papers there are works that, similar to our approach, use the numerical integration methods to approximate kernels. While (Bach, 2017) carefully inspects the connection between random features and quadratures, they did not provide any practically useful explicit mappings for kernels. Leveraging the connection (Dao et al., 2017) propose several methods with Gaussian quadratures. Among them three schemes are data-independent and one is data-dependent. The authors do not compare them with the approaches for random feature generation other than random Fourier features. The data-dependent scheme optimizes the weights for the quadrature points to yield better performance.

Chapter 4

Applications

This chapter is dedicated to the applications of the developed techniques in several different problems. We demonstrate how the described approaches for GP regression and kernel approximation can be used in a tensor completion problem, density estimate, and simultaneous localization and mapping (SLAM). This is a diverse set of problems: density estimation is a key problem in statistics, SLAM is one of the main problems in robotics, while tensor completion is a very general problem, which can be encountered in computer vision, signal processing, machine learning and many others. All of them require large-scale or online methods and can benefit from using GP- or kernel-based approaches. In this chapter, we show how large-scale GP methods can be incorporated into existing approaches. The resulting techniques provide state-of-the-art results in terms of both accuracy and computational complexity.

Section 4.1 describes our approach for tensor completion using Gaussian Processes. Our contributions here are the following:

- We consider the case when the tensor is generated by some smooth function. Using this assumption, we propose an initialization approach that can be used with a wide range of tensor completion techniques.
- We demonstrate empirically on real-world problems that different optimization methods for tensor completion benefit from using the proposed initialization. It considers assumption about tensor generating function and, therefore, allows to increase generalization power.

In section 4.2 we develop randomized feature maps-based approach for density estimation. We seek our solution in kernel exponential family of distributions with the denoising score matching loss function. Our main results are as follows:

- We derived an analytical solution for the described setup.
- We show that our solution has implicit regularization in contrast to other approaches that usually add special regularization term based on higher order derivatives of the probability density function.
- Finally, we demonstrate the benefits of the proposed approach on a set of different benchmarks and compare it with other techniques.

The last section 4.3 is devoted to GP models with random feature approximations in SLAM problem. The GP model is used here to model state of the robot depending on time. We contribute to this field by

- Developing an approach based on random features for time-continuous SLAM.
- Compared to other state-of-the-art approaches, our method is more accurate in case of noisy observations because we do not assume the sparse structure of the inverse covariance matrix. We demonstrate it on a set of synthetic benchmarks as well as real-world problems.

4.1 Tensor Completion using Gaussian Processes

In this section, we consider the tensor completion problem. We suppose that values of tensor $\mathcal{X}$ are generated by some smooth function, i.e.

$$\mathcal{X}_{i_1,\dots,i_d} = f(x_{i_1}, \dots, x_{i_d}),$$

where $(x_{i_1}, \dots, x_{i_d})$ is a point on some multi-dimensional grid and $f(\cdot)$ is some unknown smooth function. However, the tensor values are known only at some small subset of the grid. The task is to complete the tensor, i.e., to reconstruct the tensor values at all points on the grid considering the properties of the *data generating process* $f(\cdot)$.

This problem statement differs from the traditional problem statement, which does not use any assumptions about the function $f(\cdot)$. Knowing some properties of the data generating function provides insights about how the tensor values relate to each other, and this, in turn, allows us to improve the results. In this work, we assume that function $f(\cdot)$ is smooth.

There are a lot of practical applications that suit the statement. For example, modeling of physical processes, solutions of differential equations, modeling probability distributions.

Here, we propose to model the smoothness of the data generating process using GP Regression (GPR). In GPR, the assumptions about the function that we approximate are controlled via the kernel function. The GPR model is then used to construct the initial solution to the tensor completion problem.

In principle, such initialization can improve any other tensor completion technique. It means that using the proposed initialization state-of-the-art results can be obtained by employing some simple optimization procedure like the Stochastic Gradient Descent.

When the tensor order is high, the problem should be solved in some low-rank format because the number of elements of the tensor grows exponentially. The proposed approach is based on the tensor train (TT) format for its computational efficiency and ability to handle large dimensions (Oseledets & Tyrtyshnikov, 2010).

4.1.1 Tensor completion

The formal problem statement is as follows. Suppose $\mathcal{Y}$ is a d-way tensor, $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ (by tensor here we mean a multi-dimensional array). Tensor values are known only at some subset of indices $\Omega \subset \{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_d\}$. By P_Ω we denote the projection onto the set Ω, i.e.

$$P_\Omega \mathcal{X} = \mathcal{Z}, \quad \mathcal{Z}(i_1, i_2, \ldots, i_d) = \begin{cases} \mathcal{X}(i_1, i_2, \ldots, i_d) & \text{if } (i_1, \ldots, i_d) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

We formulate the tensor completion as an optimization problem

$$\begin{aligned} \min_{\mathcal{X}} \quad & f(\mathcal{X}) = \|P_\Omega \mathcal{X} - P_\Omega \mathcal{Y}\|_F^2 \\ \text{subject to} \quad & \mathcal{X} \in \mathcal{M}_r = \{\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots n_d} \mid \text{rank}_{TT}(\mathcal{X}) = \mathbf{r}\}, \end{aligned} \tag{4.1}$$

where $\text{rank}_{TT}(\mathcal{X})$ is a tensor train rank of $\mathcal{X}$ (Oseledets, 2011), which is a generalization of the matrix rank, and $\|\cdot\|_F$ is the Frobenius norm. A tensor $\mathcal{X}$ is said to be in tensor train format if its elements are represented as

$$\mathcal{X}(i_1, \ldots, i_d) = \sum_{j_1, j_2, \ldots j_d} \mathcal{G}^{(1)}_{1, i_1, j_1} \mathcal{G}^{(2)}_{j_1, i_2, j_2} \cdots \mathcal{G}^{(d)}_{j_{d-1}, i_d, 1},$$

where $\mathcal{G}^{(i)}$ is a three-way tensor core with size $r_{i-1} \times n_i \times r_i$, $r_0 = r_d = 1$. Vector $\mathbf{r}_{TT} = (r_0, \ldots, r_d)$ is called TT-rank.

Tensor train format assumes that the full tensor can be approximated by a set of 3-way core tensors, the total number of elements in core tensors is $\mathcal{O}(dnr^2)$, where $r = \max\limits_{i=0,\dots,d}\{r_i\}$, $n = \max\limits_{i=1,\dots,d}\{n_i\}$, which is much smaller than n^d.

In problem (4.1), we optimize the objective function straightforwardly with respect to tensor cores $\mathcal{G}^{(1)}, \dots \mathcal{G}^{(d)}$ while having their sizes fixed. Problem (4.1) is non-convex, so optimization methods can converge to a local minimum. To get an efficient solution, we impose two requirements:

1. Initial tensor $\mathcal{X}_0$ in tensor train format should be as close to the optimum as possible.
2. Availability of an efficient optimization procedure that will be launched from the obtained initial tensor.

These steps are independent, and one can apply any desired algorithm in each of them.

In this section, we develop the initialization algorithm, which allows obtaining accurate initial tensor for the case when the tensor of interest is generated by some smooth function. The experimental section below demonstrates that our initialization can improve the results of many optimization procedures and shows the potential of our approach to be adapted to a large number of different tensor completion techniques.

4.1.2 Initialization

We consider tensors that are generated by some function, i.e., tensor values are computed as follows

$$\mathcal{Y}_{i_1,\dots,i_d} = f(x_{i_1}, \dots, x_{i_d}),$$

where $f(\cdot)$ is some unknown smooth function and $(x_{i_1}, \dots, x_{i_d})^\top \in \mathbb{R}^d$, $i_k = 1, \dots, n_k$, $n_1, \dots, n_d$ are tensor sizes. The set of points $\{(x_{i_1}, \dots x_{i_d}) : i_k = 1, \dots, n_k; k = 1, \dots, d\}$ is a full factorial DoE, i.e., a multi-dimensional grid, and we also assume that the grid is uniform.

In this setting, the tensor completion can be considered a regression problem and can be solved by any regression technique that guarantees the smoothness of the solution. However, in the tensor completion problem, we are interested in a tensor of values of $f(\cdot)$ at a predefined finite grid of points. The tensor should be in a low-rank format to be able to perform various operations with the tensor efficiently (e.g., calculation of the norm of the tensor, dot product and other).

These observations give us the solution: build regression model $\widehat{f}$ using the observed values of the tensor, then use the obtained approximation as a black-box for the TT-cross approximation algorithm (Oseledets & Tyrtyshnikov, 2010). The last step results in a tensor $\widehat{\mathcal{X}}$ in TT format, which is a low-rank format and allows efficient computations. The next step (which is optional) is to improve the obtained solution $\widehat{\mathcal{X}}$ by using it as initialization for any other tensor completion technique.

Let us write down the set of observed tensor values into a vector $\mathbf{y}$ and the corresponding indices into a matrix $\mathbf{X}$ (each row is a vector of indices $(i_1, i_2, \ldots, i_d)$). Then, the approach for tensor completion (in TT format) can be written as follows

1. Construct initial tensor $\mathcal{X}_0$ in TT format:
   (a) Apply some regression technique using the given data set $(\mathbf{X}, \mathbf{y})$ to construct approximation of the function that generates tensor values.
   (b) Apply TT-cross method (see Section 4.1.2.2, (Oseledets & Tyrtyshnikov, 2010)) to the constructed approximation to obtain $\mathcal{X}_0$.
2. Apply some tensor completion technique using $\mathcal{X}_0$ as an initial value.

At step 1(a), the choice of the regression technique affects the result of the initialization, although it can be arbitrary. It is required to choose the regression algorithm such that it will capture the peculiarities of the tensor we would like to restore. As we stated above, we suppose that the tensor generating function is smooth (which is a common situation when modeling physical processes). Therefore, we choose a regression technique that is good at approximating smooth functions. A reasonable choice, in this case, is to use Gaussian Process Regression. GP models is a favorite tool in many engineering applications as they have proved to be efficient, especially for problems where it is required to model some smooth function (Belyaev et al., 2016). The points $(x_{i_1}, \ldots, x_{i_d})$ are not given; all we know is that at the point with multi-index $(i_1, \ldots, i_d)$ on the grid the function value is equal to $\mathcal{X}_{i_1,\ldots,i_d}$. To make the problem statement reasonable we assume that the indices are connected with the points as follows: $x_{i_k} = a_k i_k + b_k$, where $a_k, b_k \in \mathbb{R}$. So, as an input for the approximation, we set a_k and b_k such that $x_{i_k} \in [0, 1]$.

At step 1(b), we use TT-cross because it allows to efficiently approximate black-box function by a low-rank tensor in TT format. Moreover, this approach can automatically select TT-rank, making it more desirable. More details on the technique are given in Section 4.1.2.2.

The described approach has the following benefits:

1. Initial tensor $\mathcal{X}_0$, which is close to the optimal value in terms of the reconstruction error at observed values. It will push the optimization to faster convergence.

2. Better generalization ability: there are many degrees of freedom. A lot of different tensor train factors can give low reconstruction error at observed positions but can give a large error at other locations. Accurate approximation model will push the initial tensor to be closer to the original tensor in both the observed positions and unobserved ones.

3. TT-cross technique chooses rank automatically, so there is no need to tune the rank of the tensor manually.

The described approach leads to the Algorithm 1. Steps 3 and 4 of the algorithm are described in Sections 1.1 and 4.1.2.2, respectively.

Algorithm 1 Initialization

Input: $\mathbf{y}, \Omega$
Output: $\mathcal{Y}_0$ in tensor train format

1: Construct the training set $(\mathbf{X}, \mathbf{y})$ from $\mathbf{y}, \Omega$
2: Rescale inputs $\mathbf{X}$ to $[0, 1]$ interval
3: Using $(\mathbf{X}, \mathbf{y})$ build GP model $\hat{f}(\mathbf{x})$ ▷ see Section 1.1 for details
4: Apply TT-cross to $\hat{f}(\mathbf{x})$ and obtain $\mathcal{Y}_0$ ▷ see Section 4.1.2.2 for details
5: **return** $\mathcal{Y}_0$

4.1.2.1 Kernel choice and smoothness assumption

We approximate function f using the GP model. The GP model is a function from some reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ which is fully defined by the kernel function. Ideally, the function f should be from the Hilbert space $\mathcal{H}$. However, for a given kernel function, it can be difficult to identify what functions lie in the corresponding RKHS.

In practice, GP models with popular kernels (RBF kernel, Matérn kernel) provide good results, when $f \in C^k$, $k \geq 1$. So, in the experiments section, we assume that the functions are from this class of functions and use RBF kernel.

4.1.2.2 Tensor-Train cross-approximation

To approximate tensor $\widehat{\mathcal{X}}$ generated by $\hat{f}$, we use Tensor-Train cross-approximation. First, let us consider the matrix case. Suppose we are given a rank-r matrix $\mathbf{A}$ of size $m \times n$. A cross-approximation for the matrix is represented as

$$\mathbf{A} = \mathbf{C}\widehat{\mathbf{A}}^{-1}\mathbf{R},$$

where $\mathbf{C} = \mathbf{A}(:, J), \mathbf{R} = \mathbf{A}(I, :)$ are some r columns and rows of the matrix $\mathbf{A}$ and $\widehat{\mathbf{A}} = A(I, J)$ is the submatrix on the intersection of these rows and columns. To construct accurate approximation, it is required to find submatrix $\widehat{\mathbf{A}}$ of large volume. It can be done in $\mathcal{O}(nr^2)$ operations (Tyrtyshnikov, 2000).

Now for tensor $\widehat{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ the procedure is the following. At the first step, let us consider unfolding $\mathbf{X}_1$ of size $n_1 \times n_2 n_3 \cdot \cdots \cdot n_d$ and rank r_1. Using row-column alternating algorithm from (Tyrtyshnikov, 2000), we can find r_1 linearly independent columns of matrix $\mathbf{X}_1$, these columns form matrix $\mathbf{C}$. After that, by applying maxvol procedure (Tyrtyshnikov, 2000) to the matrix $\mathbf{C}$, we can find a set of row indices $I_1 = \left[i_1^{\alpha_1}\right], \alpha_1 = 1, \ldots, r_1$, matrix $\mathbf{R}$ and matrix $\widehat{\mathbf{A}}_1$ that will give the cross-approximation of unfolding $\mathbf{X}_1$:

$$\mathbf{X}_1 = \mathbf{C}\widehat{\mathbf{A}}_1^{-1}\mathbf{R}.$$

We set

$$\mathbf{G}_1 = \mathbf{C}\widehat{\mathbf{A}}_1^{-1},$$

where $\mathbf{G}_1$ is of size $n_1 \times r_1$. Next, let us form tensor $\mathcal{R}$ from r_1 rows of $\mathbf{X}_1$:

$$\mathcal{R}(\alpha_1, i_2, \ldots, i_d) = \widehat{\mathcal{X}}(i_1^{\alpha_1}, i_2, \ldots, i_d),$$

and reshape it into a tensor of size $r_1 n_2 \times n_3 \times \cdots \times n_d$. The next step is to apply the same procedure to the unfolding $\mathbf{R}_1$ of the tensor $\mathcal{R}$ and obtain the matrices $\mathbf{C}$, $\widehat{\mathbf{A}}_2$ and

$$\mathbf{G}_2 = \mathbf{C}\widehat{\mathbf{A}}_2^{-1}$$

of size $r_1 n_2 \times r_2$.

Repeating the described procedure d times, we will end up with matrices $\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_d$ of sizes $n_1 \times r_1, r_1 n_2 \times r_2, \ldots, r_{d-1} n_d \times 1$. Then, each matrix can be reshaped to the 3-way tensor of size $r_{d-1} \times n_d \times r_d$, $r_0 = r_d = 1$ and can be used as core tensors for TT format. It turns out that such representation is a TT decomposition of the initial tensor $\widehat{\mathcal{X}}$.

The exact ranks $r_1, \ldots, r_d$ are not known to us in general. They can only be estimated from the above (e.g., by the maximum rank of the corresponding unfolding). If the rank is overestimated, then the calculation of matrices $\mathbf{G}_i$ is an unstable operation (because we obtain almost rank-deficient unfolding matrices). However, in (Oseledets & Tyrtyshnikov, 2010), the authors suggest some simple modifications that overcome this issue. Therefore, we need to estimate the ranks from the above, but the estimate should not be much larger than the real rank. So, the approach is to start from some small rank, construct the tensor in TT format and then apply recompression (see (Oseledets, 2011)).

If there is a rank that is not reduced, then we underestimated that rank and should then increase it and repeat the procedure.

4.1.2.3 Computational complexity

The computational complexity of TT cross-approximation method is as follows. To perform the procedure, we need to evaluate the GP model $\mathcal{O}(dnr^2)$ times at some subset of grid points and then perform $\mathcal{O}(dnr^3)$ operations to find all maximum volume submatrices. The complexity of evaluating the GP model at one point is $\mathcal{O}(Nd)$, where N is the number of observed tensor elements. The total complexity is thus $\mathcal{O}(Nd^2nr^2 + dnr^3)$.

4.1.3 Experimental results

In this section, we present the results of the application of our approach to two engineering problems and also test it on some artificial problems to investigate how its properties depend on smoothness.

The experimental setup is the following. We try the following optimization algorithms

1. SGD – stochastic gradient descent (Zhao et al., 2018),
2. Ropt – Riemannian optimization (Steinlechner, 2016),
3. TTWopt – weighted tensor train optimization (Zhao et al., 2018),
4. ALS – alternating least squares (Grasedyck et al., 2013).

We run each algorithm with random initialization and with the proposed GP-based initialization and then compare the results.

4.1.3.1 Functions generated from GP prior

In order to study the dependence of the solution on the smoothness of the generating function, we applied the proposed approach to the toy functions generated from GP prior with different kernels. The smoothness of the generated functions is the same as the kernel that we used to generate them. Thus, we can investigate the performance of the approach for different smoothness. We considered shift-invariant kernels, i.e. $k(\mathbf{x}, \mathbf{y}) = k(r)$, where $r = \|\mathbf{x} - \mathbf{y}\|$. The list of kernels is as follows:

- Exponential kernel
$$k(r) = \exp\left(-\frac{r}{\sigma}\right).$$
The kernel is not differentiable at $\mathbf{x} = \mathbf{y} = 0$, thus the functions generated with this kernel are from C^0.

- $\text{Matern}_{3/2}$
$$k_{3/2}(r) = \left(1 + \frac{\sqrt{3}r}{\sigma}\right)\exp\left(-\frac{\sqrt{3}r}{\sigma}\right).$$
This kernel is 1-time differentiable.

- $\text{Matern}_{5/2}$
$$k_{5/2}(r) = \left(1 + \frac{\sqrt{5}r}{\sigma} + \frac{5r^2}{3\sigma^2}\right)\exp\left(-\frac{\sqrt{5}r}{\sigma}\right).$$
This kernel is 2-times differentiable.

- Radial Basis Function (RBF) kernel
$$k(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right).$$
This kernel is infinitely differentiable.

Note that despite the fact that the functions were generated using different kernel functions in the proposed approach, we used RBF kernel.

To compare how much one approach is better than the other, we calculate the relative MSE error
$$\text{MSE}_{\text{rel}} = \frac{1}{|\Omega_{test}|}\left\|\frac{P_{\Omega_{test}}\hat{\mathcal{Y}} - P_{\Omega_{test}}\mathcal{Y}}{\hat{\sigma}}\right\|_F^2,$$
where Ω_{test} is some set of indices independent from the given observed set of indices Ω, $|\Omega_{test}|$ is a size of the set Ω_{test} and $\widehat{\mathcal{Y}}$ is an obtained approximation of the actual tensor $\mathcal{Y}$, $\hat{\sigma}$ is a standard deviation of $P_{\Omega_{test}}\mathcal{Y}$. Such error can be interpreted as the ratio of unexplained variance. For each optimization technique, we calculate the difference between the error obtained using random initialization and the error obtained using GP based initialization.

For each kernel function, we generated several data sets with different number of observed points ($N \in \{100, 500, 1000, 2000, 5000\}$) and different dimensionalities ($d \in \{2, 3, \ldots, 10, 11, 13, \ldots, 19\}$). The quantity is illustrated in Figure 4.1 (note that we clamp the values to $[-1, 1]$ interval to make the figures more illustrative). It shows the improvement of one initialization over another. We can see from the figure that in most cases, GP-based initialization gives high improvement in the relative error. The only

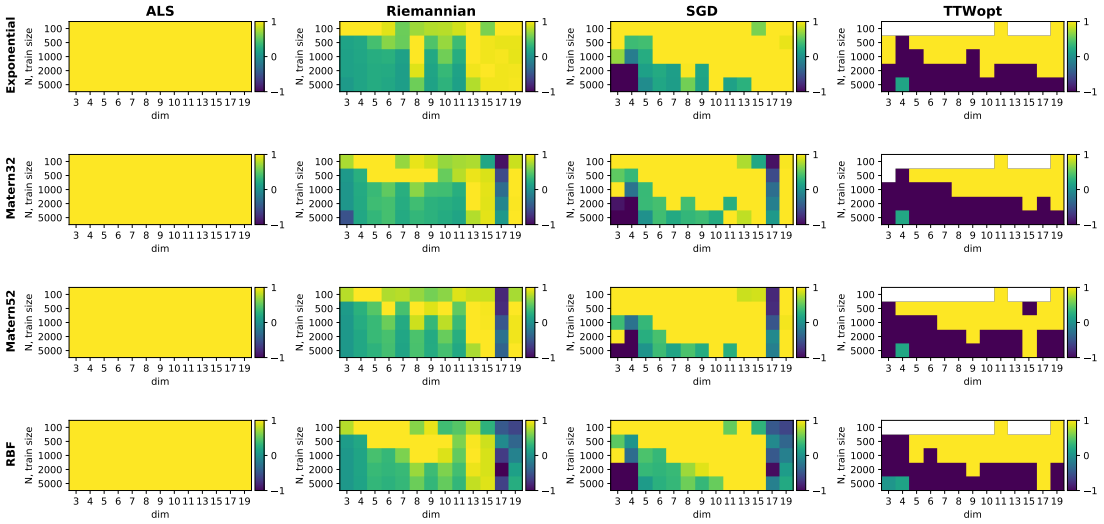


FIGURE 4.1: Improvement of GP based initialization over random initialization for different tensors and optimization methods. We clamp the improvement value to $[-1, 1]$ interval to make plots more illustrative.

exception is TTWopt method for which the benefit of the proposed initialization scheme takes place only in about half of the cases. You can also note white squares for TTWopt; they mean that the implementation of TTWopt crashed for the given data set (for some unknown reason).

4.1.4 Real world functions

We compared the approaches on two real-world problems: CMOS oscillator model and Cookie problem (see Sections 4.1.4.2 and 4.1.4.1 correspondingly). In CMOS oscillator problem, we run each optimization 10 times with different random training sets and then calculate the average reconstruction error as well as standard deviation. Cookie problem is more computationally intensive because each evaluation of the tensor value takes more resources and time. Therefore, for Cookie problem, we performed 10 runs, and the training set was the same during all runs.

The quality of the methods is measured using mean squared error (MSE)

$$MSE = \frac{1}{|\Omega_{test}|} \|P_{\Omega_{test}} \widehat{\mathcal{Y}} - P_{\Omega_{test}} \mathcal{Y}\|_F^2.$$

We also report the error of the initial tensor for random and the proposed initializations for each problem.

Note that when we use GP based initialization, the TT-rank $\mathbf{r}_{\mathrm{TT}}$ of the tensor is selected automatically by the TT-cross algorithm and max value of $\mathbf{r}_{\mathrm{TT}}$ can be larger than n.

The optimization algorithms with random initialization do not have a procedure for automatic rank selection, so we ran them with different ranks (from 1 to $\min_k n_k$) and then chose the best one.

TTWopt implementation[1] does not support high-dimensional problems. For higher dimensional problems, the authors of TTWopt propose to use SGD. The authors of TTWopt also propose truncated SVD-based initialization. The idea is to fill missing values using the mean value of the observed part of the tensor and then apply truncated SVD to obtain TT cores. However, such approach is only applicable to low-dimensional tensors, as it requires to calculate full matrices of large size.

For Ropt and ALS, we used publically available MATLAB codes [2].

4.1.4.1 Cookie problem

Let us consider parameter-dependent PDE (Ballani & Grasedyck, 2015; Tobler, 2012):

$$-\text{div}(a(x,p)\nabla u(x,p)) = 1, \quad x \in D = [0,1]^2,$$
$$u(x,p) = 0, \quad x \in \partial D,$$

where

$$a(x,p) = \begin{cases} p_\mu, & \text{if } x \in D_{s,t}, \mu = mt + s, \\ 1, & \text{otherwise,} \end{cases}$$

$D_{s,t}$ is a disk of radius $\rho = \frac{1}{4m+2}$ and m^2 is a number of disks which form $m \times m$ grid. This is a heat equation where heat conductivity $a(x,p)$ depends on x (see illustration in Figure 4.2) and p is an m^2-dimensional.

We are interested in average temperature over D: $u(p) = \int_{[0,1]^2} u(x,p)\text{d}x$. If p takes 10 possible values, then there are $\mathbf{10^{m^2}}$ possible values of $u(p)$.

In this work, we used the following setup for the Cookie problem: each parameter p lies in the interval $[0.01, 1]$, number of levels for each p is 10, number of cookies is $m^2 = 9$ and 16, size of the observed set is $N = 5000$, for the test set, we used 10000 independently generated points.

The results of tensor completion are presented in Table 4.1 (the variance of the initialization error for GP-based init is not presented as it is negligible in this case). One can see that GP-based initialization gives lower reconstruction errors both on the training set and test set except for ALS technique. ALS method with the proposed initialization

[1] https://github.com/yuanlonghao/T3C_tensor_completion
[2] https://anchp.epfl.ch/index-html/software/ttemps/

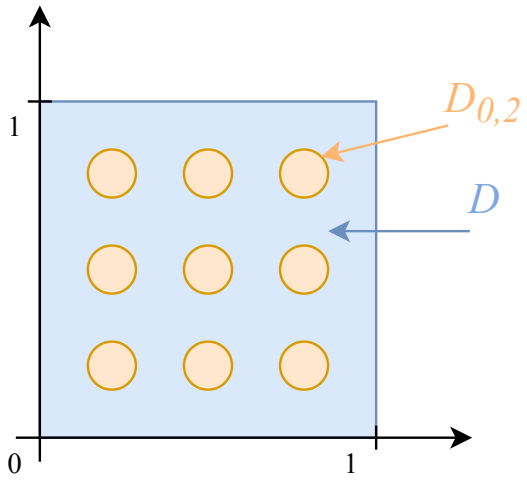


FIGURE 4.2: Illustration of Cookie problem with $m = 3$ (9 cookies).

overfits: the error on the training set is close to 0, whereas the test error is much more significant. The error on the training set is about 10^{-29}, which means that the training set was approximated with machine precision. It is not surprising if we recall that there are only 5000 observed values, while the number of free parameters that are used to construct TT is much higher.

4.1.4.2 CMOS ring oscillator

Let us consider the CMOS ring oscillator (Zhang et al., 2017). It is an electric circuit which consists of 7 stages of CMOS inverters. We are interested in the oscillation frequency of the oscillator. The characteristics of the electric circuit are described by 57 parameters. Each parameter can take one of 3 values, so the total size of the tensor is $3^{57} \approx 1.57 \times 10^{27}$. The number of observed values that were used during the experiments is $N = 5000$. For the test set, we used 10000 independently generated points.

The results of the experiments are given in Table 4.2. The table demonstrates that utilizing GP based initialization improves the results for all algorithms except ALS. ALS, in this case, overfits again: training error is extremely small, whereas the test error is much larger, though it is rather small compared to other techniques and ALS with random initialization.

All in all, the obtained results prove that GP-based initialization improves the tensor completion results in general. At least, it provides better training error. As for the error on the test set, one should be more careful as the number of degrees of freedom is large and there are many solutions that give a small error for the observed values but large errors for other values.

TABLE 4.1: MSE errors for Cookie problem

	Training set $m = 3$			
	Random init		GP init	
	error	average N iters	error	average N iters
SGD	$(1.66 \pm 0.067) \times 10^{-2}$	1500	$\mathbf{(2.86 \pm 0.18) \times 10^{-5}}$	150
Ropt	$(4.13 \pm 2.20) \times 10^{-8}$	1000	$\mathbf{(5.48 \pm 1.10) \times 10^{-10}}$	1000
TTWopt	$(2.73 \pm 0.19) \times 10^{-4}$	100	$\mathbf{(9.21 \pm 2.17) \times 10^{-7}}$	100
ALS	$(1.07 \pm 1.07) \times 10^{-4}$	100	$\mathbf{(2.39 \pm 0.60) \times 10^{-30}}$	100
Init error	$(1.15 \pm 0.24) \times 10^{8}$		$\mathbf{2.66 \times 10^{-4}}$	
	$m = 4$			
SGD	$(3.14 \pm 1.08) \times 10^{-2}$	1500	$\mathbf{(1.65 \pm 0.13) \times 10^{-5}}$	150
Ropt	$(1.42 \pm 0.01) \times 10^{-2}$	1000	$\mathbf{(3.42 \pm 0.50) \times 10^{-4}}$	1000
TTWopt	$(1.31 \pm 0.00) \times 10^{-4}$	100	$\mathbf{(1.80 \pm 0.16) \times 10^{-6}}$	100
ALS	$(6.59 \pm 3.30) \times 10^{-5}$	100	$\mathbf{(1.33 \pm 0.46) \times 10^{-29}}$	100
Init error	$(8.32 \pm 2.52) \times 10^{14}$		$\mathbf{3.14 \times 10^{-4}}$	
	Test set $m = 3$			
SGD	$(2.06 \pm 2.31) \times 10^{-1}$	—	$\mathbf{(9.97 \pm 0.40) \times 10^{-5}}$	—
Ropt	$\mathbf{(1.48 \pm 0.90) \times 10^{-7}}$	—	$(3.45 \pm 0.0165) \times 10^{-4}$	—
TTWopt	$(4.52 \pm 0.50) \times 10^{-4}$	—	$\mathbf{(5.27 \pm 0.74) \times 10^{-6}}$	—
ALS	$\mathbf{(4.37 \pm 7.73) \times 10^{-2}})$	—	$(3.78 \pm 1.08) \times 10^{0}$	—
Init error	$(1.12 \pm 0.23) \times 10^{8}$		$\mathbf{4.12 \times 10^{-4}}$	
	$m = 4$			
SGD	$(2.40 \pm 2.76) \times 10^{1}$	—	$\mathbf{(1.15 \pm 0.05) \times 10^{-4}}$	—
Ropt	$(1.47 \pm 0.003 \times 10^{-2}$	—	$\mathbf{(5.38 \pm 0.07) \times 10^{-4}}$	—
TTWopt	$(2.42 \pm 0.00) \times 10^{-4}$	—	$\mathbf{(3.02 \pm 0.17) \times 10^{-5}}$	—
ALS	$\mathbf{(3.57 \pm 5.65) \times 10^{-1}}$	—	$(1.85 \pm 60.5) \times 10^{0}$	—
Init error	$(8.33 \pm 2.46) \times 10^{14}$		$\mathbf{5.37 \times 10^{-4}}$	

TABLE 4.2: MSE errors for CMOS oscillator

	Training set			
	Random init		GP init	
	error	N iters	error	N iters
SGD	$(7.77 \pm 15.25) \times 10^{5}$	1500	$\mathbf{(3.11 \pm 4.87) \times 10^{-4}}$	150
Ropt	$(6.22 \pm 0.01) \times 10^{3}$	1000	$\mathbf{(9.50 \pm 4.28) \times 10^{-5}}$	1000
ALS	$(9.95 \pm 0.26) \times 10^{-2}$	300	$\mathbf{(3.57 \pm 0.45) \times 10^{-26}}$	300
Init error	$(1.67 \pm 3.19) \times 10^{12}$		$\mathbf{(3.95 \pm 2.19) \times 10^{-4}}$	
	Test set			
SGD	$(3.45 \pm 9.68) \times 10^{8}$	—	$\mathbf{(4.65 \pm 5.01) \times 10^{-4}}$	—
Ropt	$(6.23 \pm 0.0) \times 10^{3}$	—	$\mathbf{(9.68 \pm 4.16) \times 10^{-5}}$	—
ALS	$(1.03 \pm 0.01) \times 10^{-1}$	—	$\mathbf{(4.09 \pm 3.10) \times 10^{-4}}$	—
Init error	$(1.04 \pm 2.53) \times 10^{15}$		$\mathbf{(3.90 \pm 2.15) \times 10^{-4}}$	

4.1.5 Related work

One set of approaches to tensor completion is based on nuclear norm minimization. The nuclear norm of a matrix is defined as a sum of all singular values of the matrix. This objective function is a convex envelope of the rank function. For a tensor, the nuclear norm is defined as a sum of singular values of matricizations of the tensor.

There are efficient off-the-shelf techniques for such types of problems that apply interior-point methods. However, they are second-order methods and scale poorly with the dimensionality of the problem. Special optimization technique was derived for nuclear norm minimization (Gandy et al., 2011; Liu et al., 2013; Recht et al., 2010; Yuan & Zhang, 2016). However, calculation of the nuclear norm for tensors is a difficult task (Hillar & Lim, 2013)

Some of the approaches involve solving large semi-definite programs (Barak & Moitra, 2016; Potechin & Steurer, 2017), so they are infeasible when the dimensionality is high. More often, such techniques are applied to matrices or low-dimensional tensors as their straightforward formulation allows finding the full tensor. It becomes infeasible when we come to high-dimensional problems.

The second type of approaches is based on low-rank tensor decomposition (Acar et al., 2011; Chen et al., 2013; Kressner et al., 2014; Steinlechner, 2016; Yuan et al., 2017). There are several tensor decompositions, and all these papers derive some optimization procedure for one of them, namely, CP decomposition, Tucker decomposition (Kolda & Bader, 2009b), or TT/MPS decomposition. The simplest technique is the alternating least squares (Grasedyck et al., 2015a). It finds the solution iteratively at each iteration, minimizing the objective function w.r.t. one core while other cores are fixed.

Another approach is based on Riemannian optimization, which tries to find the optimal solution on the manifold of low-rank tensors of the given structure (Steinlechner, 2016). The same can be done using Stochastic Gradient Descent (Yuan et al., 2017). Riemannian optimization, TTWopt, ALS, and its modifications (e.g., ADF, alternating directions fitting (Grasedyck et al., 2013)), try to find the TT representation of the actual tensor iteratively (Phien et al., 2016; Grasedyck et al., 2015b). At each iteration, it optimizes TT cores such that the resulting tensor approximates well the tensor, which coincides with the real tensor at observed indices and with the result of the previous iteration at other indices. All these approaches need to specify rank manually. In (Suzuki, 2015; Zhao et al., 2015) the authors apply the Bayesian framework for CP decomposition, which allows them to select the rank of the decomposition automatically. The work (Yokota et al., 2016) introduces smoothness constraints on PARAFAC decomposition to improve the results.

In some papers, the objective is modified by introducing special regularizers to suit the problem better (Yokota et al., 2016). For example, in (Chen et al., 2013; Zhao et al., 2015) to obtain better results for visual data, a special prior regularizer was utilized. (Chen et al., 2017) proposes a regularizer based on some specific matrix norm. Many papers use total variation regularizer (Li et al., 2017) for low-rank tensor completion. The paper Qin et al. (2020) employs Shannon total variation as a regularization.

Our proposed algorithm is an initialization technique for the tensor completion problems in TT format and can be used with most of the algorithms solving such problems. If the assumptions from Section 4.1.2 (the tensor values are values of some rather smooth function of tensor indices) are satisfied, the initial value will be close to the optimal, providing better results. The question of a good initialization is rarely considered. In paper (Ko et al., 2018), a special initialization is proposed for visual data. The idea is to use some crude technique (like bilinear interpolation) to fill missing values and after that, apply SVD-based tensor train decomposition. The drawback of the approach is that it can be applied only in case of small-dimensional tensors, as we need to fill all missing values. In (Grasedyck et al., 2013), they propose special initialization for the Alternating Direction Fitting (ADF) method. This is a general technique for the tensor completion, and it does not consider the assumptions on the data generating function.

4.2 Score Matching based on Random Features

One of the core problems in statistics is a density estimation. The most well-known approach is the Maximum Likelihood Estimation (MLE). However, MLE and all other approaches based on MLE require normalizing constant to be known or computed efficiently, which is not the case in many real world problems. The intractability of the normalizing constant makes the approach infeasible. In contrast, an unsupervised score matching estimator Hyvärinen (2005) based on Fisher divergence minimization, does not depend on the normalizing constant. The resulting estimate is proved to be asymptotically normal and consistent in the case when data and model distributions supports coincide. There numerous developments of the idea Hyvärinen (2007); Lyu (2012); Gutmann & Hirayama (2012); Mardia et al. (2016); Dai et al. (2019).

Another important part of the density estimation is the class of models to search the solution in. A special interest is paid here to an exponential family of distributions which leads to the closed-form solution Hyvärinen (2007); Forbes & Lauritzen (2015); Lin et al. (2016); Yu et al. (2018); Monti & Hyvärinen (2018). A generalization of the finite-dimensional exponential families is a kernel exponential family (KEF). In this case the natural parameter is treated a function from some Reproducing Kernel Hilbert Space

(RKHS). It can be seen as infinite-dimensional generalization of the exponential family. The KEF contains all well-known exponential family densities such as Exponential, Gaussian, Gamma, etc. In addition, RKHS reveals a sufficiently rich class of estimators with convergence guarantees w.r.t. different metrics Sriperumbudur et al. (2013); Strathmann et al. (2015). The main disadvantage is the computational cost for the sample matrix inversion making this method inapplicable even for a moderate amount of the training data.

To approach the computational complexity issue Sutherland et al. (2017); Wenliang et al. (2018) propose to use Nyström-type approximation of the kernel function. Here we propose to use randomized feature maps as considered in Chapter 3. Employing special structure on the discussed in Section 3.2.4 we come up with a faster model than Nyström-type approximation. There are a lot of papers studying the convergence of the RFF models for the regression problem. The optimal learning rate with $\mathcal{O}(\sqrt{n}\log n)$ features is the same as for the full kernel Rudi & Rosasco (2017) which gives substantial speed up. The theoretical properties of using RFF for score matching is less studied, though there are some general theoretical results on RFF and higher order kernel derivatives Chamakh et al. (2020); Brault et al. (2016).

Naive approach to score matching with random features suffer from several issues. The first one is an oscillating behaviour in the tails of the distribution Strathmann et al. (2015). The second problem is poor convergence in the case of disjoint support (consistency could not be guaranteed) or in the areas where density value is close to zero Wenliang et al. (2018). Inconsistency explains low approximation accuracy in regions of almost zero density.

It was shown that convolution with small Gaussian noise (which is equivalent to the noisy data perturbation) improves learning behaviour and approximation quality, e.g. Song & Ermon (2019); Arjovsky & Bottou (2017); Roth et al. (2017). It makes the support of both densities (distribution of the data and the model distribution) the same and allows to overcome the aforementioned issue. For most of the models the convolution cannot be calculated analytically, so authors usually stick to the second-order Taylor series expansion Kingma & Cun (2010); Reehorst & Schniter (2019); Roth et al. (2017) which results in a special regularization term in the loss function. It turns out that the noise level is an important parameter. With large noise level we have better convergence but lower accuracy. With smaller noise level the convergence is less stable, but the solution is more accurate. This means, that tuning of noise level is required. Recently, it was proposed to use several noise levels optimizing cumulative objective Song & Ermon (2019).

In this section we introduce method to estimate unknown distribution using denoising score matching combined with random features. To tackle the convergence issues we convolve the loss function with symmetric noise analytically. It allows to avoid additional regularization terms as they are embedded into the loss function naturally. The derived expression of the loss function explicitly contains the noise parameters that allows us to use simple gradient-based approaches to tune these parameters. In the experimental section we demonstrate the performance of our approach both in terms of accuracy and training time. While the quality is comparable to Nyström-type approximations, the training speed is much faster.

4.2.1 Score matching

Let $\mathcal{D} = \{\mathbf{x}_a\}_{a=1}^d, \mathbf{x}_a \in \mathbb{R}^d$ be a set of observations drawn from an unknown distribution with a probability density function $p_0(\mathbf{x})$. Let $p(\mathbf{x}, \boldsymbol{\theta})$ be a model density parameterized by $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^m$. The task is to find such $\boldsymbol{\theta}^*$ that the model density is close to the real one: $p(\mathbf{x}, \boldsymbol{\theta}^*) \approx p_0(\mathbf{x})$. In score matching approach we minimize the Fisher divergence:

$$J(p_0 \| p_{\boldsymbol{\theta}}) = \frac{1}{2} \int p_0(\mathbf{x}) \| \nabla \log p(\mathbf{x}, \boldsymbol{\theta}) - \nabla \log p_0(\mathbf{x}) \|_2^2 d\boldsymbol{x}. \tag{4.2}$$

Under sufficiently weak regularity conditions (see Hyvärinen (2005)) the minimization of the Fisher divergence is equivalent to minimization of

$$J(p_0 \| p_{\boldsymbol{\theta}}) \sim \mathbb{E}_{p_0} \left[\Delta \log p(\mathbf{x}, \boldsymbol{\theta}) + \frac{1}{2} \| \nabla \log p(\mathbf{x}, \boldsymbol{\theta}) \|^2 \right]. \tag{4.3}$$

Note, that the normalizing constant does not depend on $\mathbf{x}$, therefore, $p(\mathbf{x}, \boldsymbol{\theta})$ in (4.3) could be replaced with unnormalized one $\tilde{p}(\mathbf{x}, \boldsymbol{\theta}) = p(\mathbf{x}, \boldsymbol{\theta}) Z(\boldsymbol{\theta})$. In an abuse of notation from now on we will use $p(\mathbf{x}, \boldsymbol{\theta})$ to denote the unnormalized density if it not stated explicitly. Objective (4.3) now does not depend on unknown density p_0 and provides an opportunity to estimate p_0 up to the normalizing constant using only samples drawn from p_0:

$$\hat{J}(p_0 \| p_{\boldsymbol{\theta}}) = \frac{1}{n} \sum_{a=1}^{n} \left[\Delta \log p(\mathbf{x}_a, \boldsymbol{\theta}) + \frac{1}{2} \| \nabla \log p(\mathbf{x}_a, \boldsymbol{\theta}) \|^2 \right] \to \min_{\boldsymbol{\theta}}. \tag{4.4}$$

This loss suffers from several issues. Firstly, the expression (4.2) assumes that model and data distributions have the same support. However, in real world the real distribution lies on a low-dimensional manifold embedded in $\mathbb{R}^d$ Song & Ermon (2019), while support of the model density is usually the whole space. Secondly, score matching convergence is guaranteed only in the case of supp $p_0 = \mathbb{R}^d$ (see Hyvärinen (2005)).

To tackle the issue we use Denoising Score Matching (DSM) Vincent (2011). In this approach we add noise to the data. The score matching loss in this case is given by

$$DSM(p_{\boldsymbol{\theta}}) = \mathbb{E}_{p_\varepsilon}\mathbb{E}_{p_0}\left[\Delta \log p(\mathbf{x}+\boldsymbol{\varepsilon},\boldsymbol{\theta}) + \frac{1}{2}\|\nabla \log p(\mathbf{x}+\boldsymbol{\varepsilon},\boldsymbol{\theta})\|^2\right], \tag{4.5}$$

where $p_\varepsilon(\boldsymbol{x})$ is a distribution of noise. Now both densities have the same support, so the solution converges. The optimal model satisfies $\nabla p_{\boldsymbol{\theta}} = \nabla\left[p_0 * p_\varepsilon\right](\boldsymbol{x})$, where $*$ is the convolution operator. However, $\nabla\left[p_0 * p_\varepsilon\right](\boldsymbol{x})$ is close to the true density $\nabla p_0(\boldsymbol{x})$ only when the noise is small enough.

To estimate the loss in general case we can generate finite set of noisy samples and use them to estimate expectation in the loss function. Another option is to use Taylor series expansion assuming that noise level is small. In both cases we get approximate value of the loss function. Moreover, when we use Taylor series expansion we need to calculate higher order derivatives of the model which can be computationally complex (for example, in case of neural networks). However, for the kernel exponential family the denoising score matching loss can be computed exactly.

4.2.1.1 Kernel exponential family

The kernel exponential family is a set of distributions where unnormalized probability density functions $p_f(\mathbf{x})$ satisfy $\log p_f(\mathbf{x}) = f(\mathbf{x}) + \log q_0(\mathbf{x})$, $f \in \mathcal{H}$, $\mathcal{H}$ is some Reproducing Kernel Hilbert Space (RKHS) with kernel k and q_0 is some generating density. The normalizing constant is usually not known and cannot be computed analytically. The class of such densities is rich enough. In fact it is dense in a set of continuous probability density functions that decay at the same rate as q_0.

In a well specified case, i.e. p_0 belongs to the kernel exponential family with RKHS $\mathcal{H}$, the score matching loss (4.3) can be expressed as (see Sriperumbudur et al. (2013))

$$J(p_0\|p_f) = \frac{1}{2}\langle f, Cf\rangle_{\mathcal{H}} + \langle f, \xi\rangle_{\mathcal{H}} + J(p_0\|q_0) \tag{4.6}$$

where $\partial^{\alpha,\beta}_{i,j+d}k(\mathbf{x},\mathbf{y}) = \frac{\partial^{\alpha+\beta}}{\partial x_i^\alpha \partial y_j^\beta}k(\mathbf{x},\mathbf{y})$ and

$$C = \mathbb{E}_{p_0}\left[\sum_{i=1}^{d}\partial_i k(\mathbf{x},\cdot)\otimes\partial_i k(\mathbf{x},\cdot)\right], \quad C\colon \mathcal{H}\to\mathcal{H}$$

$$\xi = \mathbb{E}_{p_0}\left[\sum_{i=1}^{d}\partial_i k(\mathbf{x},\cdot)\partial_i \log q_0(\mathbf{x}) + \partial_i^2 k(\mathbf{x},\cdot)\right] \in \mathcal{H}.$$

Using the general representer theorem the optimal $f(\mathbf{x})$ can be found as a weighted sum of the kernel derivatives located at the training samples. To find the weights we need to invert $nd \times nd$ matrix and the computational complexity, therefore, is $O(n^3d^3)$. While the convergence in RKHS of this estimator implies the convergence in L^r, in terms of Kullback-Leibler divergence and Hellinger distance, in the misspecified case a density estimator remains the same, but with convergence guarantees only for Fisher divergence.

To reduce complexity the authors of Sutherland et al. (2017) proposed to find solution in a span over a randomly selected subset of training samples (inducing points). The computational cost of this approach is $O(m^3d^3)$, where m is a number of inducing points. Additional sub-sampling over md basis functions enables even more computationally efficient approach. In this extreme case the complexity is $O(m^2nd + m^3)$. As in the case of full data usage, obtained estimator is consistent when p_0 lies in the kernel exponential family, but the rate of convergence is slower (under assumptions presented in Sutherland et al. (2017)). The misspecified case was not studied.

To obtain consistent estimator from the kernel exponential family the authors of Wenliang et al. (2018) used denoising score matching with Taylor series expansion. This results in an additional regularization term in the loss function that penalizes second derivatives of the model. The need to calculate second derivatives restricts the approach only to relatively low-dimensional cases.

Random Features Random Features based approaches were discussed in Chapter 3. The difference in score matching problems is that we work with the derivatives of the kernel function. However, the same idea can be applied to the kernel derivatives

$$\partial^{\mathbf{p},\mathbf{q}} k(\mathbf{x}-\mathbf{y}) = \int p(\mathbf{w}) \partial^{\mathbf{p}} \left[e^{j\mathbf{w}^\top \mathbf{x}} \right] \partial^{\mathbf{q}} \left[e^{-j\mathbf{w}^\top \mathbf{y}} \right] d\mathbf{w} \tag{4.7}$$

where $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$ denote multi-indices, $\partial^{\mathbf{p}} f = \frac{\partial^{|\mathbf{p_1}+\mathbf{p_2}+\cdots+\mathbf{p_d}|}}{\partial x_1^{p_1} \ldots \partial x_d^{p_d}} f$ and $\mathbf{p}$ and $\mathbf{q}$ act on the first and the second arguments of the kernel correspondingly. The theoretical properties of using random features are well studied only for the kernel ridge regression Rudi & Rosasco (2017); Li et al. (2019).

4.2.2 Kernel Denoising Score Matching

This section provides the optimal solution for the Kernel Denoising Score Matching, its RFF approximation and some error bounds of the resulting model. Here we assume that the noise distribution is symmetric, i.e. $p_\varepsilon(\mathbf{x}) = \mathbf{p}_\varepsilon(-\mathbf{x})$.

4.2.3 Denoising Score Matching in RKHS

We start from rewriting the expression for the Denoising Score Matching objective (4.5) and follow the same logic in derivation as in paper Sriperumbudur et al. (2013) with the difference that our objective function is the convolution of the usual score matching objective with noise distribution.

Let $V : \mathbb{R}^m \to \mathbb{R}$ be a convex and differentiable function. Assume that the objective function takes the form

$$J(f) = V(\langle \phi_1, f\rangle_{\mathcal{H}}, \langle \phi_2, f\rangle_{\mathcal{H}}, \dots, \langle \phi_m, f\rangle_{\mathcal{H}}) + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}},$$

for any set $\{\phi_i(\cdot)\}_{i=1}^m$, $\phi_i \in \mathcal{H}$.

For our case we define the set of functions $\{\phi_i(\cdot)\}$ as follows

$$\phi_{(a-1)d+i}(\cdot) = \partial_i k(\mathbf{x}_a + \mathbf{y}, \cdot),$$
$$\phi_{nd+1}(\cdot) = \frac{1}{n}\sum_{a,i=1}^{n,d} \partial_i^2 k(\mathbf{x}_a + \mathbf{y}, \cdot) + \partial_i k(\mathbf{x}_a + \mathbf{y}, \cdot)\partial_i \log q_0(\mathbf{x}_a + \mathbf{y}),$$

and for simplicity let us denote it as $\{\phi_i(y, \cdot)\}_{i=1}^m$, $m = nd + 1$. Now let us define a linear operator $A(\mathbf{y}) : \mathcal{H} \to \mathbb{R}^m$, $f \to \{\langle \phi_i(\mathbf{y}, \cdot), f\rangle_{\mathcal{H}}\}_{i=1}^m$. Then the objective (4.5) can be written as

$$f^* = \arg\min_{f\in\mathcal{H}} \int p_\varepsilon(\mathbf{y}) V(A(\mathbf{y})f)d\mathbf{y} + \frac{\lambda}{2}\|f\|^2, \tag{4.8}$$

with $V(\theta_1, \dots, \theta_{nd+1}) = \frac{1}{2n}\sum_{a=1}^{n}\sum_{i=1}^{d}\theta^2_{(a-1)d+i} + \theta_{nd+1}$.

Using the first order optimality condition we can see that the solution takes the form

$$f = \int p_\varepsilon(\mathbf{y}) A^*(\mathbf{y})\boldsymbol{\alpha}(\mathbf{y})d\mathbf{y}, \quad \boldsymbol{\alpha}(\mathbf{y}) = -\frac{1}{\lambda}\nabla V(A(\mathbf{y})f),$$

where $A^*(\mathbf{y}) : \mathbb{R}^m \to \mathcal{H}$ is an adjoint to $A(\mathbf{y})$. Now we are ready to formulate the proposition.

Proposition 4.1. *The solution to (4.8) has the following form*

$$f^* = B\left[-\frac{1}{n\lambda}C(\boldsymbol{\beta}^*) + \frac{1}{n\lambda^2}b\right],$$

where $\hat{A}(\mathbf{y})\colon \mathcal{H} \to \mathbb{R}^{m-1}$, $\left(\hat{A}(\mathbf{y})f\right)_i = (A(\mathbf{y})\mathbf{f})_i, i = 1, \dots, m-1$,
$B = \int p_\varepsilon(\mathbf{y})\hat{A}^*(\mathbf{y})\hat{A}(\mathbf{y})d\mathbf{y}$, $b = \int p_\varepsilon(\mathbf{y})\phi_m(\mathbf{y}, \cdot)d\mathbf{y}$, $C(\boldsymbol{\beta}) = \int p_\varepsilon(\mathbf{x})\hat{A}(\mathbf{y})\boldsymbol{\beta}(\mathbf{y})d\mathbf{y}$ *and*

$\boldsymbol{\beta}^*(\mathbf{y})$ *is the solution to*

$$\boldsymbol{\beta}(\mathbf{y}) = -\frac{1}{n\lambda}\int p_\varepsilon(\mathbf{z})\hat{A}(\mathbf{y})\hat{A}(\mathbf{z})^*\boldsymbol{\beta}(\mathbf{z})d\mathbf{z} + \frac{1}{n\lambda^2}\int p_\varepsilon(\mathbf{z})\hat{A}(\mathbf{y})\phi_m(\mathbf{z},\cdot)d\mathbf{z}. \tag{4.9}$$

See the details on derivation in B.1.1.

The optimal model requires solution of operator equation and in general this is a difficult task. In order to avoid this, let us consider a Monte-Carlo approximation of (4.9). Suppose we sampled K noise vectors $\{\mathbf{z}_k\}_{k=1}^K$, $\mathbf{z}_k \sim p_\varepsilon$. In this case the approximation to the optimal $\boldsymbol{\beta}^*$ can be found by solving the system of equations

$$\boldsymbol{\beta}_K(\mathbf{y}) = -\frac{1}{nK\lambda}\sum_{k=1}^{K}\hat{A}(\mathbf{y})\hat{A}(\mathbf{z}_k)^*\boldsymbol{\beta}_K(\mathbf{z}_k) + \frac{1}{n\lambda^2}\int p_\varepsilon(\mathbf{z})\hat{A}(\mathbf{y})\phi_m(\mathbf{z},\cdot)d\mathbf{z}. \tag{4.10}$$

The obtained result can then be used to derive an approximation of f^*, but the computational complexity is $O(n^3d^3K^3 + n^2d^2K)$. Moreover, the convolution in the second term of (4.10) could be directly computed only for a limited set of kernels, e.g. Radial Basis Function kernel (RBF).

In order to improve the computational complexity we employ RFF approach to the kernel function approximation.

4.2.3.1 RFF for Denoising Score Matching

For the RFF (3.3) we introduce the following matrix of RFF derivatives $\partial\boldsymbol{\Phi}_y$ corrupted by noise $\mathbf{y}$. The $((a-1)d+i)$-th row of matrix $\partial\boldsymbol{\Phi}_y$ is given by $[\partial\boldsymbol{\Phi}_y]_{(a-1)d+i} = \partial_i\boldsymbol{\phi}^\top(\boldsymbol{W}(\mathbf{x}_a+\mathbf{y})+\mathbf{b})$, where $\partial_i\boldsymbol{\phi}^\top(\boldsymbol{W}(\mathbf{x}_a+\mathbf{y})+\mathbf{b})$ is an element-wise partial derivative of the feature vector at point $\mathbf{x}_a$. Similarly, for the second derivatives we have $[\partial^2\boldsymbol{\Phi}_y]_{(a-1)d+i} = \partial_i^2\boldsymbol{\phi}^\top(\boldsymbol{W}(\mathbf{x}_a+\mathbf{y})+\mathbf{b})$. The finite sample solution to (4.10) is given by

$$f_K = \frac{1}{n\lambda^2}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{H}\left[-\frac{1}{K}\left(\frac{1}{K}\partial\boldsymbol{\Phi}_K^\top\partial\boldsymbol{\Phi}_K + n\lambda\mathbf{I}\right)^{-1}\partial\boldsymbol{\Phi}_K^\top\partial\boldsymbol{\Phi}_K\odot\boldsymbol{h}+\boldsymbol{h}\right] - \frac{1}{\lambda}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{h}.$$

Here operator $\odot$ denotes the Hadamard product. Let us denote

$$\boldsymbol{H} = \int p_\varepsilon(\mathbf{y})\partial\boldsymbol{\Phi}_y^\top\partial\boldsymbol{\Phi}_y d\mathbf{y}, \quad \boldsymbol{h} = \frac{1}{n}(\partial^2\boldsymbol{\Phi}_z * p(\mathbf{z}))^\top\mathbf{1}. \tag{4.11}$$

Then by taking limit over $K\to\infty$ we obtain the final RFF solution

$$f_m^* = \lim_{K\to\infty} f_K = \frac{1}{\lambda}\boldsymbol{\phi}(\cdot)^\top(\boldsymbol{H}+n\lambda\mathbf{I})^{-1}\boldsymbol{H}h - \frac{1}{\lambda}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{h}. \tag{4.12}$$

The detailed derivation can be found in B.1.2.

Similar result can be derived for the Nyström-type approximation (see B.1.7). The disadvantage in this case is that we need to calculate convolution of the first and second order derivatives with the noise distribution for each kernel. For RFF, on the other hand, all the terms remains the same for any shift-invariant kernel except the distribution of weights $\mathbf{W}$, which is much more convenient.

Another important thing we would like to stress is that in the resulting solution each feature has a weight proportional to $\exp\left(-\frac{\sigma^2}{2}\|\mathbf{w}_i\|\right)$ (see B.1.2 for details). This means that the high-frequency features have weight which is close to zero. Such behaviour can be interpreted as a regularization that penalizes oscillating terms.

There are several hyper-parameters in the approach that affects the resulting quality, namely, the kernel hyper-parameters $\boldsymbol{\theta}$, the regularization parameter λ and, assuming that the noise is zero-mean Gaussian, the noise variance σ. To tune these parameters we use the loss on the hold-out (validation) set. The loss in this case is ordinary score matching (no denoising) loss as we would like to estimate how good our model approximates the original data, not the noisy one.

Another important part of the algorithm is the base density q_0. From a theoretical point of view, base density is responsible for the tails of the distribution and do not affect the estimator in the areas with high density. Therefore, in this section we consider three different options for q_0: uniform distribution with support bounded by particular training sample, multivariate Gaussian distribution and the mixture of Gaussians. In the latter case, q_0 is fitted before training using Bayesian Mixture Model Bishop (2007).

At the end of the training we estimate the normalizing constant via importance sampling as was proposed in Wenliang et al. (2018). It should be noted that in the case of uniform base density normalization could not be estimated properly due to the unknown data support measure. The whole method is summarized in Algorithm 2.

The total complexity of the proposed approach is $O(m^3 + nm^2 + nmd)$, where $O(nmd)$ operations are required generate random features, $O(nm^2)$ is to compute feature matrix $\boldsymbol{H}$ and $O(m^3)$ corresponds to the matrix inversion which can be reduced to $O(m^2)$ in some cases by using iterative methods for solving systems of linear equations.

Now, let us provide the bounds on the error of the approximation of the proposed approach. Let us introduce the derivatives of the exact kernel matrix

$$\partial^p \partial^q \boldsymbol{K}_{(a-1)d+i,(b-1)d+j} = \partial_i^p \partial_{d+j}^q k(\mathbf{x}_a, \mathbf{x}_b), \quad p, q \in \mathbb{N}_+.$$

Algorithm 2 Kernel denoising score matching.

Require: Training set $\mathcal{D}$, m — number of Fourier features, n_z — number of samples to estimate normalization constant, initial regularization parameter λ

1: Fit $q_0(\mathbf{x})$ to the given data set.
2: **while** not stopping condition **do**
3: **for** mini-batches $\mathcal{D}_t, \mathcal{D}_v \in \mathcal{D}$ **do**
4: Compute random Fourier approximation f_m^* using equation (4.12) on $\mathcal{D}_t$.
5: Compute ordinary score matching loss on validation

$$\hat{J}_{val}(\lambda, \sigma, \boldsymbol{p}_k) = \frac{1}{|\mathcal{D}_v|}\left[\mathbf{1}^\top \partial^2 \boldsymbol{\Phi}_v \boldsymbol{b}_t + \frac{1}{2}\boldsymbol{b}_t^\top \partial \boldsymbol{\Phi}_v^\top \partial \boldsymbol{\Phi}_v \boldsymbol{b}_t + \boldsymbol{b}_t^\top \partial \boldsymbol{\Phi}_v^\top \nabla \log q_0(\mathbf{x})\right]$$

6: Do gradient step over hyper-parameters (λ, σ, $\boldsymbol{p}_k$)
7: **end for**
8: **end while**
9: Compute $f_m^* = \boldsymbol{\phi}(\cdot)^\top \boldsymbol{b}_{\mathcal{D}}$ using full dataset $\mathcal{D}$
10: Compute normalization constant approximation $\hat{Z}$ via importance sampling

$$\hat{Z} = \frac{1}{n_z}\sum_{i=1}^{n_z} \frac{f_m^*(\mathbf{x}_i)}{q_0(\mathbf{x}_i)}, \quad \mathbf{x}_i \sim q_0(\mathbf{x})$$

return $\log p_f = f_m^* - \hat{Z}$

We also denote the derivatives of the random feature vector as

$$\partial^p \boldsymbol{\phi} = \left(\partial_1^p \boldsymbol{\phi}(\mathbf{w}^\top \mathbf{x}_1 + b) \quad \cdots \quad \partial_d^p \boldsymbol{\phi}(\mathbf{w}^\top \mathbf{x}_n + b)\right)^\top.$$

The error bounds for score matching with RFF is given by the following theorem.

Theorem 4.2. *Let $\delta \in (0,1)$, $\varepsilon > 0$, then for $n \geq \frac{8}{3\varepsilon^2}\log\frac{m}{\delta}$ and assuming that $\mathbf{D}_1 = \mathbb{E}_{\mathbf{w}}\mathrm{tr}[\partial\boldsymbol{\phi}\partial\boldsymbol{\phi}^\top]\partial\boldsymbol{\phi}\partial\boldsymbol{\phi}^\top < \infty$, $\mathbf{D}_2 = \mathbb{E}_{\mathbf{w}}\mathrm{tr}[\partial^2\boldsymbol{\phi}\partial^2\boldsymbol{\phi}^\top]\partial\boldsymbol{\phi}\partial\boldsymbol{\phi}^\top < \infty$, we have that with probability at least $(1-\delta)$ the following upper bound on distance between an averaged RFF score matching solution $f_{n,m}^*$ and exact kernel solution f_n^* holds*

$$\begin{aligned}\mathbb{E}_{\mathbf{x},\mathbf{w}}(f_{n,m}^*(\mathbf{x}) - f_n^*(\mathbf{x}))^2 \leq \frac{2}{\lambda^2 n^2 m^2}\Big[&m\|\partial\partial\boldsymbol{K}^{\frac{1}{2}}(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2 + \\ &m\mathbf{1}^\top\partial^2\partial^2\boldsymbol{K}\mathbf{1} + (1+\varepsilon m)\|\mathbf{D}_2^{\frac{1}{2}}\mathbf{1}\|^2 + \\ &(1+\varepsilon m)\|\mathbf{D}_1^{\frac{1}{2}}(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2\Big].\end{aligned}$$

The proof of the theorem is given in B.1.3.

4.2.3.2 Discussion

While RFF kernel approximation admits computationally efficient solution of score matching, the convergence properties remains an open question. Using results from Sriperumbudur et al. (2013) the convergence can be established only in the RKHS that corresponds to the approximate kernel. So we have the following relation

$$J(p_0 \| p_{\lambda,n,m}) \to \inf_{p \in \tilde{\mathcal{P}}} J(p_0 \| p) \quad \lambda \to 0,\ \lambda n \to \infty,\ n \to \infty,$$

where $p_{\lambda,n,m}$ is the density obtained using m features and $\tilde{\mathcal{P}}$ is an exponential family with sufficient statistic $\phi(\boldsymbol{W}\mathbf{x} + \boldsymbol{b})$. To upper bound the error of the approximation we can consider the following inequality

$$\|f_{\lambda,n,m} - f_0\| \leq \|f_{\lambda,n,m} - f_{\lambda,m}\| + \|f_{\lambda,m} - f_\lambda\| + \|f_\lambda - f_0\|,$$

where f_λ minimizes (4.6) and $f_{\lambda,n}$ is a solution of a finite sample version of (4.6), $\|\cdot\|$ is a norm in $L^2(\mathbb{R}^d, p_0)$. $\|f_{\lambda,n,m} - f_{\lambda,m}\|$ includes the term $\|\hat{\xi}_m - \hat{\xi}\| = O(m^{-\frac{1}{2}})$ that can obtained using concentration lemma from Sutherland et al. (2017) under additional assumption on the boundness of the derivatives of the approximate kernel. This implies that we should potentially use $O(n)$ features to obtain the same convergence rates as in the case of exact solution. To reduce the lower bounds on number of features we need to provide refined analysis in a way similar to Rudi & Rosasco (2017); Li et al. (2019) in the future study.

In the case of Denoising Score Matching the estimated density will converge to the $p^* * p_\varepsilon$, where $p^* = \inf_{p \in \tilde{\mathcal{P}}} J(p_0 \| p)$ is the density from $\tilde{\mathcal{P}}$ closest to p_0. So, on one hand the noise variance should be as small as possible. On the other hand, the Wasserstein distance between the approximation and the true density for the Denoising Score Matching can be upper bounded as follows

$$W(p_0, p) \leq \mathbb{E}[\|\boldsymbol{\varepsilon}\|^2]^{\frac{1}{2}} + \tilde{C}\sqrt{J(p_0 * p_\varepsilon, p_{\lambda,n,m} * p_\varepsilon)},$$

where $\mathbb{E}[\|\boldsymbol{\varepsilon}\|^2] = n\sigma^2$. The second term takes large value for small noise levels (due to different supports of the approximate density and the true density) and smaller values for large noise values. So the choice of σ is a trade-off between estimator stability and how close it is to the unknown density function p_0.

4.2.4 Results

4.2.4.1 Experimental setup

In all our experiments we used RBF kernel with diagonal covariance matrix, the noise was assumed to be isotropic Gaussian (though in general we can use arbitrary noise covariance matrix).

We compare the proposed approach (DSM RFF), ordinary score matching with RFF (SM RFF), exact kernel solution (4.6) (Exact) and its Nyström version with subsampled basis Sutherland et al. (2017) (Nyström). We used original implementations of this model from Gre.

The comparison is conducted on two types of data: artificially generated 2D densities and datasets from the UCI repository Dua & Graff (2017) (the particular choice of data is motivated by previous research on kernel exponential family Sutherland et al. (2017); Strathmann et al. (2015); Wenliang et al. (2018)):

1. Synthetic data generated from the following densities: a mixture of Gaussians, Uniform, Mixture on Uniforms, Cosine, Funnel, Banana, Ring, Mixture of Rings.
2. RedWine, WhiteWine, MiniBoone.

The Exact kernel model was not compared on MiniBoone dataset because it's too computationally expensive.

To estimate the quality of models the following metrics were used:

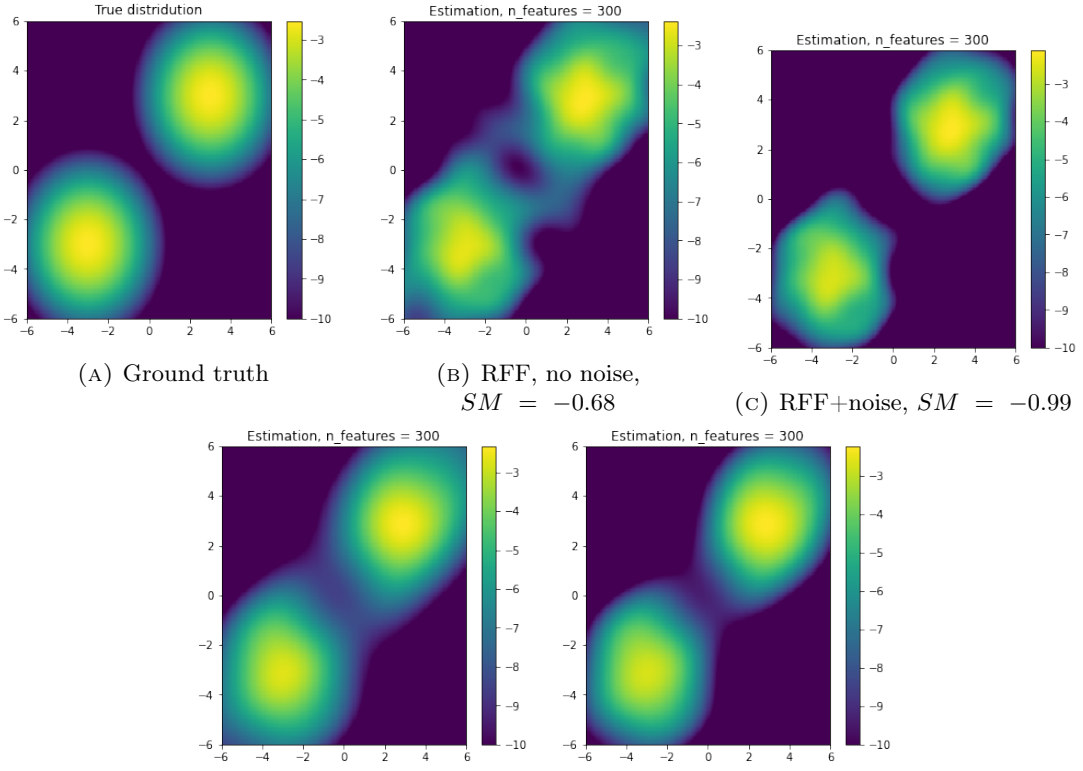
1. Log-likelihood (higher is better). It requires the normalization constant which can only be approximated, so the log-likelihood tends to be overestimated Wenliang et al. (2018).
2. Fisher divergence (lower is better). It requires true log-density gradient to be known and hence could be estimated only for artificial data, moreover, for uniform settings could be computed only on the support of true density. Alternatively, score-matching could be used, but scores for different models are not comparable in general.
3. Finite-Set Stein Discrepancy (FSSD) goodness of fit test Jitkrittum et al. (2017); GOF with 0.05 significance level. We used Gaussian kernel, and its lengthscale was chosen to be median over pairwise distances between samples in order to avoid optimization over test points for the particular model, otherwise, we can not compare

models. FSSD statistic almost surely equals to zero if and only if model density and p_0 coincide.

4. Wasserstein distance. In order to estimate this quantity, we used Metropolis Adjusted Langevin Algorithm (MALA) Roberts & Rosenthal (2001) to draw samples from the model densities. We used step-size 0.1, chain length was 10^4 with $5 \cdot 10^3$ burn-in.

4.2.5 Results

We start by considering an approximate denoising approach (see B.1.6 for derivation) to figure out if there is a benefit from the convolution with noise. To accomplish this we construct illustrative experiment with 300 RFF features for Gaussian mixture. We used multivariate Gaussian distribution for q_0 and the training set size was 10^3. The results are presented on Fig. 4.3, from which it is clear that noisy approach better estimates ground truth in between components region even with the presence of small noise Also note the that there are less oscillations when we add noise to the data.



(A) Ground truth

(B) RFF, no noise, $SM = -0.68$

(C) RFF+noise, $SM = -0.99$

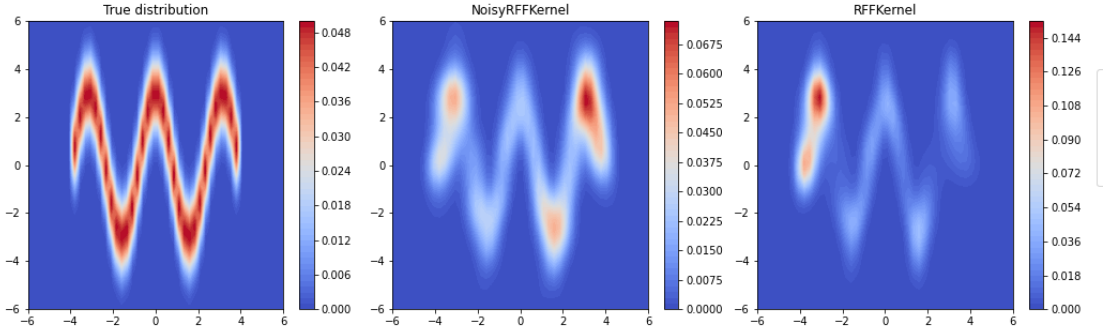
(D) RFF, no noise, wide kernel, $SM = -0.49$

(E) RFF+noise, wide kernel, $SM = -0.67$

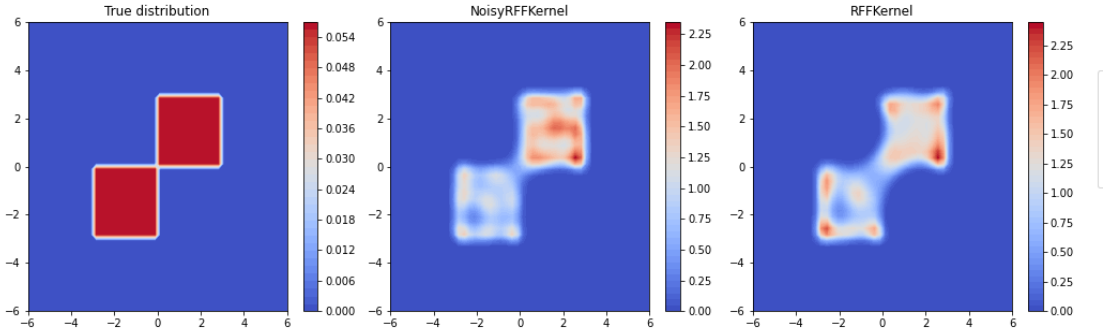
FIGURE 4.3: Comparison of score-matching with and without noise, noise variance is $\sigma = 5 \cdot 10^{-4}$. We clip values of log-density that less then -10.

The next step is to compare the proposed algorithm to other approaches on synthetic 2D data and datassets from UCI. In this case we used 512 Random Fourier Features. As the base density q_0 we used mixture of Gaussians. As in the previous example a relatively small sample size was used. Our models were trained for 60 iterations using Adam optimizer with 0.1 learning rate, 512 features were used.

For cosine data the form of distribution estimated via DSM RFF is much closer to the real one. However, for the mixture of uniforms it fails to correctly estimate weights of the components. In both of this densities, we observe model misspecification in the case of SM RFF because all other 2D densities have full space support. For the rest distributions there is no significant visual difference.



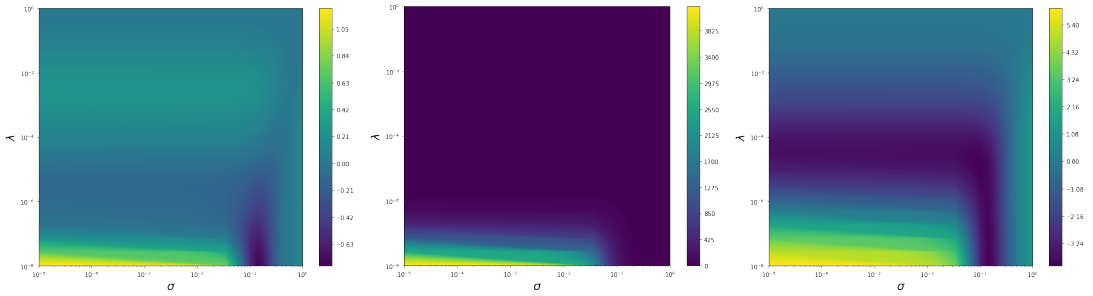
(A) Cosine



(B) Mixture of uniforms

FIGURE 4.4: Density estimates using DSM RFF (middle column) and SM RFF (right column). The ground truth density is in the first column.

These experiments showed that denoising score matching with RFF in general works better for distributions with bounded support. For the multimodal distributions it could fail to correctly estimate weights of components or oversmooth the areas between components. Another observation about the approach is that in some cases it tends to choose large noise variance. In Fig. 4.5 we visualize the dependence of the loss on the regularization parameter and noise variance for several 2D distributions. Interestingly, for "good" distributions (like Funnel, that have full space support and one mode) the loss surface has wide minimum w.r.t regularization and noise variance. For multimodal distributions



(A) Uniform (B) Funnel (C) Two Rings

FIGURE 4.5: Dependence of loss on the regularization parameter λ (y axis) and noise σ (x axis).

TABLE 4.3: Metrics for the data sets from UCI repository.

Data set	Model	Log-likelihood	FSSD	Wasserstein distance	time, s
RedWine	DSM RFF	-11.64	0.38	0.24	62
	SM RFF	-11.72	0.43	0.25	61
	Nyström	-17.23	0.11	0.73	0.2×10^4
WhiteWine	DSM RFF	-12.81	0.57	0.33	180
	SM RFF	-12.22	0.53	0.11	180
	Nyström	-17.79	0.23	0.67	1×10^4
MiniBoone	DSM RFF	-93.11	307.67	0.49	0.5×10^4
	SM RFF	-4580.20	2×10^8	0.48	0.5×10^4
	Nyström	-46.06	0.02	0.75	0.6×10^4

the loss surface has narrower minimum. For uniform distribution (which differs from other that it has bounded support) the minimum w.r.t noise variance is narrow but it is also separated from zero. This indicates the need for the noise in such cases.

In Figure 4.6 we plot all the metrics for all data sets. For each dataset each metric was normalized across methods to have unit norm. This was done only for better visualization. The original values are given in B.2. The figure illustrates the mean value of the metrics and corresponding variance calculated across 10 runs. From the figure we can see, that w.r.t. almost all metrics (except the log-likelihood) the proposed approach shows better or comparable results in many cases. Actually, the Wasserstein distance is smaller for DSM RFF for all data sets. We can also see, that SM RFF tends to have larger variance than its noisy version.

In Table 4.3 we provide results for the datasets from the UCI repository as well as the training time. The MiniBoone dataset is large and the Nyström-based implementation could not fit into memory, so we had to train the model using only a subset of 15000 samples. Other methods were trained using the whole data set. To fairly compare the training time the experiments were conducted on Intel(R) Core(TM) i7-7820X CPU @ 3.60GHz with 64Gb RAM. We can see that the proposed approach is much faster than the implementation of the Nyström based approach.

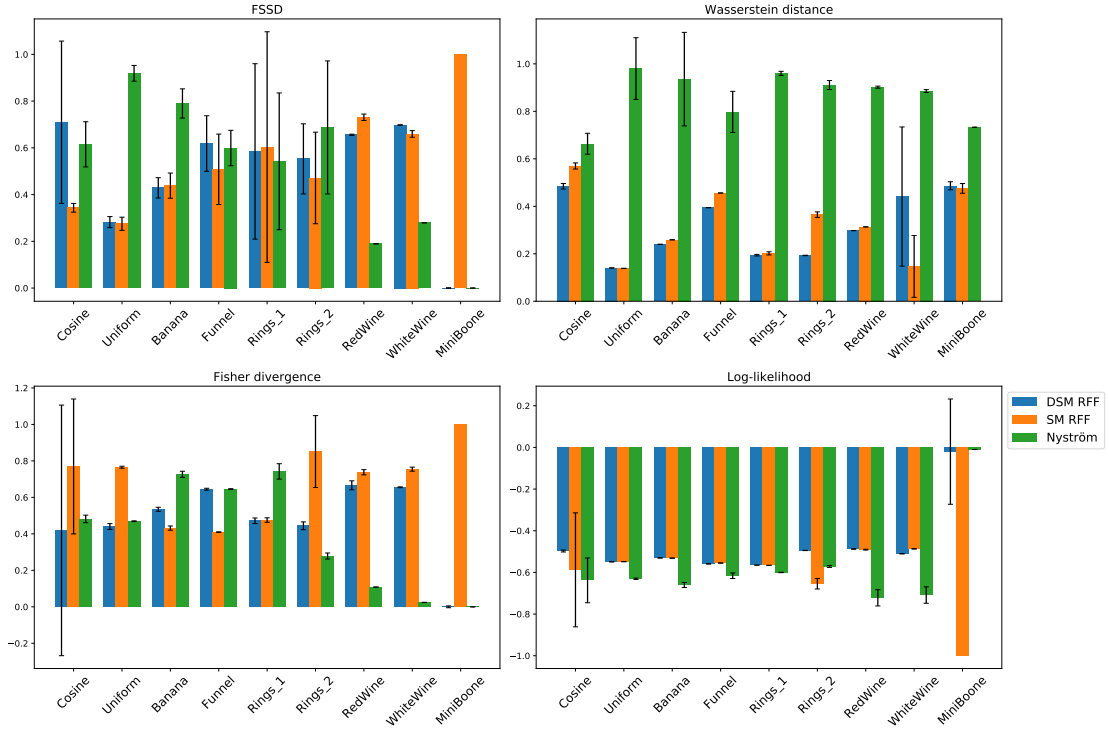


FIGURE 4.6: Metrics on different datasets for different methods. For each dataset each metric was normalized across methods to have unit norm. We did it only for better visualization.

4.3 Simultaneous Localization and Mapping with Random Features

Since the last century, probabilistic state estimation has been a core topic in mobile robotics, often as part of the problem of simultaneous localization and mapping Bailey & Durrant-Whyte (2006); Durrant-Whyte & Bailey (2006). Recovery of a robot's position and a map of its environment from sensor data is a complicated problem due to both map and trajectory are unknown as well as the correspondences between observations and landmarks Thrun et al. (2005).

Well-known approaches to this problem, such as square root smoothing and mapping (SAM)Dellaert & Kaess (2006), relied on regression-based techniques that leverage the problem's sparse structure to calculate a solution effectively. However, this technique had a great disadvantage, i.e., we may need to collect all the data before trajectory estimation (batch updates)Kaess et al. (2007). This method was improved in Kaess et al. (2008), where incremental smoothing and mapping (iSAM) was introduced. iSAM requires expensive periodic batch steps to keep re-linearization and sparsity. This method has been further improved in iSAM 2.0 Kaess et al. (2012). In iSAM 2.0 an effective graph structure, the Bayes tree Kaess et al. (2010), is used to accomplish incremental variable

reordering and just-in-time reordering, thus reducing the bottleneck caused by batch variable reordering and re-linearization. This method is widely used as state-of-the-art, and this work has gained several sequels, such as Wang et al. (2018); Cunningham et al. (2013); Strömberg (2017); Dong & Lv (2019).

Most trajectory estimation and mapping methods, including SAM-based ones, have considered the problem in a discrete-time fashion. However, discrete-time representations are constrained because they are not easily adapted to irregularly distributed poses or asynchronous measurements over trajectories. Such limitations would be well addressed by a continuous-time version of the SAM problem where measurements regulate the trajectory at any time step. The robot trajectory, seen from this perspective, is a function $\boldsymbol{x}(t)$, which corresponds to a robot state at every time t. Simultaneous trajectory estimation and mapping (STEAM) presents the problem of estimating this function along with landmark positions Barfoot et al. (2014a); Barfoot (2017). In the work Furgale et al. (2012) they formally derive a continuous-time SLAM problem and demonstrates the use of a parametric solution for atypical SLAM calibration problems. The use of cubic splines to parameterize the robot trajectory can also be seen in the estimation schemes in Bibby & Reid (2010); Fleps et al. (2011); Droeschel & Behnke (2018). In the work Tong et al. (2013) the parametric state representation was proposed due to practicality and effectiveness. The advantages of this method are that they can precisely model and interpolate asynchronous data to recover a trajectory and estimate landmark positions. The disadvantages of that algorithm are that it requires batch updates and considerable computational problems that are natural for regression.

In the work Yan et al. (2017), the critical update to increase the efficiency of existing GP approach to solve the STEAM problem was introduced. It combines benefits of iSAM 2.0 and Barfoot's work Barfoot et al. (2014b) and provide the GP-based solution to the STEAM problem that computationally efficiently even for large datasets. However, in this work GP have several constraints to be able to deal with sparse measurements and provide robot position at any point of interest.

The main drawback of the paper is that the proposed GP prior impose constraints on the kernel function and thus limits the number of possible functions that GP can model. They use state-space formulation to conduct computationally efficient inference for GP. However, the efficiency is achieved by the means of using kernel functions that impose Markovian structure on the trajectory: it is supposed that two points on the trajectory are conditionally independent given all other points if these two points are not neighboring. However, in some cases the accuracy of the trajectory estimate can be increased by adjusting the estimate at the current point using all previous points in the trajectory, especially when the observations contain a considerable amount of noise.

In recent years a lot of effort has been put to develop large-scale GP models without any constraints on the kernel function Rudi et al. (2017); Wang et al. (2019); Munkhoeva et al. (2018). There are two main approaches to scale up the GP model. The first one is based on Nyström approximation Quiñonero-Candela & Rasmussen (2005). The idea is to approximate the kernel function using a finite set of basis functions that are based on eigenvectors of the kernel matrix. This approach is data-dependent and needs updating the basis function when new observations arrive. Another set of methods is based on Random Fourier Features Rahimi & Recht (2008). In these approaches the basis functions (features) are constructed based solely on the kernel function and independent of the data set. It provides additional computational benefits and is more attractive for SLAM problems.

In this section we develop random features based SLAM approach. It is constructed solely based solely on the kernel function and independent of the data set. It provides additional computational benefits compared to other approaches and, therefore, is more attractive for SLAM problems. We build a low-rank approximation of the kernel matrix. It is dense, so we do not assume the conditional independence of the points on the trajectory. At the same time we maintain reasonable computational complexity by limiting the number of random features. We conduct several experiments and discuss the results of running the method on synthetic data and well-known real-world benchmarks.

4.3.1 SLAM

From a probabilistic point of view, there are two main forms of the problem: online-SLAM and FULL-SLAM. Online SLAM (4.13) involves estimating the posterior over the current pose along the map:

$$p\left(\boldsymbol{x}_t, \boldsymbol{l} | \mathbf{z}, \mathbf{u}\right), \tag{4.13}$$

where $\boldsymbol{x}_t$ is the pose at time $t, \boldsymbol{l}$ is the map (in this work, we consider $\boldsymbol{l}$ to be the map of landmarks), and $\mathbf{z} = \begin{bmatrix}\boldsymbol{z}_(t_1) & \cdots \boldsymbol{z}(t_N)\end{bmatrix}$ and $\mathbf{u} = \begin{bmatrix}\boldsymbol{u}_(t_1) & \cdots \boldsymbol{u}(t_N)\end{bmatrix}$ are the measurements and controls, respectively. In full SLAM (4.14), we seek to calculate a posterior over the entire path $\mathbf{x} = \begin{bmatrix}\boldsymbol{x}_(t_0) & \cdots \boldsymbol{x}(t_N)\end{bmatrix}$ along with the map, instead of just the current pose $\boldsymbol{x}_t$:

$$p\left(\mathbf{x}, \boldsymbol{l} | \mathbf{z}, \mathbf{u}\right). \tag{4.14}$$

4.3.2 RFF-SLAM

We use GP regression with RFF to estimate the state variables corresponding to trajectory and map (landmarks). Our model is as follows

$$\begin{aligned} \boldsymbol{x}(t) &\sim \mathcal{GP}(\boldsymbol{\mu}_x(t), \boldsymbol{k}(t, t')), \\ \boldsymbol{l} &\sim \mathcal{N}(\boldsymbol{\mu}_l, \mathbf{L}), \\ \boldsymbol{z}_i &= \boldsymbol{h}\left(\begin{bmatrix} \boldsymbol{x}(t_i) \\ \boldsymbol{l} \end{bmatrix}\right) + \boldsymbol{n}_i, \end{aligned} \tag{4.15}$$

where $\boldsymbol{x}(t)$ is a state of the robot at timestamp t, $\boldsymbol{l}$ is a vector of M landmarks, $\boldsymbol{h}(\cdot)$ is a non-linear measurement model, $\boldsymbol{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i)$ is measurement noise, $t_1, \ldots, t_N$ is a sequence of measurement times and $(\boldsymbol{\mu}_l, \mathbf{L})$ are prior mean and the covariance of the landmarks positions.

The paper Tong et al. (2013) uses GP for SLAM and provides the main equations to solve the problem. We follow their approach with the difference that we utilize RFF approximation (see Section 3.1.1) of the RBF kernel given by

$$k(t, t') = \sigma^2 \exp\left(-\frac{\|t - t'\|_2^2}{2\sigma_l^2}\right).$$

For the RBF kernel its Fourier transform is defined by $p(\boldsymbol{w})$ being a Gaussian distribution $\mathcal{N}\left(0, \frac{1}{\sigma_l^2}\mathbf{I}\right)$. Explicit mapping (3.4) allows working in weight-space view

$$\boldsymbol{x}(t) = \boldsymbol{\mu}_x(t) + \begin{bmatrix} \phi_1(t)\boldsymbol{b}_x^{(1)} \\ \cdots \\ \phi_d(t)\boldsymbol{b}_x^{(d)} \end{bmatrix} + \varepsilon, \quad \boldsymbol{b}_x^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_b^{(i)}, \mathbf{K}_i), i = \overline{1, d},$$

where d is a state size, $\boldsymbol{b}_x^{(i)} \in \mathbb{R}^d$, $\phi_i(\boldsymbol{x})$ is a feature map for the i-th state variable and $\mathbf{K}_i$ is a prior covariance matrix of the random variable $\boldsymbol{b}_x^{(i)}$. In principle, the same feature map can be used for all variables, however, it can be reasonable to use different features (corresponding to different kernels) to model different types of variables (for example, coordinates on the map and angles).

Let us denote

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_x^{(1)} \\ \cdots \\ \boldsymbol{b}_x^{(d)} \\ \boldsymbol{l} \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_b^{(1)} \\ \cdots \\ \boldsymbol{\mu}_b^{(d)} \\ \boldsymbol{\mu}_l \end{bmatrix}, \mathbf{K} = \begin{bmatrix} \mathbf{K}_1 & & \\ & \ddots & \\ & & \mathbf{K}_d \end{bmatrix},$$
$$\mathbf{P} = \begin{bmatrix} \mathbf{K} & \\ & \mathbf{L} \end{bmatrix}, \boldsymbol{\Phi}_i = \begin{bmatrix} \phi_1(t_i) & & & \\ & \cdots & & \\ & & \phi_d(t_i) & \\ & & & \mathbf{I}_{2M} \end{bmatrix}, \tag{4.16}$$
$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 & & \\ & \ddots & \\ & & \mathbf{R}_N \end{bmatrix}.$$

Now to obtain both the robot states and landmarks position $\boldsymbol{b}$ we employ maximum a posteriori (MAP) estimate

$$\begin{aligned} p(\boldsymbol{b}|\boldsymbol{z}) &= \frac{p(\boldsymbol{z}|\boldsymbol{b})p(\boldsymbol{b})}{p(\boldsymbol{z})} \\ &\propto -\frac{1}{2}\left(\sum_{i=1}^{N} \|\boldsymbol{z}_i - \boldsymbol{h}(\boldsymbol{\Phi}_i\boldsymbol{b})\|^2_{\mathbf{R}_i} + \|\boldsymbol{b} - \boldsymbol{\mu}\|^2_{\mathbf{P}}\right) \\ &\to \max_{\boldsymbol{b}}. \end{aligned} \tag{4.17}$$

To solve the problem we do the following. Suppose, that we have an initial guess $\bar{\boldsymbol{b}}$. We update the estimate iteratively by finding the optimal perturbation vector $\delta\boldsymbol{b}^*$ for the linearized measurement model. Namely, we apply the first order Taylor expansion to the measurements model

$$\boldsymbol{h}(\boldsymbol{\Phi}_i\boldsymbol{b}) \approx \boldsymbol{h}(\boldsymbol{\Phi}_i\bar{\boldsymbol{b}}) + \mathbf{H}_i\delta\boldsymbol{b}, \quad \mathbf{H}_i = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{y}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{\Phi}_i\bar{\boldsymbol{b}}}.$$

Plugging linearized measurement into (4.17) we obtain the following optimization problem

$$\delta\boldsymbol{b}^* = \arg\min_{\delta\boldsymbol{b}} \frac{1}{2}\left(\sum_{i=1}^{N} \|\boldsymbol{z}_i - \boldsymbol{h}(\boldsymbol{\Phi}\bar{\boldsymbol{b}}) - \mathbf{H}_i\boldsymbol{\Phi}_i\delta\boldsymbol{b}\|^2_{\mathbf{R}_i} + \|\bar{\boldsymbol{b}} + \delta\boldsymbol{b} - \boldsymbol{\mu}\|^2_{\mathbf{P}}\right).$$

The solution is given by

$$\begin{aligned} \delta \boldsymbol{b}^* &= \mathbf{A}^{-1}\boldsymbol{g}, \\ \mathbf{A} &= \sum_{i=1}^{N} \boldsymbol{\Phi}_i^\top \mathbf{H}_i^\top \mathbf{R}_i^{-1} \mathbf{H}_i \boldsymbol{\Phi}_i + \mathbf{P}^{-1}, \\ \boldsymbol{g} &= \sum_{i=1}^{N} \boldsymbol{\Phi}_i^\top \mathbf{H}_i^\top \mathbf{R}_i^{-1} \left(\boldsymbol{z}_i - \boldsymbol{h}(\Phi_i \bar{\boldsymbol{b}})\right) + \mathbf{P}^{-1} \left(\bar{\boldsymbol{b}} - \boldsymbol{\mu}_b\right). \end{aligned} \tag{4.18}$$

We update the model parameters $\bar{\boldsymbol{b}} \leftarrow \bar{\boldsymbol{b}} + \delta \boldsymbol{b}^*$, then update all the matrices and vectors in (4.16) and repeat the procedure predefined number of iterations or until convergence. The described approach is known as Gauss-Newton method to solve non-linear least squares problems, and it is used in Tong et al. (2013); Barfoot et al. (2014a). It does not guarantee convergence, so in this work we apply Levenberg-Marquardt approach. It modifies the system

$$\delta \boldsymbol{b}^* = \left(\mathbf{A} + \lambda \mathrm{diag}\left(\mathbf{A}\right)\right)^{-1} \boldsymbol{g} \tag{4.19}$$

where λ is a dampening parameter. The overall update procedure is summarized in Algorithm 3.

The size of the system matrix $\mathbf{A}$ is $(Dd{+}2M){\times}(Dd{+}2M)$ for two-dimensional landmarks. The top-left block of size $Dd \times Dd$ of the matrix corresponds to the weights $\boldsymbol{b}$ and is typically dense. The bottom-right block of size $2M \times 2M$ corresponds to landmarks and it is usually diagonal (because we assume that landmarks are independent). Therefore, the cost of solving (4.18) is $\mathcal{O}(D^3d^3 + MDd)$ using Schur complement. However, we use iterative solver and in practice it converges much faster. The cost of construction of the matrices in (4.18) is $\mathcal{O}(N(D^2d^2 + M)$. The total complexity is $\mathcal{O}(\max(ND^2d^2 + NM), (D^3d^3 + MDd))$.

When we use the iterative solver that utilizes only matrix-vector products, we can also rearrange operations to obtain different computational complexity. Instead of calculating the matrix $\mathbf{A}$ explicitly we can multiply each term of the sum in (4.18) by a vector and then take the sum. Taking into account that matrices $\mathbf{R}_i$ are (usually) diagonal, the part of the Jacobi matrix $\mathbf{H}_i$ that corresponds to derivatives of w.r.t landmarks is block-diagonal, the complexity of matrix-vector product for one term in the sum is $\mathcal{O}((M + D)d)$. The overall complexity of solving the system is $\mathcal{O}(N(M + D)dk)$, where k is the number of iterations.

4.3.2.1 State prior

In GP regression it is common make zero-mean assumption, i.e. $\boldsymbol{\mu}(t) = \mathbf{0}$. However, having good prior $\boldsymbol{\mu}(t)$ is essential when modeling trajectory with GP, because usually shift-invariant kernel functions are used. Therefore, GP with such kernels is most suited to model stationary functions. This does not always apply to trajectories. Non-stationarity can be accounted by non-zero mean function $\boldsymbol{\mu}(t)$. Here we use one of two prior mean functions.

1. Motion model
$$\boldsymbol{\mu}_x(t_i) = \mathbf{F}(t_i)\hat{\boldsymbol{x}}(t_{i-1}) + \mathbf{B}(t_i)\boldsymbol{u}(t_i),$$
where $\mathbf{F}(t), \mathbf{B}(t)$ are time-dependent system matrices. We use this model if we have odometry measurements.

2. Smoothing splines applied to the estimated trajectory with smoothing parameter 0.98 (we used De Boor's formulation, see De Boor (2001)). We also use weights that are inverse proportional to the data fit error $\|\boldsymbol{z}_i - \boldsymbol{h}(\boldsymbol{\Phi}_i \boldsymbol{b})\|_{\mathbf{R}_i}$. The motivation behind this prior mean model is the following: in case of some non-stationarity the GP can produce inaccurate predictions (for example, there can be oscillations in regions where the function quickly changes its value). Smoothing the trajectory reduces such effects.

With a non-zero prior mean for the trajectory we can set all mean vectors $\boldsymbol{\mu}_b^{(i)}$ to zero, thus, the GP will only correct the errors of the mean $\boldsymbol{\mu}(t)$. The whole trajectory estimate is updated with every new measurement, so we also update the prior $\boldsymbol{\mu}_x(t_i)$ for all $i = 1, \ldots, N$ for each new observation.

Algorithm 3 Update state at measurement times

1: Initial values $\bar{\boldsymbol{b}}$, measurement times $t_1, \ldots, t_N$, measurements $\mathbf{z}$, ε, maximum number of iterations K
2: $n \leftarrow 0$
3: **repeat**
4: Using $\bar{\boldsymbol{b}}$ update vectors and matrices in (4.16)
5: Calculate update $\delta\boldsymbol{b}^*$ by applying (4.19) to solve (4.18)
6: $\bar{\boldsymbol{b}} \leftarrow \bar{\boldsymbol{b}} + \delta\boldsymbol{b}^*$
7: $n \leftarrow n + 1$
8: **until** relative error is less than ε **or** $n = K$

4.3.3 Experiments

In this section, we evaluate our approach on several synthetic 2D trajectories as well as real-world benchmarks. In all our experiments, we consider the state vector to be a 2D

pose:

$$\boldsymbol{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \alpha(t) \end{bmatrix}.$$

We use the range/bearing observation model which takes the form

$$\boldsymbol{h}\left(\begin{bmatrix} \boldsymbol{x}(t_i) \\ \boldsymbol{l}_j \end{bmatrix}\right) = \begin{bmatrix} \sqrt{(x_j - x(t_i))^2 + (y_j - y(t_i)^2} \\ \text{atan2}(y_j - y(t_i), x_j - x(t_i)) - \alpha(t_i) \end{bmatrix}, \tag{4.20}$$

where $\boldsymbol{l}_j = \begin{bmatrix} x_j \\ y_j \end{bmatrix}$ is a vector of coordinates of j-th landmark. The covariance matrices $\mathbf{R}_j$ are given and typically determined by the precision of the sensor. In some of the experiments we consider only range measurements (the first row in measurement model), in some of the experiments we have only bearing measurements (the second row in the measurement model) and in other experiments we have both range and bearing measurements. The proposed approach is compared against model based on linear time-variant stochastic differential equation (LTV SDE) Barfoot et al. (2014a)[3].

To evaluate the quality of the estimated trajectories, we calculate two metrics.

- Absolute Pose Error (APE). This metric estimates global consistency of the trajectory. It is based on the relative pose on the estimated trajectory and ground truth trajectory:

  $$e_i^{abs} = \widehat{\mathbf{P}}_i \ominus \mathbf{P}_i = (\mathbf{P}_i)^{-1} \widehat{\mathbf{P}}_i, \quad \mathbf{P}_i, \widehat{\mathbf{P}}_i \in SE(3),$$

  where $\mathbf{P}_i, \widehat{\mathbf{P}}_i$ are ground truth pose and estimated pose at time step t_i represented by an element from $SE(3)$ group of rigid body transformations (translation and rotation). We represent 2D points as 3D point by adding zero z-coordinate and zero roll and pitch angles. Then we can calculate the translational error

  $$\text{APE}_{trans} = \sqrt{\frac{1}{N} \|trans(e_i^{abs})\|_2^2},$$

  where $trans(e)$ is a translational part of e. And we can calculate the rotational error

  $$\text{APE}_{rot} = \sqrt{\frac{1}{N} \|rot(e_i^{abs})\|_2^2},$$

  where $rot(e)$ is a rotational part of e.

[3]The implementation was taken from https://github.com/gtrll/gpslam

- Relative Pose Error (RPE). This metric estimates the local consistency of the trajectory. It is invariant to drifts, i.e., if we translate and rotate the whole trajectory the RPE error will remain the same. RPE is based on the relative difference of the poses on the estimated trajectory and ground truth trajectory:

$$e_i^{rel} = \hat{\delta}_i \ominus \delta_i = \left((\mathbf{P}_{i-1})^{-1}\mathbf{P}_i\right)^{-1}\left(\left(\widehat{\mathbf{P}}_{i-1}\right)^{-1}\widehat{\mathbf{P}}_i\right)$$
$$\mathbf{P}_i, \widehat{\mathbf{P}}_i \in SE(3),$$

where $\mathbf{P}_i, \widehat{\mathbf{P}}_i$ are as in APE. Similarly to APE, we calculate translational and rotational errors

$$\text{RPE}_{trans} = \sqrt{\frac{1}{N}\|trans(e_i^{rel})\|_2^2},$$

$$\text{RPE}_{rot} = \sqrt{\frac{1}{N}\|rot(e_i^{rel})\|_2^2}.$$

4.3.3.1 Synthetic trajectories

We generated 10 different random trajectories, for each trajectory we conducted several experiments with different noise level in observations, different number of landmarks (from 5 to 100) and different measurement types (range, bearing, range/bearing). The noise was generated from Gaussian distribution with standard deviation varying in $[1, 5]$ interval for range measurements and bearing varying in $[1°, 10°]$ interval.

Number of features D For the synthetic dataset we conducted experiments with different number of features. We observed that for a small number of features ($D \sim 10$) the trajectory starts diverging when its length increases (at about 100 observations). Increasing the number of features increases the length of the trajectory for which the estimate does not diverge. For the trajectories that we used in our experiments $D = 100$ was enough to obtain good state estimates.

Kernel parameters. The main kernel parameter is its lengthscale σ_l. Typically, in the GP regression model the lengthscale is one of the most important parameters affecting the quality of the model. It controls the smoothness of the obtained approximation. Larger lengthscale should be used for smooth trajectories and smaller values for less smooth trajectories.

In our experiments a rather wide kernel worked well, we set $\sigma_l = 3.0$. The qualitative results can be found in Table 4.4. We can see that in the case of range and range-bearing measurements the proposed approach looks more accurate.

TABLE 4.4: Relative Errors for synthetic trajectories

		Pos.	Rot.	Landmarks
RangeBearing	RFF	0.022	**0.154**	6e-4
	LTV SDE	0.025	5.602	0.110
Range	RFF	**0.016**	**0.320**	1e-6
	LTV SDE	0.025	5.580	0.003
Bearing	RFF	0.035	**0.152**	8e-6
	LTV SDE	**0.025**	1.200	0.016

We also study the dependency of the estimation error on the noise level and the number of landmarks. In Figure 4.7 you can see the APE translation errors for different noise levels, the number of landmarks and measurement types. We make several observations based on these results.

- Our approach does not estimate bearing in the range-only measurements because there is no information about bearing in the data. In this case we calculate heading by calculating the movement direction of the estimated trajectory. Barfoot's method handle this situation due to their mean prior based on the differential equation.
- The proposed approach provides better rotation errors in all cases.
- The translation errors in range only setting and rotation errors of RFF approach in bearing only measurements increase slower with noise level compared to LTV SDE errors. For range-bearing case the difference is less significant.

4.3.3.2 Autonomous Lawn-Mower

In this experiment we evaluate our approach on a Plaza data set collected from an autonomous lawn-mower Djugash (2010). The data set contains range measurements recorded using time-of-flight and odometry measurements. Odometry measurements come more frequently than range measurements. The ground truth data was collected from GPS measurements and according to Djugash (2010) its accuracy is 2cm.

The resulting trajectory is given in Figure 4.8. In this experiment we did batch updates, i.e. we updated the trajectory after each new 5 range measurements. The motion model based prior was used as we have odometry measurements. We can see slight oscillations of the estimated trajectory. They can be explained by the nature of the Fourier features. However, the errors are comparable as can be seen from Table 4.5. The estimated trajectories are depicted in Figure 4.8.

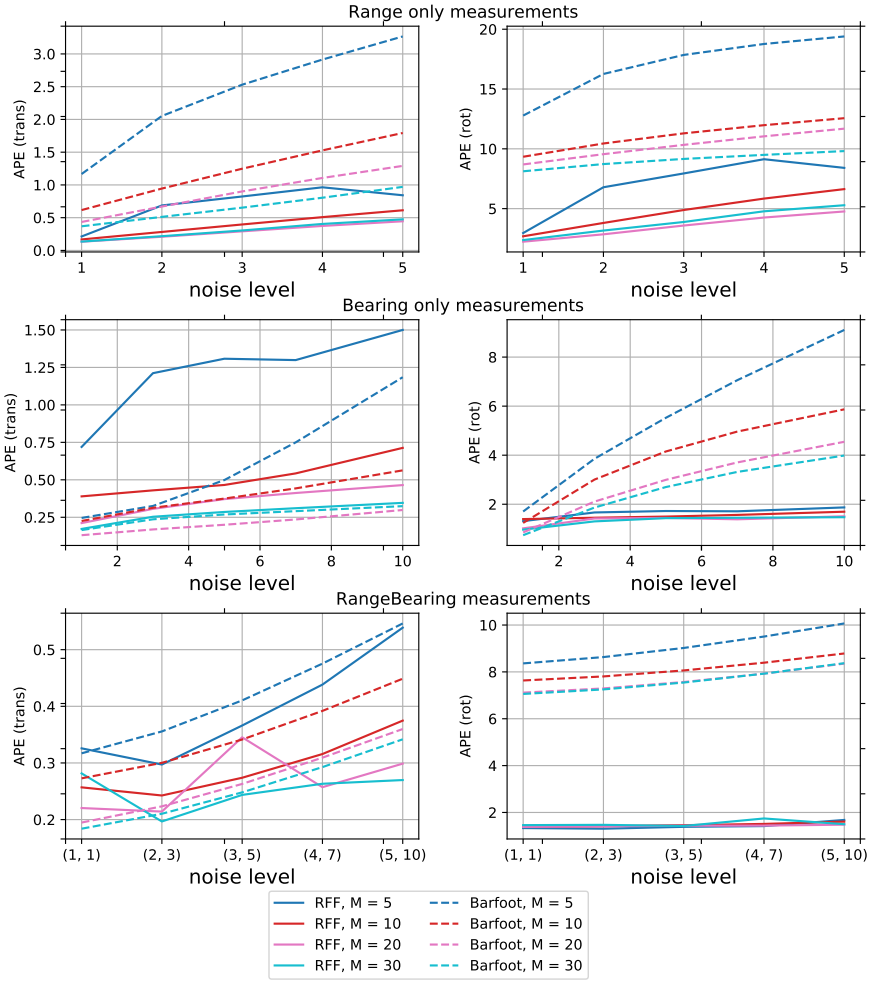


FIGURE 4.7: Average APE errors for synthetic trajectories at different noise levels and number of landmarks

4.3.3.3 KITTI-projected dataset

To evaluate our approach, we proceed with the world-famous dataset KITTI odometryGeiger et al. (2013). We chose the dataset part that contains stereo sequences – a sequence of stereo images taken while moving along a specific trajectory. The dataset includes stereo images, ground truth trajectory and camera information. For our approach, we need a dataset in 2D with observations and bearing. To extract observations in the form (4.20) we do visual SLAM from ORB-SLAM2 Mur-Artal & Tardós (2017). For each keypoint we find its coordinates in the local frameHartley & Zisserman (2003). Therefore, we can calculate bearing observations for each keypoint. Thus, the keypoints in this case play the role of landmarks and we have bearing measurements. The pipeline to project KITTI into a 2D dataset is following:

- Input: KITTI-odometry dataset (e.g. sequence 08);

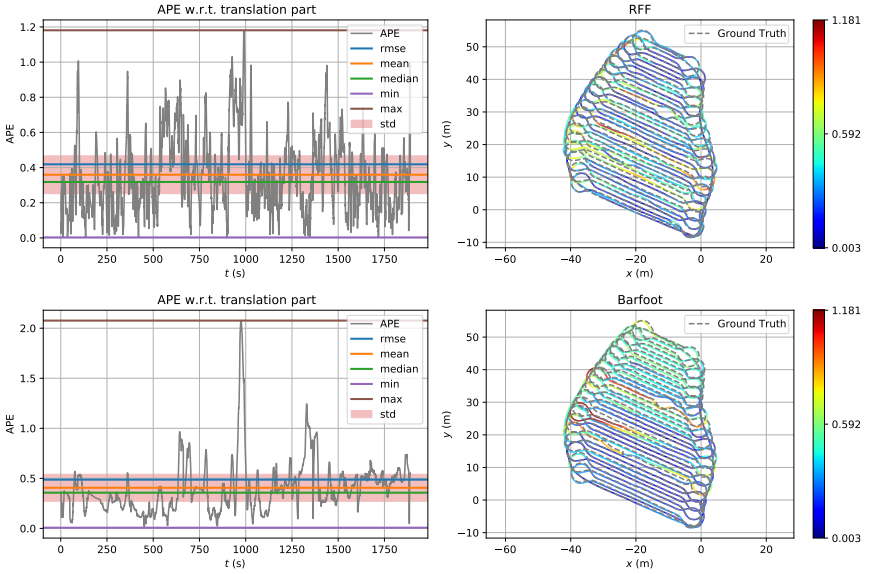


FIGURE 4.8: Distribution of APE errors along the trajecotry for Autonomous Lawn-Mower benchmark

- Run ORB-SLAM2 to get observations (keypoints, timestamps);
- Correct camera pose and keypoints by a transformation term that aligns with the vertical axis to make them independent of their original camera pose

$$R_t \cdot R_{correction} = \text{Exp}([0, 0, \alpha]^\top); \tag{4.21}$$

- Calculate weights for each observations based on how much time this point was observed/visited;
- Filter observations;
- Do orthonormal projection for each observation;
- Calculate bearing.

The number of landmarks (keypoints) is 80771. With such a big number of landmarks, the experiments are very slow, so we reduced the number of landmarks to 3975. We selected the landmarks randomly with probabilities proportional to their weights, but for each keyframe we left not less than 10 landmarks. Also, to speed up the calculations we split the trajectory into 10 consecutive slices, do estimation on each slice independently and then average the estimation errors.

The extracted data we then use in our approach to estimate the trajectory. To check our assumption that kernels with dense precision matrices should work better in case of

noisy observations, we also generated a noisy version of the KITTI-projected dataset. To do so, we added Gaussian noise with standard deviation σ.

The results are given in Table 4.5. We can see that without additional noise the results are comparable (with RFF based approach being slightly better). When we increase the amount of noise, absolute errors of our approach remains the same while errors for LTV SDE increase.

TABLE 4.5: Real-world benchmark errors.

Autonomous Lawn-Mower

	APE (trans)	APE (rot)	RPE (trans)	RPE (rot)
LTV SDE	0.48	1.44	0.021	0.10
RFF	0.42	2.25	0.026	0.54
KITTI-projected				
LTV SDE	5.130	1.059	0.068	0.113
RFF	5.070	0.489	0.040	0.048
KITTI-projected + noise, $\sigma = 1°$				
LTV SDE	5.126	1.086	0.068	0.133
RFF	5.070	0.544	0.0454	0.052
KITTI-projected + noise, $\sigma = 3°$				
LTV SDE	5.491	3.136	0.139	0.259
RFF	5.075	1.027	0.073	0.115
KITTI-projected + noise, $\sigma = 5°$				
LTV SDE	12.915	5.114	0.242	0.358
RFF	5.077	1.304	0.084	0.119

Chapter 5

Conclusion

This thesis contributes to the Gaussian process models and kernel methods as well as different machine learning techniques.

In Chapter 2, we considered regression problems where the data set has either the full or incomplete factorial design of experiments. Using the special data set's structure, we developed a computationally efficient technique for drawing the exact inference of the GP model. We also provided a special regularization for such cases that allows the avoidance of the degeneration of the model and improves the overall quality. Lastly, the experimental section in this chapter demonstrated the good performance of the approach and justified low computational complexity.

Chapter 3 developed a general method for approximating the kernel function based on randomized feature maps. For this, we proposed a quadrature-based approach to build such feature maps, which allowed us to obtain a low approximation error with a small number of features thus reducing the computational complexity. In addition, the theoretical analysis provided error bounds for the developed technique. The subsequent experiments showed the superiority of the method compared with other random feature-based approaches.

Chapter 4 is dedicated to applications of the developed methods, which considered three different problems: tensor completion, probability density estimate, and simultaneous localization and mapping. For the tensor completion problem, we developed an initialization scheme based on the GP model. We supposed that the function that generates tensor values is smooth and hence modelled it by combining the GP model with the tensor-train-based approximation method. As a result, we developed a general tensor completion approach. The subsequent experiments showed that the proposed initialization improves the overall quality.

We also applied the random features approach to the density estimate problem. In this case, we derived an analytical solution for the denoising score matching loss, which was impossible to derive using the exact GP model or Nyström-type approximations. Therefore, the resulting model has natural additional regularization. The performance also improved compared with the Nyström-based models.

Finally, the developed large-scale GP models allowed to re-utilize it in simultaneous localization and mapping problems. In robotics, the state-space formulation of the GP model and a restricted class of kernel functions that give a high computational performance are used. With the proposed approach, we not only kept a low computational complexity but also enjoyed the advantages of a broader class of kernel that we can could apply. Moreover, we demonstrated that it can improve the quality of the estimates in some cases, especially when there is a considerable amount of noise in observations.

Currently, deep neural networks outperform other techniques in many problems. Interestingly, there is a connection between neural networks and kernel methods. One of the first works in this direction was (Williams, 1996), which showed that a single-layer network with infinite neurons is equivalent to the Gaussian process. More modern works (Lee et al., 2017; Jacot et al., 2018) show the connection between deeper neural networks, Gaussian processes and kernels. The work (Daniely, 2017) studies the behaviour of neural networks with all the weight randomized except the last one, which is learnt using SGD. This can be seen as a random feature model.

In light of the mentioned works, it is interesting to study how the properties of random feature models (and kernels) can be transferred to deep neural networks in practice. For example, the double descent phenomenon is encountered in over-parameterized neural networks; although in some architectures, it is impossible to observe it (Ba et al., 2019), in random feature models, it seems to be more robust (Mei & Montanari, 2019). Therefore, the question is whether the random features idea could help develop over-parameterized architectures with better double-descent guarantees.

It is also interesting to study whether random feature models can be helpful in neural network compression. In the paper (Frankle & Carbin, 2018), the authors introduce the Lottery Ticket Hypothesis, which states that a network contains a smaller sub-network, which can achieve the same performance as a whole network when trained in isolation. In context of the hypothesis, shallow neural sub-networks are equivalent to random feature models (Malach et al., 2020). Thus, further studying this connection and finding a way to apply the random features idea to compress neural networks might be a promising research direction.

Finally, I believe that in the existing pipelines, there are parts that can be efficiently approximated using randomized maps. For example, in the recent work (Choromanski et al., 2020), they applied the random features approach to approximate an attention mechanism. Also, replacing some traditional layers or computations with a more complicated kernel version and then reducing the complexity using some kind of approximation could yield results.

Appendix A

Additional results for quadrature-based features

A.1 Proof of Proposition 3.6

A.1.1 Variance of the degree $(3,3)$ quadrature rule

Let us denote $\mathbf{q} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \in \mathcal{X}^2$, $k(\mathbf{q}) = k(\mathbf{x}, \mathbf{y})$, $h_j(\mathbf{q}) = d\frac{f_{\mathbf{xy}}(-\rho_j \mathbf{Q}\mathbf{v}_j) + f_{\mathbf{xy}}(\rho_j \mathbf{Q}\mathbf{v}_j)}{2\rho_j^2} - k(\mathbf{q}) = s_j(\mathbf{q}) - k(\mathbf{q})$. Then, it is easy to see that $\mathbb{E}h_j(\mathbf{q}) = 0$.

Let us denote $I(\mathbf{q}) = SR^{3,3}_{\mathbf{Q}_1,\rho_1}(f_{\mathbf{xy}})$, $g(\mathbf{q}) = I(\mathbf{q}) - k(\mathbf{x}, \mathbf{y})$. Using the above definitions we obtain

$$\begin{aligned} \mathbb{V}g(\mathbf{q}) = \mathbb{V}\left(1 - \sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2}\right) + \mathbb{E}\left(\frac{1}{d+1}\sum_{i=1}^{d+1} h_i(\mathbf{q})\right)^2 \\ +2cov\left(1 - \sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2}, \frac{1}{d+1}\sum_{i=1}^{d+1} h_i(\mathbf{q})\right). \end{aligned} \tag{A.1}$$

Variance of the first term

$$\begin{aligned} \mathbb{V}\left(1 - \sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2}\right) &= \mathbb{E}\left(1 - \sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2}\right)^2 \\ &= \mathbb{E}\left(1 - \sum_{j=1}^{d+1} \frac{2d}{(d+1)\rho_j^2} + \left(\sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2}\right)^2\right) \\ &= 1 - 2 + \frac{d}{(d+1)(d-2)} + \frac{d}{d+1} = \frac{2}{(d+1)(d-2)}. \end{aligned} \tag{A.2}$$

Variance of the second term (using independence of $h_i(\mathbf{q})$ and $h_j(\mathbf{q})$ for $i \neq j$)

$$\mathbb{E}\left(\frac{1}{d+1}\sum_{i=1}^{d+1} h_i(\mathbf{q})\right)^2 = \mathbb{E}\left(\frac{1}{(d+1)^2}\sum_{i,j=1}^{d+1} h_i(\mathbf{q})h_j(\mathbf{q})\right) = \frac{1}{(d+1)^2}\sum_{i=1}\mathbf{E}h_i(\mathbf{q})^2 = \frac{\mathbf{E}h_1(\mathbf{q})^2}{d+1}. \tag{A.3}$$

Variance of the last term (using the Cauchy-Schwarz inequality)

$$\begin{aligned} cov\left(1 - \sum_{j=1}^{d+1}\frac{d}{(d+1)\rho_j^2}, \frac{1}{d+1}\sum_{i=1}^{d+1} h_i(\mathbf{q})\right) &= \mathbb{E}\left[\left(1 - \sum_{j=1}^{d+1}\frac{d}{(d+1)\rho_j^2}\frac{1}{d+1}\right)\sum_{i=1}^{d+1} h_i(\mathbf{q})\right] \\ &= -\mathbb{E}\frac{d}{d+1}\sum_{i,j=1}^{d+1}\frac{h_i(\mathbf{q})}{\rho_j^2} \\ &\leq \frac{1}{d+1}\sum_{i=1}^{d+1}\sqrt{\mathbb{E}\frac{1}{\rho_i^4}}\sqrt{\mathbb{E}h_i(\mathbf{q})^2} \\ &= \sqrt{\frac{\mathbb{E}h_1(\mathbf{q})^2}{d(d-2)}}. \end{aligned} \tag{A.4}$$

Now, let us upper bound term $\mathbb{E}h_1(\mathbf{q})^2$

$$\mathbb{E}h_1(\mathbf{q})^2 = \mathbb{E}\left(\frac{d\phi(\mathbf{w}^\top\mathbf{x})\phi(\mathbf{w}^\top\mathbf{y})}{\rho^2}\right)^2 - k(\mathbf{q})^2 \leq \frac{d\kappa^4}{d-2}.$$

Using this expression and plugging (A.2), (A.3), (A.4) into (A.1), we obtain

$$\begin{aligned} \mathbb{V}\left[\frac{1}{n}\sum_{i=1}^{n} SR^{3,3}_{\mathbf{Q}_i,\rho_i}(f_{\mathbf{xy}})\right] &\leq \frac{2}{n(d+1)(d-2)} + \frac{d\kappa^4}{n(d+1)(d-2)} + \frac{1}{n}\sqrt{\frac{d\kappa^4}{d(d-2)^2}} \leq \\ &\leq \frac{2}{n(d+1)(d-2)} + \frac{d\kappa^4}{n(d+1)(d-2)} + \frac{\kappa^2}{n(d-2)} \leq \frac{2+\kappa^4+\kappa^2}{n(d-2)}, \end{aligned} \tag{A.5}$$

thereby concluding the proof.

A.1.2 Error probability

The proof strategy closely follows that of (Sutherland & Schneider, 2015); we use the Chebyshev-Cantelli inequality instead of Hoeffding's and Bernstein inequalities and then calculate all the expectations according to our quadrature rules.

Let $\mathbf{q} = \begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} \in \mathcal{X}^2$, where $\mathcal{X}^2$ is a compact set in $\mathbb{R}^{2d}$ with diameter $\sqrt{2}l$, so we can cover it with an ε-net using $T = (2\sqrt{2}l/r)^{2d}$ balls of radius r at most. Let $\{\mathbf{q}_i\}_{i=1}^T$ denote

their centers, and L_g be the Lipschitz constant of $g(\mathbf{q}) : \mathbb{R}^{2d} \to \mathbb{R}$. If $|g(\mathbf{q}_i)| < \varepsilon/2$ for all i and $L_g < \varepsilon/(2r)$, then $g(\mathbf{q}) < \varepsilon$ for all $\mathbf{q} \in \mathcal{X}^2$.

A.1.2.1 Regularity condition

Similar to (Sutherland & Schneider, 2015) (regularity condition section in appendix), it can be proven that $\mathbb{E}\nabla g(\mathbf{q}) = \nabla \mathbb{E} g(\mathbf{q})$.

A.1.2.2 Lipschitz constant

Since g is differentiable, $L_g = \|\nabla g(\mathbf{q}^*)\|$, where $\mathbf{q}^* = \arg\max_{\mathbf{q}\in\mathcal{X}^2} \|\nabla g(\mathbf{q})\|$. Via Jensen's inequality, $\mathbb{E}\|\nabla h(\mathbf{q})\| \geq \|\mathbb{E}\nabla h(\mathbf{q})\|$. Then, using the independence of $h_i(\mathbf{q})$ and $h_j(\mathbf{q})$ for $i \neq j$

$$\begin{aligned}
\mathbb{E}[L_g]^2 = \mathbb{E}\left[\|\nabla I(\mathbf{q}^*) - k(\mathbf{q}^*)\|^2\right] = \mathbb{E}\left[\left\|\frac{1}{d+1}\sum_{i=1}^{d+1}\nabla h_i(\mathbf{q}^*)\right\|^2\right] = \mathbb{E}\left[\frac{1}{d+1}\|\nabla h_1(\mathbf{q}^*)\|^2\right] = \\
= \frac{1}{d+1}\mathbb{E}_{\mathbf{q}^*}\left[\mathbb{E}\|\nabla s_1(\mathbf{q}^*)\|^2 - 2\|\nabla k(\mathbf{q}^*)\|\mathbb{E}\|\nabla s_1(\mathbf{q}^*)\| + \|\nabla k(\mathbf{q}^*)\|^2\right] \leq \\
\leq \frac{1}{d+1}\mathbb{E}\left[\|\nabla s_1(\mathbf{q}^*)\|^2 - \|\nabla k(\mathbf{q}^*)\|^2\right] \leq \frac{1}{d+1}\mathbb{E}\|\nabla s_1(\mathbf{q}^*)\|^2 = \\
= \frac{1}{d+1}\mathbb{E}\left[\|\nabla_{\mathbf{x}^*} s_1(\mathbf{q}^*)\|^2 + \|\nabla_{\mathbf{y}^*} s_1(\mathbf{q}^*)\|^2\right] \leq \frac{2d^2\kappa^2\mu^2\sigma_p^2}{d+1}\mathbb{E}\frac{1}{\rho_1^2} = \frac{2d\kappa^2\mu^2\sigma_p^2}{d+1},
\end{aligned}$$

where $|\phi'(\cdot)| \leq \mu$. Then, using Markov's inequality, we obtain

$$\mathbb{P}(L_g \geq \frac{\varepsilon}{2r}) \leq 8\frac{d}{d+1}\left(\frac{\sigma_p r\kappa\mu}{\varepsilon}\right)^2$$

A.1.2.3 Anchor points

Let us upper bound the following probability as

$$\mathbb{P}\left(\bigcup_{i=1}^{T}|g(\mathbf{q}_i)| \geq \frac{1}{2}\varepsilon\right) \leq T\mathbb{P}\left(|g(\mathbf{q}_i)| \geq \frac{1}{2}\varepsilon\right).$$

Now, let us rewrite the function $g(\mathbf{q})$ as

$$g(\mathbf{q}) = 1 - \frac{1}{d+1}\sum_{i=1}^{d+1}\frac{d}{\rho_i^2} + \frac{1}{d+1}\sum_{i=1}^{d+1}\frac{d\phi_{\mathbf{q}}(\rho_i\mathbf{z}_i)}{\rho_i^2} - k(\mathbf{q}) = \frac{1}{d+1}\sum_{i=1}^{d+1}\left(\frac{d(1-\phi_{\mathbf{q}}(\rho_i\mathbf{z}_i))}{\rho_i^2} + 1 - k(\mathbf{q})\right),$$

where $\phi_{\mathbf{q}}(\rho_i \mathbf{z}_i) = \frac{f_{\mathbf{xy}}(-\rho_j \mathbf{Q}\mathbf{v}_j) + f_{\mathbf{xy}}(\rho_j \mathbf{Q}\mathbf{v}_j)}{2\rho_j^2}$. Next, let us suppose that $\left|\frac{1-\phi_{\mathbf{q}}(\rho \mathbf{z})}{\rho^2}\right| \leq M$. so that we can apply Hoeffding's inequality

$$\mathbb{P}(|g(\mathbf{q})| \geq \frac{1}{2}\varepsilon) \leq 2\exp\left(-\frac{2D\frac{1}{4}\varepsilon^2}{(M-(-M))^2}\right) = 2\exp\left(-\frac{D\varepsilon^2}{8M^2}\right)$$

A.1.2.4 Optimizing over r

Now, the probability of $\sup_{\mathbf{q}\in\mathcal{X}^2} |g(\mathbf{q})| \leq \varepsilon$ takes the form

$$p = \mathbb{P}\left(\sup_{\mathbf{q}\in\mathcal{X}^2} |g(\mathbf{q})| \leq \varepsilon\right) \geq 1 - \kappa_1 r^{-2d} - \kappa_2 r^2,$$

where $\kappa_1 = 2\left(2\sqrt{2}l\right)^{2d} \exp\left(-\frac{D\varepsilon^2}{8M^2}\right)$ and $\kappa_2 = \frac{8d}{d+1}\left(\frac{\kappa\mu\sigma_p}{\varepsilon}\right)^2$. Maximizing this probability over r gives us the following bound:

$$\mathbb{P}\left(\sup_{\mathbf{q}\in\mathcal{X}^2} |g(\mathbf{q})| \geq \varepsilon\right) \leq \left(d^{\frac{-d}{d+1}} + d^{\frac{1}{d+1}}\right) 2^{\frac{6d+1}{d+1}} \left(\frac{d}{d+1}\right)^{\frac{d}{d+1}} \left(\frac{\sigma_p l \kappa \mu}{\varepsilon}\right)^{\frac{2d}{d+1}} \exp\left(-\frac{D\varepsilon^2}{8M^2(d+1)}\right).$$

For the RBF kernel, $\kappa = \mu = 1$ and $M = \frac{1}{2}$, so we obtain the following bound:

$$\mathbb{P}\left(\sup_{\mathbf{q}\in\mathcal{X}^2} |g(\mathbf{q})| \geq \varepsilon\right) \leq \left(d^{\frac{-d}{d+1}} + d^{\frac{1}{d+1}}\right) 2^{\frac{6d+1}{d+1}} \left(\frac{d}{d+1}\right)^{\frac{d}{d+1}} \left(\frac{\sigma_p l}{\varepsilon}\right)^{\frac{2d}{d+1}} \exp\left(-\frac{D\varepsilon^2}{2(d+1)}\right).$$

Now, let us compare it with the bound for RFF:

$$\mathbb{P}\left(\sup_{\mathbf{q}\in\mathcal{X}^2} |g(\mathbf{q})| \geq \varepsilon\right) \leq \left(d^{\frac{-d}{d+1}} + d^{\frac{1}{d+1}}\right) 2^{\frac{5d+1}{d+1}} 3^{\frac{d}{d+1}} \left(\frac{\sigma_p l}{\varepsilon}\right)^{\frac{2d}{d+1}} \exp\left(-\frac{D\varepsilon^2}{32(d+1)\alpha'_\varepsilon}\right).$$

A.2 Butterfly matrices

For orthogonal matrix $\mathbf{Q}$ in the quadrature rules, the so-called butterfly matrix is used. As it happens to be a product of the butterfly-structured factors, a matrix of this type conveniently possesses the property of fast multiplication. An example of the butterfly orthogonal matrix with $d = 4$ is

$$\mathbf{B}^{(4)} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_3 & -s_3 \\ 0 & 0 & s_3 & c_3 \end{bmatrix} \begin{bmatrix} c_2 & 0 & -s_2 & 0 \\ 0 & c_2 & 0 & -s_2 \\ s_2 & 0 & c_2 & 0 \\ 0 & s_2 & 0 & c_2 \end{bmatrix} = \begin{bmatrix} c_1c_2 & -s_1c_2 & -c_1s_2 & s_1s_2 \\ s_1c_2 & c_1c_2 & -s_1s_2 & -c_1s_2 \\ c_3s_2 & -s_3s_2 & c_3c_2 & -s_3c_2 \\ s_3s_2 & c_3s_2 & s_3c_2 & c_3c_2 \end{bmatrix}.$$

Definition A.1. Let $c_i = \cos\theta_i$, $s_i = \sin\theta_i$ for $i = 1, \ldots, d-1$ be given. Assume $d = 2^k$ with $k > 0$. Then, an orthogonal matrix $\mathbf{B}^{(d)} \in \mathbb{R}^{d\times d}$ is defined recursively as follows:

$$\mathbf{B}^{(2d)} = \begin{bmatrix} \mathbf{B}^{(d)}c_d & -\mathbf{B}^{(d)}s_d \\ \hat{\mathbf{B}}^{(d)}s_d & \hat{\mathbf{B}}^{(d)}c_d \end{bmatrix}, \quad \mathbf{B}^{(1)} = 1,$$

where $\hat{\mathbf{B}}^{(d)}$ is the same as $\mathbf{B}^{(d)}$ with indexes i shifted by d, e.g.,

$$\mathbf{B}^{(2)} = \begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix}, \quad \hat{\mathbf{B}}^{(2)} = \begin{bmatrix} c_3 & -s_3 \\ s_3 & c_3 \end{bmatrix}.$$

Matrix $\mathbf{B}^{(d)}$ by the vector product has computational complexity $O(d\log d)$ since $\mathbf{B}^{(d)}$ has $\lceil\log d\rceil$ factors and each factor requires $O(d)$ operations. Another advantage is space complexity; $\mathbf{B}^{(d)}$ is fully determined by $d-1$ angles θ_i, yielding $O(d)$ memory complexity.

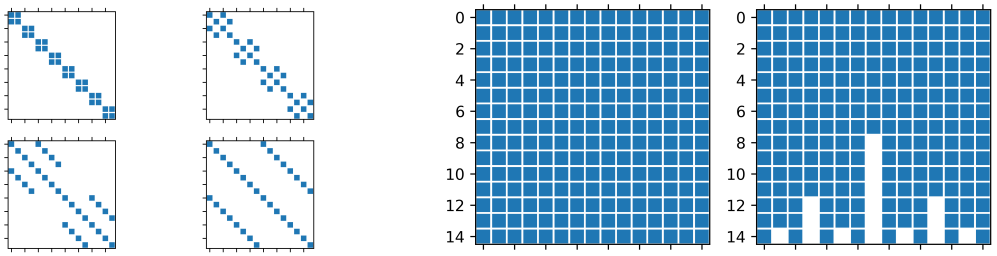
The randomization is based on the sampling of angles θ. We follow the(Fang & Li, 1997) algorithm that first computes a uniform random point $\mathbf{u}$ from U_d. It then calculates the angles by taking the ratios of the appropriate $\mathbf{u}$ coordinates $\theta_i = \frac{u_i}{u_{i+1}}$, followed by computing cosines and sines of the θ's. Consequently, one can easily define the butterfly matrix $\mathbf{B}^{(d)}$ for the cases when d is not a power of two.

A.2.1 Not a power of two

Here, we discuss the procedure to generate butterfly matrices of size $d \times d$ when d is not a power of 2.

Let the number of butterfly factors $k = \lceil\log d\rceil$. Then, $\mathbf{B}^{(d)}$ is constructed as a product of k-factor matrices of size $d \times d$, obtained from the k matrices used for generating $\mathbf{B}^{(2^k)}$. For each matrix in the product for $\mathbf{B}^{(2^k)}$, we delete the last $2^k - d$ rows and columns. Next, we replace with 1 every c_i in the remaining $d \times d$ matrix that is in the same column as the deleted s_i.

For the cases when d is not a power of two, the resulting $\mathbf{B}$ has deficient columns with zeros (Figure A.1b, right), which introduces a bias to the integral estimate. To correct for this bias, one may apply additional randomization using a product $\mathbf{BP}$, where $\mathbf{P} \in \{0,1\}^{d\times d}$ is a permutation matrix. It is even better to use a product of several $\mathbf{BP}$'s: $\widetilde{\mathbf{B}} = (\mathbf{BP})_1(\mathbf{BP})_2 \ldots (\mathbf{BP})_t$. We then set $t = 3$ in the experiments.



(A) (B)

FIGURE A.1: (a) Butterfly orthogonal matrix factors for $d = 16$. (b) Sparsity pattern for **BPBPBP** (left) and **B** (right), where $d = 15$.

A.3 Remarks on quadrature rules

Even functions. We note here that for specific functions $f_{\mathbf{xy}}(\mathbf{w})$, we can derive better versions of the SR rule by using the knowledge about the integrand to our advantage. For example, the Gaussian kernel has $f_{\mathbf{xy}}(\mathbf{w}) = \cos(\mathbf{w}^\top(\mathbf{x} - \mathbf{y}))$. Note that f here is even, so we can discard an excessive term in the summation in the degree $(3, 3)$ rule, since $f(\mathbf{w}) = f(-\mathbf{w})$, i.e., the $SR^{3,3}$ rule reduces to

$$SR^{3,3}_{\mathbf{Q},\rho}(f) = \left(1 - \sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2}\right) f(\mathbf{0}) + \frac{d}{d+1} \sum_{j=1}^{d+1} \frac{f(\rho_j \mathbf{Q}\mathbf{v}_j)}{\rho_j^2}. \tag{A.6}$$

Obtaining a proper $\boldsymbol{\rho}$ It may be the case while sampling ρ that $1 - \sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2} < 0$, which results in a complex a_0 term. In that case, the simple solution is just to resample ρ_j to satisfy the non-negativity of the expression. According to the central limit theorem, $\sum_{j=1}^{d+1} \frac{d}{(d+1)\rho_j^2}$ tends to a normal random variable with mean 1 and variance $\frac{1}{d+1}\frac{2}{d-2}$. The probability that these values are non-negative equals $p = \mathbb{P}(1 - \sum_{j=1} \frac{d}{(d+1)\rho^2} \geq 0) \rightsquigarrow \frac{1}{2}$. The expectation of the number of resamples needed to satisfy the non-negativity constraint is $\frac{1}{p}$ tending to 2.

A.4 Arc-cosine kernels

Arc-cosine kernels were originally introduced by (Cho & Saul, 2009) upon studying the connections between deep learning and kernel methods. The integral representation of the b^{th}-order arc-cosine kernel is

$$k_b(\mathbf{x}, \mathbf{y}) = 2 \int_{\mathbb{R}^n} \Theta(\mathbf{w}^\top \mathbf{x})\Theta(\mathbf{w}^\top \mathbf{y})(\mathbf{w}^\top \mathbf{x})^b (\mathbf{w}^\top \mathbf{y})^b p(\mathbf{w}) d\mathbf{w},$$

$$k_b(\mathbf{x}, \mathbf{y}) = 2 \int_{\mathbb{R}^d} \phi_b(\mathbf{w}^\top \mathbf{x})\phi_b(\mathbf{w}^\top \mathbf{y}) p(\mathbf{w}) d\mathbf{w},$$

where $\phi_b(\mathbf{w}^\top \mathbf{x}) = \Theta(\mathbf{w}^\top \mathbf{x})(\mathbf{w}^\top \mathbf{x})^b$, $\Theta(\cdot)$ is the Heaviside function and p is the density of the standard Gaussian distribution. Such kernels can be seen as an inner product of the representation produced by an infinitely wide single-layer neural network with random Gaussian weights. They also have closed-form expressions in terms of the angle $\theta = \cos^{-1}\left(\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}\right)$ between $\mathbf{x}$ and $\mathbf{y}$.

The arc-cosine kernel of the 0^{th}-order shares the property of mapping the input on the unit hypersphere with RBF kernels, while the order 1 arc-cosine kernel preserves the norm as the linear kernel (Gram matrix on original features):

These expressions for 0^{th}-order and 1^{st}-order arc-cosine kernels are given by

$$k_0(\mathbf{x}, \mathbf{y}) = 1 - \frac{\theta}{\pi}, \qquad k_1(\mathbf{x}, \mathbf{y}) = \frac{\|\mathbf{x}\|\|\mathbf{y}\|}{\pi}(\sin\theta + (\pi - \theta)\cos\theta).$$

The 0-order arc-cosine kernel is given by $k_0(\mathbf{x}, \mathbf{y}) = 1 - \frac{\theta}{\pi}$; the 1-order kernel is given by $k_1(\mathbf{x}, \mathbf{y}) = \frac{\|\mathbf{x}\|\|\mathbf{y}\|}{\pi}(\sin\theta + (\pi - \theta)\cos\theta)$.

Let $\phi_0(\mathbf{w}^\top \mathbf{x}) = \Theta(\mathbf{w}^\top \mathbf{x})$ and $\phi_1(\mathbf{w}^\top \mathbf{x}) = \max(0, \mathbf{w}^\top \mathbf{x})$. We can now rewrite the integral representation as follows:

$$k_b(\mathbf{x}, \mathbf{y}) = 2\int_{\mathbb{R}^d} \phi_b(\mathbf{w}^\top \mathbf{x})\phi_b(\mathbf{w}^\top \mathbf{y})p(\mathbf{w})d\mathbf{w} \approx \frac{2}{n}\sum_{i=1}^{n} SR^{3,3}_{\mathbf{Q}_i,\boldsymbol{\rho}_i}.$$

For an arc-cosine kernel of the order 0, the value of the function $\phi_0(0) = \Theta(0) = 0.5$ results in

$$SR^{3,3}_{\mathbf{Q},\rho}(f) = 0.25\left(1 - \sum_{j=1}^{d+1}\frac{d}{(d+1)\rho_j^2}\right) + \frac{d}{d+1}\sum_{j=1}^{d+1}\frac{f(\rho_j \mathbf{Q}\mathbf{v}_j) + f(-\rho_j \mathbf{Q}\mathbf{v}_j)}{2\rho^2}.$$

Whereas in the case of an arc-cosine kernel of the order 1, the value of $\phi_1(0)$ is 0, so the $SR^{3,3}$ rule reduces to

$$SR^{3,3}_{\mathbf{Q},\rho}(f) = \frac{d}{d+1}\sum_{j=1}^{d+1}\frac{f(|\rho \mathbf{Q}\mathbf{v}_j|)}{2\rho_j^2}.$$

Appendix B

Additional results for score matching

B.1 Technical results

B.1.1 Exact solution for the kernel denoising score matching with RFF

We start with the first-order optimality condition:

$$\int p_\varepsilon(\mathbf{y})A^*(\mathbf{y})\nabla V(A(\mathbf{y})f)d\mathbf{y} + \lambda f = 0,$$

where $A^*(\mathbf{y}) : \mathbb{R}^m \to \mathcal{H}$ is an adjoint to $A(\mathbf{y})$ and $A^*(\mathbf{y})\boldsymbol{\alpha} = \sum_{i=1}^m \alpha_i\phi_i(\mathbf{y}, \cdot)$. Denoting

$$\boldsymbol{\alpha}(\mathbf{y}) = -\frac{1}{\lambda}\nabla V(A(\mathbf{y})f), \quad f = \int p_\varepsilon(\mathbf{y})A^*(\mathbf{y})\boldsymbol{\alpha}(\mathbf{y})d\mathbf{y},$$

the first-order optimality condition could be rewritten as an integral equation on $\boldsymbol{\alpha}(\mathbf{y})$:

$$\boldsymbol{\alpha}(\mathbf{y}) = -\frac{1}{\lambda}\nabla V\left(\int p_\varepsilon(\mathbf{z})A(\mathbf{y})A^*(\mathbf{z})\alpha(\mathbf{z})d\mathbf{z}\right), \tag{B.1}$$

where $A(\mathbf{y})A^*(\mathbf{z})\alpha(\mathbf{z}) = \sum\limits_{i=1}^m \alpha_i(\mathbf{z})\{\langle\phi_j(\mathbf{y}, \cdot), \phi_i(\mathbf{z}, \cdot)\rangle\}_{j=1}^m = \boldsymbol{K}(\mathbf{y}, \mathbf{z})\boldsymbol{\alpha}(\mathbf{z})$.

The gradient of V is given by $\nabla V = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}, 1)^\top$. Then, we have $\boldsymbol{\alpha}(\mathbf{y}) = (\boldsymbol{\beta}^\top(\mathbf{y}), \delta)^\top$, where $\delta = -\frac{1}{\lambda}$. Therefore, the integral equation on $\boldsymbol{\beta}(\mathbf{y})$ can be expressed as

$$\boldsymbol{\beta}(\mathbf{y}) = -\frac{1}{n\lambda}\int p_\varepsilon(\mathbf{z})\hat{A}(\mathbf{y})\hat{A}(\mathbf{z})^*\boldsymbol{\beta}(\mathbf{z})d\mathbf{z} + \frac{1}{n\lambda^2}\int p_\varepsilon(\mathbf{z})\hat{A}(\mathbf{y})\phi_m(\mathbf{z}, \cdot)d\mathbf{z}, \tag{B.2}$$

where $(\hat{A}(\mathbf{y})f)_i = (A(\mathbf{y})f)_i$, $i = 1, \dots, m-1$. Let $b = \int p_\varepsilon(\mathbf{y})\phi_m(\mathbf{y}, \cdot)d\mathbf{y}$ and $C = \int p_\varepsilon(\mathbf{y})\hat{A}^*(\mathbf{y})\boldsymbol{\beta}(\mathbf{y})d\mathbf{y}$. Next, we search for the solution of (4.9) in the form $\boldsymbol{\beta}(\mathbf{y}) = -\frac{1}{n\lambda}\hat{A}(\mathbf{y})C + \frac{1}{n\lambda^2}\hat{A}(\mathbf{y})b$. In this case, we have

$$\hat{A}(\mathbf{y})\left[C + \frac{1}{n\lambda}BC - \frac{1}{n\lambda^2}Bb\right] = 0,$$

where $B = \int p_\varepsilon(\mathbf{y})\hat{A}^*(\mathbf{y})\hat{A}(\mathbf{y})d\mathbf{y}$ and $b \in \mathcal{H}$ is a convolution of ϕ_m and noise density p_ε.

Solution C^* of the above equation provides $\boldsymbol{\beta}^*(\mathbf{y})$ and, as a result, the solution to the initial problem. Let us show that the obtained estimator belongs to H. In fact, since

$$C + \frac{1}{n\lambda}BC - \frac{1}{n\lambda^2}Bb \in \mathrm{Ker}\hat{A}(\mathbf{y}) \subseteq \mathcal{H}$$

and $B + n\lambda I$ is continuously invertible, we have that $C^* \in \mathcal{H}$. Finally, we have

$$f^* = B\left[-\frac{1}{n\lambda}C^* + \frac{1}{n\lambda^2}b\right] = C^* - \frac{1}{\lambda}b - \gamma \in \mathcal{H},$$

where we assume $\gamma \in \mathrm{Ker}\hat{A}(\mathbf{y})$.

B.1.2 RFF solution derivation

Let us use the expressions for the solution without noise (for simplicity, the term with $\partial_i \log q_0(\mathbf{x}_a)$ is omitted here):

$$\left[\hat{A}(\mathbf{0})\hat{A}(\mathbf{0})^*\right]_{(a-1)d+i,(b-1)d+j} = \partial_i\partial_{j+d}k(\mathbf{x}_a, \mathbf{x}_b), \quad a, b \in [n],\ i, j \in [d],$$

$$\left[\hat{A}(\mathbf{0})\boldsymbol{\phi}_m(\mathbf{0}, \cdot)\right]_{(a-1)d+i} = \frac{1}{n}\sum_{b,j=1}^{n,d}\partial_i\partial^2_{j+d}k(\mathbf{x}_a, \mathbf{x}_b), \quad a \in [n],\ i \in [d].$$

Then we have $\hat{A}(\mathbf{y})\hat{A}(\mathbf{z})^* \approx \partial\boldsymbol{\Phi}_y\partial\boldsymbol{\Phi}_z^\top$ and $\hat{A}(\mathbf{y})\boldsymbol{\phi}_m(\mathbf{z}, \cdot) * p_\varepsilon(\mathbf{z}) \approx \frac{1}{n}\partial\boldsymbol{\Phi}_y(\partial^2\boldsymbol{\Phi}_z * p_\varepsilon(\mathbf{z}))^\top\mathbf{1}$.

Now, we can obtain the RFF approximation of (4.10):

$$\boldsymbol{\beta}_K = -\frac{1}{nK\lambda}\partial\boldsymbol{\Phi}_K\partial\boldsymbol{\Phi}_K^\top\boldsymbol{\beta}_K + \frac{1}{n^2\lambda^2}\partial\boldsymbol{\Phi}_K \odot (\partial^2\boldsymbol{\Phi}_z * p(\mathbf{z}))^\top\mathbf{1},$$

where

$$\partial\boldsymbol{\Phi}_K = \begin{bmatrix}\boldsymbol{\Phi}_{z_1}\\ \vdots \\ \boldsymbol{\Phi}_{z_K}\end{bmatrix}, \quad \partial\boldsymbol{\Phi}_K \odot (\partial^2\boldsymbol{\Phi}_z * p(\mathbf{z}))^\top\mathbf{1} = \begin{bmatrix}\partial\boldsymbol{\Phi}_{z_1}(\partial^2\boldsymbol{\Phi}_z * p(\mathbf{z}))^\top\mathbf{1}\\ \vdots \\ \partial\boldsymbol{\Phi}_{z_K}(\partial^2\boldsymbol{\Phi}_z * p(\mathbf{z}))^\top\mathbf{1}\end{bmatrix}.$$

Denoting

$$\boldsymbol{h} = \frac{1}{n}(\partial^2 \boldsymbol{\Phi}_z * p(\mathbf{z}))^\top \mathbf{1}, \quad \boldsymbol{H} = \int p_\varepsilon(\mathbf{y}) \partial \boldsymbol{\Phi}_y^\top \partial \boldsymbol{\Phi}_y d\mathbf{y}$$

we obtain the following expression for the descretized solution f_K:

$$\begin{aligned} f_K &= \frac{1}{n\lambda^2}\boldsymbol{\phi}(\cdot)^\top \boldsymbol{H}\left[-\frac{1}{K}\partial\boldsymbol{\Phi}_K^\top\left(\frac{1}{K}\partial\boldsymbol{\Phi}_K\partial\boldsymbol{\Phi}_K^\top + n\lambda\mathbf{I}\right)^{-1}\partial\boldsymbol{\Phi}_K \odot \boldsymbol{h} + \boldsymbol{h}\right] - \frac{1}{\lambda}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{h} \\ &= \frac{1}{n\lambda^2}\boldsymbol{\phi}(\cdot)^\top \boldsymbol{H}\left[-\frac{1}{K}\left(\frac{1}{K}\partial\boldsymbol{\Phi}_K^\top\partial\boldsymbol{\Phi}_K + n\lambda\mathbf{I}\right)^{-1}\partial\boldsymbol{\Phi}_K^\top\partial\boldsymbol{\Phi}_K \odot \boldsymbol{h} + \boldsymbol{h}\right] - \frac{1}{\lambda}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{h}. \end{aligned}$$

By taking a limit over $K \to \infty$, and using $\boldsymbol{H} = \lim_{K\to\infty} \frac{1}{K}\partial\boldsymbol{\Phi}_K^\top\partial\boldsymbol{\Phi}_K$ along with

$$\lim_{K\to\infty}(n\lambda\mathbf{I} + \frac{1}{K}\partial\boldsymbol{\Phi}_K^\top\partial\boldsymbol{\Phi}_K)\lim_{K\to\infty}\frac{1}{K}\partial\boldsymbol{\Phi}_K^\top\boldsymbol{\beta}_K = \lim_{K\to\infty}\frac{1}{K\lambda}\partial\boldsymbol{\Phi}_K^\top\partial\boldsymbol{\Phi}_K\boldsymbol{h},$$

the solution f^* is given as

$$\begin{aligned} f_m^* = \lim_{K\to\infty} f_K &= -\frac{1}{n\lambda^2}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{H}(\boldsymbol{H}+n\lambda\mathbf{I})^{-1}\boldsymbol{H}h + \frac{1}{n\lambda^2}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{H}h - \frac{1}{\lambda}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{h} \\ &= \frac{1}{\lambda}\boldsymbol{\phi}(\cdot)^\top(\boldsymbol{H}+n\lambda\mathbf{I})^{-1}\boldsymbol{H}h - \frac{1}{\lambda}\boldsymbol{\phi}(\cdot)^\top\boldsymbol{h}, \end{aligned}$$

where index m refers to the number of RFF features.

B.1.3 Proof for the error bounds of score matching with RFF

The idea of this proof is to upper bound the expected square difference between solutions:

$$\mathbb{E}_{\mathbf{x},\mathbf{w}}(f_{n,m}^*(\mathbf{x}) - f_n^*(\mathbf{x}))^2, \tag{B.3}$$

where the difference between RFF and exact kernel solutions ($f_{n,m}^*,\ f_n^*$) is expressed as follows:

$$\begin{aligned} f_{n,m}^* - f_n^* &= -\frac{1}{\lambda n}\partial^2\boldsymbol{k}(\cdot)^\top\mathbf{1} + \frac{1}{\lambda n}\boldsymbol{\phi}^\top(\cdot)\partial^2\boldsymbol{\Phi}^\top\mathbf{1} \\ &\quad + \frac{1}{\lambda n}\partial\boldsymbol{k}(\cdot)^\top(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1} - \frac{1}{\lambda n}\boldsymbol{\phi}^\top(\cdot)\partial\boldsymbol{\Phi}^\top(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}\partial\boldsymbol{\Phi}\partial^2\boldsymbol{\Phi}^\top\mathbf{1} \\ &= \frac{1}{\lambda n}(\partial^2\boldsymbol{\Phi}\boldsymbol{\phi}(\cdot) - \partial^2\boldsymbol{k}(\cdot))^\top\mathbf{1} + \frac{1}{\lambda n}(\partial\boldsymbol{k}(\cdot) - \partial\boldsymbol{\Phi}\boldsymbol{\phi}(\cdot))^\top(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1} \\ &\quad + \frac{1}{\lambda n}\boldsymbol{\phi}^\top(\cdot)\partial\boldsymbol{\Phi}^\top\left[(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1} - (\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}\right]\partial\partial^2\boldsymbol{K}\mathbf{1} \\ &\quad + \frac{1}{\lambda n}\boldsymbol{\phi}^\top(\cdot)\partial\boldsymbol{\Phi}^\top(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}(\partial\partial^2\boldsymbol{K} - \partial\boldsymbol{\Phi}\partial^2\boldsymbol{\Phi}^\top)\mathbf{1}. \end{aligned}$$

The above expectation is taken jointly over random Fourier weights and given points $\mathbf{x} \sim p_0$. It can then be written as $\mathbb{E}_{\mathbf{x},\mathbf{w}}[f] = \mathbb{E}_{\mathbf{w}}\mathbb{E}_{\mathbf{x}}[f|\mathbf{w}]$. The first term in the above

expression is the difference between $\hat{\xi}$ and its RFF approximation $\hat{\xi}_m$, so, we have:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{w}}\mathbf{1}^\top(\partial^2\boldsymbol{\Phi}\boldsymbol{\phi}(\cdot) - \partial^2\boldsymbol{k}(\cdot))(\partial^2\boldsymbol{\Phi}\boldsymbol{\phi}(\cdot) - \partial^2\boldsymbol{k}(\cdot))^\top\mathbf{1} \\
&\quad= \mathbf{1}^\top\left[\mathbb{E}_{\mathbf{w}}[\partial^2\boldsymbol{\Phi}\boldsymbol{\phi}(\cdot)\boldsymbol{\phi}(\cdot)^\top\partial^2\boldsymbol{\Phi}^\top] - \partial^2\boldsymbol{k}(\cdot)\partial^2\boldsymbol{k}(\cdot)^\top\right]\mathbf{1} \\
&\quad\leq \frac{m-1}{m}\mathbf{1}^\top\partial^2\boldsymbol{k}(\cdot)\partial^2\boldsymbol{k}^\top(\cdot)\mathbf{1} + \frac{1}{m}\mathbf{1}^\top\partial^2\partial^2\boldsymbol{K}\mathbf{1} - \mathbf{1}^\top\partial^2\boldsymbol{k}(\cdot)\partial^2\boldsymbol{k}(\cdot)^\top\mathbf{1} \\
&\quad\leq \frac{1}{m}\mathbf{1}^\top\partial^2\partial^2\boldsymbol{K}\mathbf{1},
\end{aligned}$$

where the first inequality is obtained using $\sup_{\mathbf{x}}|\phi_i(\boldsymbol{W}\mathbf{x}+\boldsymbol{b})| \leq 1$. As this expression does not depend on $\mathbf{x}$, the joint expectation will be the same.

For the second term in $f^*_{n,m}(\mathbf{x}) - f^*_n(\mathbf{x})$, derivation of the upper bound is technically the same, but with lower-order derivatives, so

$$\begin{aligned}
\mathbb{E}_{\mathbf{w}}\left[(\partial\boldsymbol{k}(\cdot) - \partial\boldsymbol{\Phi}\boldsymbol{\phi}(\cdot))^\top(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\right]^2 \leq \\
\frac{1}{m}\|\partial\partial\boldsymbol{K}^{\frac{1}{2}}(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2.
\end{aligned}$$

The third them is

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x},\mathbf{w}}\left[\boldsymbol{\phi}^\top(\cdot)\partial\boldsymbol{\Phi}^\top\left[(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1} - (\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}\right]\partial\partial^2\boldsymbol{K}\mathbf{1}\right]^2 \\
&\quad= \mathbb{E}_{\mathbf{x},\mathbf{w}}\left[\boldsymbol{\phi}^\top(\cdot)\partial\boldsymbol{\Phi}^\top(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}(\partial\partial\boldsymbol{K} - \partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top)(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\right]^2 \\
&\quad\leq \mathbb{E}_{\mathbf{w}}\mathbb{E}_{\mathbf{x}}\left[\|\mathbf{R}\|_2\|(\partial\partial\boldsymbol{K} - \partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top)(\partial\partial\boldsymbol{K} + \lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2|\mathbf{w}\right],
\end{aligned}$$

where only $\mathbf{R}$ depends on $\mathbf{x}$.

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}}\mathbf{R} &= (\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}\partial\boldsymbol{\Phi}\mathbb{E}_{\mathbf{x}}\left[\boldsymbol{\phi}(\cdot)\boldsymbol{\phi}^\top(\cdot)\right]\partial\boldsymbol{\Phi}^\top(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1} \\
&= \frac{1}{n}(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}\partial\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \varepsilon nI)\partial\boldsymbol{\Phi}^\top(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}
\end{aligned}$$

This inequality holds with the probability $1-\delta$ for $n \geq \frac{8}{3\varepsilon^2}\log\frac{m}{\delta}$ and is obtained from the Bernstein inequality assuming that the weights are fixed Tropp (2015).

$$\begin{aligned}
\lambda_{\max}(\mathbf{R}) &= \frac{1}{n}\lambda_{\max}\left[(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}\partial\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + n\varepsilon I)\partial\boldsymbol{\Phi}^\top(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}\right] \\
&\leq \frac{1}{n}\lambda_{\max}\left[(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + n\varepsilon I)\partial\boldsymbol{\Phi}^\top(\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top + \lambda n\mathbf{I})^{-1}\partial\boldsymbol{\Phi}\right] \\
&\leq \frac{1}{n}\lambda_{\max}\left[(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + n\varepsilon I)\right] \leq \frac{1}{n}\mathrm{tr}\left[(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + n\varepsilon I)\right] \\
&\leq \left(\frac{1}{m}\max_i\sup_{\mathbf{x}}\|\phi_i(\boldsymbol{W}\mathbf{x}+\boldsymbol{b})\|^2 + \varepsilon\right) \leq \frac{1}{m} + \varepsilon
\end{aligned}$$

$$\mathbb{E}_{\mathbf{w}}\|(\partial\partial\boldsymbol{K}-\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top)(\partial\partial\boldsymbol{K}+\lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2$$
$$=\mathbf{1}^\top\partial\partial^2\boldsymbol{K}^\top(\partial\partial\boldsymbol{K}+\lambda n\mathbf{I})^{-1}(\mathbb{E}_{\mathbf{w}}\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top-\partial\partial\boldsymbol{K}^2)(\partial\partial\boldsymbol{K}+\lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}$$

$$\mathbb{E}_{\mathbf{w}}\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top=\frac{m-1}{m}\partial\partial\boldsymbol{K}^2+\frac{1}{m}\mathbf{D}_1.$$

The latter term here is obtained under the assumption that $\mathbf{D}_1$ does not depend on $\mathbf{x}$. This assumption holds for sufficiently smooth kernels and we can rewrite the expression under an expectation as the polynomial of weights times the trigonometric function.

$$\mathbb{E}_{\mathbf{w}}\|(\partial\partial\boldsymbol{K}-\partial\boldsymbol{\Phi}\partial\boldsymbol{\Phi}^\top)(\partial\partial\boldsymbol{K}+\lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2\leq\frac{1}{m}\|\mathbf{D}_1^{\frac{1}{2}}(\partial\partial\boldsymbol{K}+\lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2$$

Analogously, for the last term, using the assumption that $\mathbf{D}_2<\infty$, we have

$$\mathbb{E}_{\mathbf{x},\mathbf{w}}\partial\boldsymbol{\Phi}\partial^2\boldsymbol{\Phi}^\top\partial\boldsymbol{\Phi}\partial^2\boldsymbol{\Phi}^\top\leq\frac{1}{m}\|\mathbf{D}_2^{\frac{1}{2}}\mathbf{1}\|^2.$$

Finally, combining all the above, we have

$$\mathbb{E}_{\mathbf{x},\mathbf{w}}(f^*_{n,m}(\mathbf{x})-f^*_n(\mathbf{x}))^2\leq\frac{2}{\lambda^2n^2m^2}\Big[m\mathbf{1}^\top\partial^2\partial^2\boldsymbol{K}\mathbf{1}+m\|\partial\partial\boldsymbol{K}^{\frac{1}{2}}(\partial\partial\boldsymbol{K}+\lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2$$
$$+(1+\varepsilon m)\|\mathbf{D}_2^{\frac{1}{2}}\mathbf{1}\|^2+(1+\varepsilon m)\|\mathbf{D}_1^{\frac{1}{2}}(\partial\partial\boldsymbol{K}+\lambda n\mathbf{I})^{-1}\partial\partial^2\boldsymbol{K}\mathbf{1}\|^2\Big].$$

B.1.4 Derivation of $\boldsymbol{H}$ and $\boldsymbol{h}$ for Gaussian noise

$$\begin{aligned}\boldsymbol{H}&=\partial\Phi^\top\partial\Phi*p_\varepsilon\\&=\sum_{a=1}^n\sum_{i=1}^d\partial_i\phi(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\partial_i\phi^\top(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})*p_\varepsilon\\&=\sum_{a=1}^n\sum_{i=1}^d\boldsymbol{W}_{:,i}\boldsymbol{W}_{:,i}^\top\odot\phi'(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\phi'^\top(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})*p_\varepsilon\\&=\frac{1}{M}\boldsymbol{W}\boldsymbol{W}^\top\odot\sum_{a=1}^n\sin(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\sin^\top(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})*p_\varepsilon\end{aligned}$$

Assuming that $p_\varepsilon=\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})$ and using

$$\cos(\mathbf{w}^\top\mathbf{x})*\mathcal{N}(0,\sigma^2\mathbf{I})=e^{-\frac{\sigma^2}{2}\|\mathbf{w}\|_2^2}\cos(\mathbf{w}^\top\mathbf{x}),$$

we will obtain

$$\begin{aligned}\sin(\mathbf{w}^\top\mathbf{x}+b)\sin(\mathbf{v}^\top\mathbf{x}+c)*p_\varepsilon &= \frac{1}{2}\left[\cos((\mathbf{w}-\mathbf{v})^\top\mathbf{x}+b-c)-\cos((\mathbf{w}+\mathbf{v}+b+c)^\top\mathbf{x})\right]*p_\varepsilon \\ &= \frac{1}{2}e^{-\frac{\sigma^2}{2}\|\mathbf{w}-\mathbf{v}\|_2^2}\cos((\mathbf{w}-\mathbf{v})^\top\mathbf{x}+b-c) \\ &\quad -\frac{1}{2}e^{-\frac{\sigma^2}{2}\|\mathbf{w}+\mathbf{v}\|_2^2}\cos((\mathbf{w}+\mathbf{v})^\top\mathbf{x}+b+c)\end{aligned}$$

$$\begin{aligned}\boldsymbol{H} = \frac{1}{2M}\boldsymbol{W}\boldsymbol{W}^\top \odot \sum_{a=1}^{n}\Big[&e^{-\frac{\sigma^2}{2}\|\mathbf{w}_i-\mathbf{w}_j\|_2^2}\cos((\mathbf{w}_i-\mathbf{w}_j)^\top\mathbf{x}_a+\mathbf{b}_i-\mathbf{b}_j) \\ &-e^{-\frac{\sigma^2}{2}\|\mathbf{w}_i+\mathbf{w}_j\|_2^2}\cos((\mathbf{w}_i+\mathbf{w}_j)^\top\mathbf{x}_a+\mathbf{b}_i+\mathbf{b}_j).\Big]\end{aligned} \tag{B.4}$$

Next, firstly assume that q_0 is uniform:

$$\begin{aligned}\boldsymbol{h} &= \frac{1}{n}\sum_{a=1}^{n}\sum_{i=1}^{d}\partial_i^2\phi(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})*p_\varepsilon \\ &= -\frac{1}{n\sqrt{M}}\sum_{a=1}^{n}\sum_{i=1}^{d}\boldsymbol{W}_{:,i}^2\odot\cos(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})*p_\varepsilon \\ &= -\frac{1}{n\sqrt{M}}\sum_{a=1}^{n}\operatorname{diag}(\boldsymbol{W}\boldsymbol{W}^\top)\odot e^{-\frac{\sigma^2}{2}\operatorname{diag}(\boldsymbol{W}\boldsymbol{W}^\top)}\odot\cos(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\end{aligned}$$

For a multivariate normal $q_0(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}\mathbf{and}\boldsymbol{\Sigma})$, $\nabla\log q_0(\mathbf{x}) = -\boldsymbol{\Sigma}^{-\mathbf{1}}(\mathbf{x}-\boldsymbol{\mu})$, there will be additional term to $\boldsymbol{h}$:

$$\begin{aligned}\boldsymbol{h} &= \frac{1}{n}\sum_{a=1}^{n}\sum_{i=1}^{d}\partial_i\phi(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\partial_i\log q_0(\mathbf{x}_a)*p_\varepsilon \\ &= -\frac{1}{n\sqrt{M}}\sum_{a=1}^{n}\sum_{i=1}^{d}\boldsymbol{W}_{:,i}\sin(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\partial_i\log q_0(\mathbf{x}_a)*p_\varepsilon \\ &= -\frac{1}{n\sqrt{M}}\sum_{a=1}^{n}\sin(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\odot\boldsymbol{W}\nabla\log q_0(\mathbf{x}_a)*p_\varepsilon.\end{aligned}$$

Using

$$\mathbf{w}^\top\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\sin(\mathbf{w}^\top\mathbf{x})*p_\varepsilon = e^{-\frac{\sigma^2\|\mathbf{w}\|^2}{2}}\mathbf{w}^\top\boldsymbol{\Sigma}^{-1}\left[(\mathbf{x}-\boldsymbol{\mu})\sin(\mathbf{w}^\top\mathbf{x})+\sigma^2\mathbf{w}\cos(\mathbf{w}^\top\mathbf{x})\right],$$

we obtain

$$\begin{aligned}\boldsymbol{h} = \frac{1}{n\sqrt{M}}e^{-\frac{\sigma^2}{2}\operatorname{diag}(\boldsymbol{W}\boldsymbol{W}^\top)}\odot\sum_{a=1}^{n}\Big[&\sin(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\odot\boldsymbol{W}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_a-\boldsymbol{\mu}) \\ &+\sigma^2\cos(\boldsymbol{W}\mathbf{x}_a+\boldsymbol{b})\odot\operatorname{diag}(\boldsymbol{W}\boldsymbol{\Sigma}^{-1}\boldsymbol{W}^\top)\Big]\end{aligned} \tag{B.5}$$

In the case of arbitrary q_0, we use the Taylor expansion:

$$\nabla \log q_0(\mathbf{x}+\boldsymbol{\varepsilon}) \approx \nabla \log q_0(\mathbf{x}) + \nabla^2 \log q_0(\mathbf{x})\boldsymbol{\varepsilon}$$

In the vicinity of $\mathbf{x}$, it is equivalent to the previous case and the additional term is obtained with a simple replacement: $-\boldsymbol{\Sigma}^{-1} \to \nabla^2 \log q_0(\mathbf{x})$ and $-\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \to \nabla \log q_0(\mathbf{x})$.

B.1.5 Derivation of $\boldsymbol{H}$ and $\boldsymbol{h}$ for arc-cosine kernels

$$\begin{aligned}
\boldsymbol{H} &= \partial\Phi^\top \partial\Phi * p_\varepsilon \\
&= \sum_{a=1}^{n}\sum_{i=1}^{d} \partial_i\phi(\boldsymbol{W}\mathbf{x}_a)\partial_i\phi^\top(\boldsymbol{W}\mathbf{x}_a) * p_\varepsilon \\
&= \sum_{a=1}^{n}\sum_{i=1}^{d} \boldsymbol{W}_{:,i}\boldsymbol{W}_{:,i}^\top \odot \mathbf{1}(\boldsymbol{W}\mathbf{x}_a) \odot p^2(\boldsymbol{W}\mathbf{x}_a)^{p-1}\left((\boldsymbol{W}\mathbf{x}_a)^{p-1}\right)^\top * p_\varepsilon
\end{aligned}$$

Considering $p=2$, with uniform base density q_0 and isotropic Gaussian noise, we will obtain:

$$\boldsymbol{W}\mathbf{x}_a\mathbf{x}_a^\top\boldsymbol{W}^\top * p_\varepsilon = \boldsymbol{W}\mathbb{E}_{p_\varepsilon}(\mathbf{x}_a+\boldsymbol{\varepsilon})(\mathbf{x}_a+\boldsymbol{\varepsilon})^\top\boldsymbol{W}^\top * p_\varepsilon = \boldsymbol{W}(\mathbf{x}_a\mathbf{x}_a^\top + \sigma^2\mathbf{I})\boldsymbol{W}^\top$$

The same holds for any symmetric noise distribution with covariance $\boldsymbol{\Sigma}$ and corresponding substitution to the above equation.

$$\boldsymbol{H} = 4\boldsymbol{W}\boldsymbol{W}^\top \odot \sum_{a=1}^{n} \mathbf{1}(\boldsymbol{W}\mathbf{x}_a) \odot \boldsymbol{W}(\mathbf{x}_a\mathbf{x}_a^\top + \sigma^2\mathbf{I})\boldsymbol{W}^\top \tag{B.6}$$

Moving to the computation of $\boldsymbol{h}$, we have:

$$\begin{aligned}
\boldsymbol{h} &= \frac{1}{n}\sum_{a=1}^{n}\sum_{i=1}^{d} \partial_i^2\phi(\boldsymbol{W}\mathbf{x}_a) * p_\varepsilon \\
&= \frac{2}{n}\sum_{a=1}^{n}\sum_{i=1}^{d} \boldsymbol{W}_{:,i}^2 \odot \mathbf{1}(\boldsymbol{W}\mathbf{x}_a) \odot (\boldsymbol{W}\mathbf{x}_a) * p_\varepsilon \\
&= \frac{2}{n}\mathrm{diag}(\boldsymbol{W}\boldsymbol{W}^\top) \odot \sum_{a=1}^{n} \mathbf{1}(\boldsymbol{W}\mathbf{x}_a) \odot (\boldsymbol{W}\mathbf{x}_a),
\end{aligned}$$

where the last line holds for any symmetric zero-mean density.

B.1.6 The Taylor approximation of denoising score-matching

Considering the data corrupted by a small Gaussian noise $\hat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and by applying the Taylor expansion to the model density p_m, we obtain

$$\log p_m(\mathbf{x} + \boldsymbol{\varepsilon}, \boldsymbol{\theta}) = \log p_m(\mathbf{x}, \boldsymbol{\theta}) + \nabla \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \boldsymbol{\varepsilon} + \frac{1}{2}\boldsymbol{\varepsilon}^\top \nabla^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \boldsymbol{\varepsilon} + O(\|\boldsymbol{\varepsilon}\|_2^3),$$

where $\boldsymbol{\theta}$ denotes a vector of model parameters, $\mathbb{E}[\varepsilon] = \mathbf{0}$, $\mathbb{E}[\varepsilon\varepsilon^\top] = \sigma^2\mathbf{I}$.

$$\mathbb{E}_{\boldsymbol{\varepsilon}}[\Delta_x \log p_m(\mathbf{x} + \boldsymbol{\varepsilon}, \boldsymbol{\theta})] \approx \Delta_x \log p_m(\mathbf{x}, \boldsymbol{\theta}) + \frac{\sigma^2}{2}\Delta_x^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})$$

$$\begin{aligned}\|\nabla_x \log p_m(\mathbf{x} + \boldsymbol{\varepsilon}, \boldsymbol{\theta})\|_2^2 = \|\nabla_x \log p_m(\mathbf{x}, \boldsymbol{\theta})\|_2^2 &+ 2\nabla_x \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \nabla_x^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})\boldsymbol{\varepsilon} \\ &+ \boldsymbol{\varepsilon}^\top \nabla_x^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \nabla_x^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})\boldsymbol{\varepsilon} \\ &+ \nabla_x \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \nabla_x \boldsymbol{\varepsilon}^\top \nabla_x^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})\boldsymbol{\varepsilon} + O(\|\boldsymbol{\varepsilon}\|_2^3)\end{aligned}$$

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\varepsilon}}\|\nabla_x \log p_m(\mathbf{x} + \boldsymbol{\varepsilon}, \boldsymbol{\theta})\|_2^2 \approx \|\nabla_x \log p_m(\mathbf{x}, \boldsymbol{\theta})\|_2^2 &+ \sigma^2 \mathrm{tr}\left[\nabla_x^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \nabla_x^2 log p_m(\mathbf{x}, \boldsymbol{\theta})\right] \\ &+ \sigma^2 \nabla_x \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \nabla_x \Delta_x \log p_m(\mathbf{x}, \boldsymbol{\theta})\end{aligned}$$

Finally, we have

$$\begin{aligned}J_{\boldsymbol{\varepsilon}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \frac{\sigma^2}{2}\mathbb{E}_{p_0}\left[(\Delta_x)^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})\right] + \mathbb{E}_{p_0}\mathrm{tr}\left[\nabla_x^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \nabla_x^2 \log p_m(\mathbf{x}, \boldsymbol{\theta})\right] \\ + \mathbb{E}_{p_0}\left[\nabla_x \log p_m(\mathbf{x}, \boldsymbol{\theta})^\top \nabla_x \Delta_x \log p_m(\mathbf{x}, \boldsymbol{\theta})\right]\end{aligned}$$

where p_0 corresponds to an unknown data distribution.

B.1.7 The Nyström kernel approximation

Let $\boldsymbol{K}$ be a sample Gram matrix, then for the Nyström kernel approximation Chen et al. (2016), we have:

$$\boldsymbol{K} = \begin{bmatrix}\boldsymbol{K}_{11} & \boldsymbol{K}_{12} \\ \boldsymbol{K}_{12}^\top & \boldsymbol{K}_{22}\end{bmatrix} \quad \boldsymbol{K} \approx \begin{bmatrix}\boldsymbol{K}_{11} \\ \boldsymbol{K}_{12}^\top\end{bmatrix} \boldsymbol{K}_{11}^{-1} \begin{bmatrix}\boldsymbol{K}_{11} & \boldsymbol{K}_{12}\end{bmatrix} \quad \phi(x) = \boldsymbol{K}_{11}^{-\frac{1}{2}} \boldsymbol{k}(\mathbf{x})$$

where $\boldsymbol{k}(\mathbf{x}) = \begin{bmatrix} \boldsymbol{k}(\mathbf{x}, \mathbf{x}_1) & \dots & \boldsymbol{k}(\mathbf{x} and \mathbf{x}_M) \end{bmatrix}^\top$, M is the amount of subsampled points.

$$\partial \boldsymbol{K} = \begin{bmatrix} \partial_1 \boldsymbol{k}^\top(\mathbf{x}_1) \\ \dots \\ \partial_d \boldsymbol{k}^\top(\mathbf{x}_1) \\ \partial_1 \boldsymbol{k}^\top(\mathbf{x}_2) \\ \dots \\ \partial_d \boldsymbol{k}^\top(\mathbf{x}_N) \end{bmatrix}, \quad \partial\Phi = \partial \boldsymbol{K} \boldsymbol{K}_{11}^{-\frac{1}{2}} \quad \partial^2 \boldsymbol{K} = \begin{bmatrix} \partial_1^2 \boldsymbol{k}^\top(\mathbf{x}_1) \\ \dots \\ \partial_d^2 \boldsymbol{k}^\top(\mathbf{x}_1) \\ \partial_1^2 \boldsymbol{k}^\top(\mathbf{x}_2) \\ \dots \\ \partial_d^2 \boldsymbol{k}^\top(\mathbf{x}_N) \end{bmatrix}, \quad \partial^2 \boldsymbol{\Phi} = \partial^2 \boldsymbol{K} \boldsymbol{K}_{11}^{-\frac{1}{2}}$$

$$\boldsymbol{G} = \partial \boldsymbol{K}^\top \partial \boldsymbol{K} * p_\varepsilon, \quad \boldsymbol{g} = \frac{1}{n} (\partial^2 \boldsymbol{K} * p_\varepsilon)^\top \mathbf{1}$$

$$\begin{aligned} f &= \frac{\boldsymbol{k}^\top(\cdot)}{\lambda} K_{11}^{-\frac{1}{2}} \left[\boldsymbol{K}_{11}^{-\frac{1}{2}} \boldsymbol{g} + (\boldsymbol{K}_{11}^{-\frac{1}{2}} \boldsymbol{G} \boldsymbol{K}_{11}^{-\frac{1}{2}} + n\lambda \mathbf{I})^{-1} \boldsymbol{K}_{11}^{-\frac{1}{2}} G \boldsymbol{K}_{11}^{-1} \boldsymbol{g} \right] \\ &= \frac{\boldsymbol{k}^\top(\cdot)}{\lambda} \left[\boldsymbol{K}_{11}^{-1} \boldsymbol{g} + (\boldsymbol{G} + n\lambda \boldsymbol{K}_{11})^{-1} \boldsymbol{G} \boldsymbol{K}_{11}^{-1} \boldsymbol{g} \right] \end{aligned}$$

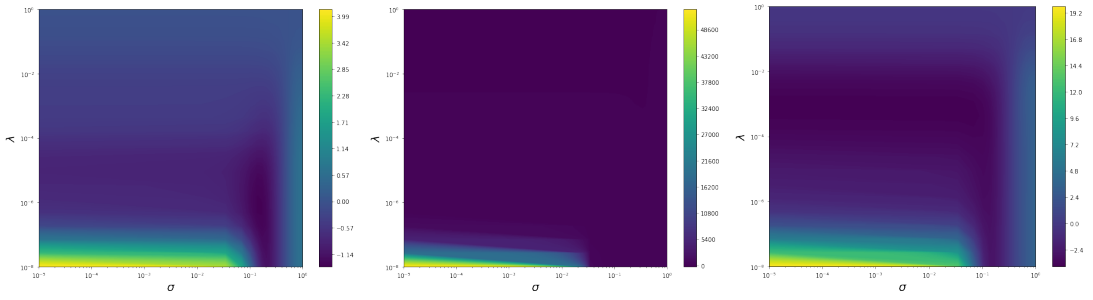
B.2 Tables and Figures

TABLE B.1: Results of score-matching algorithms; 100 features and 1000 sample size for cosine, uniform, banana and funnel distributions.

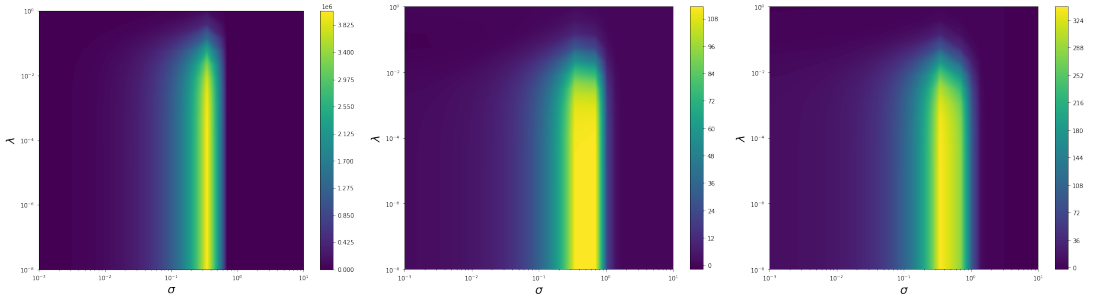
Distribution	Cosine		Uniform		Banana		Funnel	
Model	KDSM	RFFSM	KDSM	RFFSM	KDSM	RFFSM	KDSM	RFFSM
F_{train}	**2.197**	5.331	**1.365**	1.785	0.301	**0.28**	0.34	**0.288**
F_{test}	**1.858**	5.102	**1.584**	1.901	**0.291**	0.319	0.339	**0.307**
LL_{train}	-5.53	-5.008	-3.66	-3.649	-3.529	-3.528	-2.867	-2.846
$LL_{p\ train}$	-3.528	-3.528	-3.584	-3.584	-2.83	-2.83	-2.868	-2.868
LL_{test}	-5.648	-5.056	-3.689	-3.692	-3.659	-3.697	-2.821	-2.783
$LL_{p\ test}$	-3.503	-3.503	-3.584	-3.584	-2.894	-2.894	-2.796	-2.796
FSSD	-0.128	0.059	0.212	0.189	-0.085	-0.045	0.058	-0.041
p-value	**0.425**	0.308	0.093	**0.131**	**0.604**	0.452	0.299	**0.392**
W_1	**0.251**	0.372	0.06	**0.055**	**0.047**	0.052	**0.06**	0.084

TABLE B.2: Results of score-matching algorithms; 100 features and 1000 sample size for ring, mixture of rings and mixture of uniforms.

Distribution	Ring		Rings		Uniforms	
Model	KDSM	RFFSM	KDSM	RFFSM	KDSM	RFFSM
F_{train}	0.862	**0.635**	3.664	**3.528**	**3.705**	4.97
F_{test}	0.803	**0.562**	3.298	3.293	**3.582**	4.82
LL_{train}	-2.35	-2.328	-3.668	**-4.221**	**-3.046**	-27.879
$LL_{p\ train}$	-3.949	-3.949	-4.68	-4.68	-2.89	-2.89
LL_{test}	-2.338	-2.346	-3.591	**-4.13**	**-3.08**	-27.904
$LL_{p\ test}$	-3.929	-3.929	-4.633	-4.633	-2.89	-2.89
FSSD	-1.316	-1.219	-0.759	-0.851	0.057	-0.367
p-value	**0.775**	0.694	**0.985**	0.859	**0.347**	0.673
W_1	**0.063**	0.086	0.212	**0.15**	**0.26**	0.327

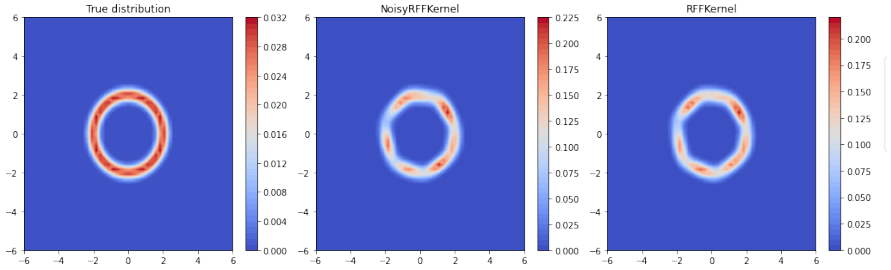


(A) Cosine (B) Banana (C) Ring

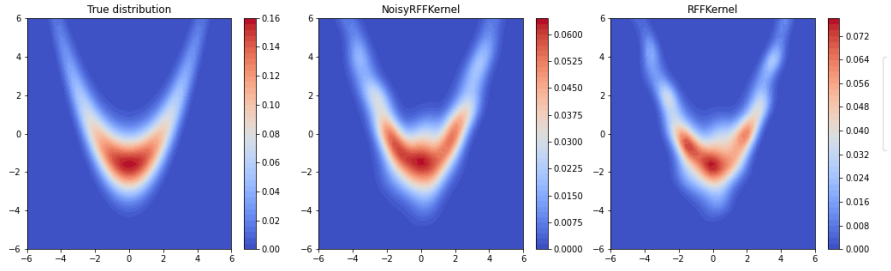


(D) MiniBoone (E) Red Wine (F) White Wine

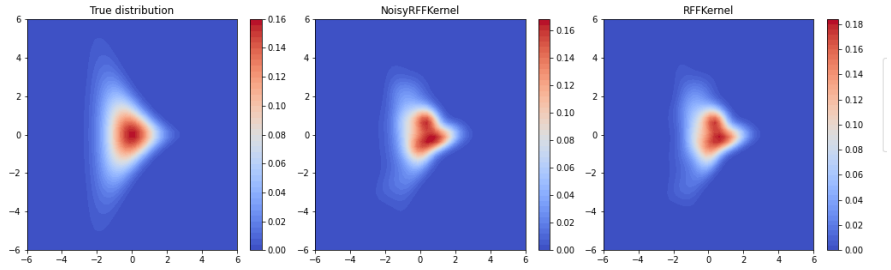
FIGURE B.1: Loss surface w.r.t. the regularization parameter λ (y axis) and noise parameter σ (x axis).



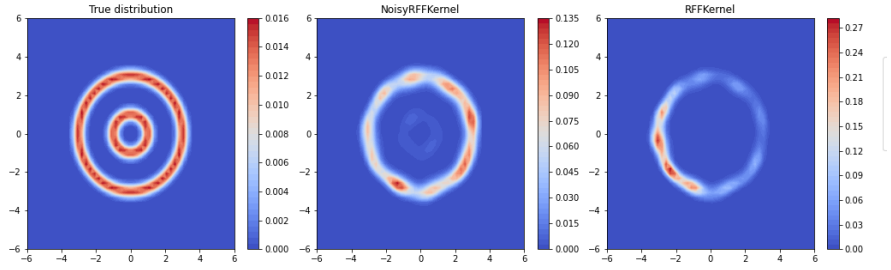
(A) Ring



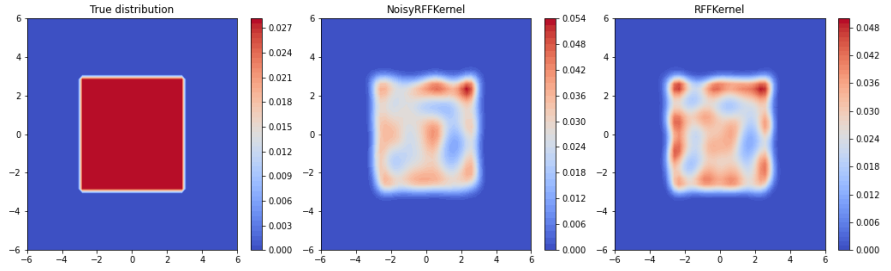
(B) Banana



(C) Funnel



(D) Mixture of rings



(E) Uniform

FIGURE B.2: Score-matching density estimation using 1000 samples. The left column is a ground truth, middle is DSM RFf, and the right is SM RFF.

Bibliography

https://github.com/wittawatj/kernel-gof. URL https://github.com/wittawatj/kernel-gof.

https://github.com/karlnapf/kernel_exp_family. URL https://github.com/karlnapf/kernel_exp_family.

Abdessalem, A. B., Dervilis, N., Wagg, D. J., and Worden, K. Automatic kernel selection for gaussian processes regression with approximate bayesian computation and sequential monte carlo. *Frontiers in Built Environment*, 3:52, 2017.

Acar, E., Dunlavy, D. M., Kolda, T. G., and Mørup, M. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

Armand, S. C. *Structural Optimization Methodology for Rotating Disks of Aircraft Engines*. NASA technical memorandum. National Aeronautics and Space Administration, Office of Management, Scientific and Technical Information Program, 1995.

Aronszajn, N. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Ba, J., Erdogdu, M., Suzuki, T., Wu, D., and Zhang, T. Generalization of two-layer neural networks: An asymptotic viewpoint. In *International Conference on Learning Representations*, 2019.

Bach, F. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research*, 18(21):1–38, 2017.

Bailey, T. and Durrant-Whyte, H. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006.

Ballani, J. and Grasedyck, L. Hierarchical tensor approximation of output quantities of parameter-dependent pdes. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1): 852–872, 2015.

Barak, B. and Moitra, A. Noisy tensor completion via the sum-of-squares hierarchy. In *Conference on Learning Theory*, pp. 417–445, 2016.

Barfoot, T. D. *State Estimation for Robotics*. Cambridge University Press, 2017.

Barfoot, T. D., Tong, C. H., and Särkkä, S. Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Robotics: Science and Systems*, volume 10. Citeseer, 2014a.

Barfoot, T. D., Tong, C. H., and Särkkä, S. Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Robotics: Science and Systems*, volume 10. Citeseer, 2014b.

Belyaev, M., Burnaev, E., Kapushev, E., Panov, M., Prikhodko, P., Vetrov, D., and Yarotsky, D. Gtapprox: Surrogate modeling for industrial design. *Advances in Engineering Software*, 102:29–39, 2016.

Bibby, C. and Reid, I. A hybrid slam representation for dynamic marine environments. In *2010 IEEE International Conference on Robotics and Automation*, pp. 257–264. IEEE, 2010.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007. ISBN 0387310738. URL http://www.amazon.com/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0387310738.

Brault, R., Heinonen, M., and Buc, F. Random fourier features for operator-valued kernels. In *Asian Conference on Machine Learning*, pp. 110–125, 2016.

Chamakh, L., Gobet, E., and Szabó, Z. Orlicz random fourier features. *Journal of Machine Learning Research*, 21(145):1–37, 2020.

Chen, H., Xia, H., Huang, H., and Cai, W. Error analysis of generalized nyström kernel regression. In *Advances in Neural Information Processing Systems*, pp. 2541–2549, 2016.

Chen, Y., Guo, Y., Wang, Y., Wang, D., Peng, C., and He, G. Denoising of hyperspectral images using nonconvex low rank matrix approximation. *IEEE Transactions on Geoscience and Remote Sensing*, 55(9):5366–5380, 2017.

Chen, Y.-L., Hsu, C.-T. C., and Liao, H.-Y. M. Simultaneous tensor decomposition and completion using factor priors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):1, 2013.

Cho, Y. and Saul, L. K. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pp. 342–350, 2009.

Choromanski, K. and Sindhwani, V. Recycling randomness with structure for sublinear time kernel expansions. *arXiv preprint arXiv:1605.09049*, 2016.

Choromanski, K., Rowland, M., and Weller, A. The unreasonable effectiveness of random orthogonal embeddings. *arXiv preprint arXiv:1703.00864*, 2017.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Cunningham, A., Indelman, V., and Dellaert, F. Ddf-sam 2.0: Consistent distributed smoothing and mapping. In *2013 IEEE international conference on robotics and automation*, pp. 5220–5227. IEEE, 2013.

Cutajar, K., Osborne, M., Cunningham, J., and Filippone, M. Preconditioning kernel matrices. In *International Conference on Machine Learning*, pp. 2529–2538, 2016.

Cutajar, K., Bonilla, E. V., Michiardi, P., and Filippone, M. Random feature expansions for deep gaussian processes. In *International Conference on Machine Learning*, pp. 884–893. PMLR, 2017.

Dai, B., Dai, H., Gretton, A., Song, L., Schuurmans, D., and He, N. Kernel exponential family estimation via doubly dual embedding. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2321–2330, 2019.

Damianou, A. and Lawrence, N. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

Daniely, A. Sgd learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems*, pp. 2422–2430, 2017.

Dao, T., De Sa, C. M., and Ré, C. Gaussian quadrature for kernel features. In *Advances in Neural Information Processing Systems*, pp. 6109–6119, 2017.

De Boor, C. A practical guide to splines, revised edition, 2001.

De G. Matthews, A. G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. Gpflow: A gaussian process library using tensorflow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

Dellaert, F. and Kaess, M. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

Djugash, J. A. Geolocation with range: Robustness, efficiency and scalability. 2010.

Dolan, E. D. and Moré, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, January 2002.

Dong, J. and Lv, Z. minisam: A flexible factor graph non-linear least squares optimization framework. *arXiv preprint arXiv:1909.00903*, 2019.

Drineas, P. and Mahoney, M. W. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6 (Dec):2153–2175, 2005.

Droeschel, D. and Behnke, S. Efficient continuous-time slam for 3d lidar-based online mapping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9. IEEE, 2018.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Durrant-Whyte, H. and Bailey, T. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

Duvenaud, D. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

Evoluationary computation pages — the function testbed. *Laappeenranta University of Technology*. URL http://www.it.lut.fi/ip/evo/functions/functions.html.

Fang, K.-T. and Li, R.-Z. Some methods for generating both an NT-net and the uniform distribution on a Stiefel manifold and their applications. *Computational Statistics & Data Analysis*, 24(1):29–46, 1997.

Felix, X. Y., Suresh, A. T., Choromanski, K. M., Holtmann-Rice, D. N., and Kumar, S. Orthogonal Random Features. In *Advances in Neural Information Processing Systems*, pp. 1975–1983, 2016.

Fine, S. and Scheinberg, K. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.

Fleps, M., Mair, E., Ruepp, O., Suppa, M., and Burschka, D. Optimization based imu camera calibration. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3297–3304. IEEE, 2011.

Forbes, P. G. and Lauritzen, S. Linear estimating equations for exponential families with application to gaussian linear concentration models. *Linear Algebra and its Applications*, 473:261–283, 2015.

Forrester, A. I. J., Sobester, A., and Keane, A. J. *Engineering Design via Surrogate Modelling - A Practical Guide*. J. Wiley, 2008.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Furgale, P., Barfoot, T. D., and Sibley, G. Continuous-time batch estimation using temporal basis functions. In *2012 IEEE International Conference on Robotics and Automation*, pp. 2088–2095. IEEE, 2012.

Gandy, S., Recht, B., and Yamada, I. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pp. 7576–7586, 2018.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

Genz, A. Methods for generating random orthogonal matrices. *Monte Carlo and Quasi-Monte Carlo Methods*, pp. 199–213, 1998.

Genz, A. and Monahan, J. Stochastic integration rules for infinite regions. *SIAM journal on scientific computing*, 19(2):426–439, 1998.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

Grasedyck, L., Kluge, M., and Krämer, S. Alternating directions fitting (adf) of hierarchical low rank tensors. *Preprint*, 149, 2013.

Grasedyck, L., Kluge, M., and Krämer, S. Alternating least squares tensor completion in the tt-format. *arXiv preprint arXiv:1509.00311*, 2015a.

Grasedyck, L., Kluge, M., and Kramer, S. Variants of alternating least squares tensor completion in the tensor train format. *SIAM Journal on Scientific Computing*, 37(5): A2424–A2450, 2015b.

Gutmann, M. and Hirayama, J.-i. Bregman divergence as general framework to estimate unnormalized statistical models. *arXiv preprint arXiv:1202.3727*, 2012.

Hartley, R. and Zisserman, A. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

Hillar, C. J. and Lim, L.-H. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

Hyvärinen, A. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 01 2005.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31: 8571–8580, 2018.

Jitkrittum, W., Xu, W., Szabó, Z., Fukumizu, K., and Gretton, A. A linear-time kernel goodness-of-fit test. In *Advances in Neural Information Processing Systems*, pp. 262–271, 2017.

Kaess, M., Ranganathan, A., and Dellaert, F. isam: Fast incremental smoothing and mapping with efficient data association. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1670–1677. IEEE, 2007.

Kaess, M., Ranganathan, A., and Dellaert, F. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

Kaess, M., Ila, V., Roberts, R., and Dellaert, F. The bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Algorithmic Foundations of Robotics IX*, pp. 157–173. Springer, 2010.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

Kingma, D. P. and Cun, Y. L. Regularized estimation of image statistics by score matching. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 23*, pp. 1126–1134. Curran Associates, Inc., 2010. URL http://papers.nips.cc/paper/4060-regularized-estimation-of-image-statistics-by-score-matching.pdf.

Ko, C.-Y., Batselier, K., Yu, W., and Wong, N. Fast and accurate tensor completion with total variation regularized tensor trains. *arXiv preprint arXiv:1804.06128*, 2018.

Kolda, T. G. and Bader, B. W. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009a.

Kolda, T. G. and Bader, B. W. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009b.

Kressner, D., Steinlechner, M., and Vandereycken, B. Low-rank tensor completion by riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Le, Q., Sarlós, T., and Smola, A. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the International Conference on Machine Learning*, 2013.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Li, X., Ye, Y., and Xu, X. Low-rank tensor completion with total variation for visual data inpainting. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Li, Z., Ton, J.-F., Oglic, D., and Sejdinovic, D. Towards a unified analysis of random fourier features. In *International Conference on Machine Learning*, pp. 3905–3914, 2019.

Lin, L., Drton, M., and Shojaie, A. Estimation of high-dimensional graphical models using regularized score matching. *Electronic Journal of Statistics*, 10(1):806–854, 2016. ISSN 1935-7524. doi: 10.1214/16-ejs1126. URL http://dx.doi.org/10.1214/16-EJS1126.

Liu, J., Musialski, P., Wonka, P., and Ye, J. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220, 2013.

Loan, C. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(1-2):85–100, November 2000.

Lyu, S. Interpretation and generalization of score matching. *arXiv preprint arXiv:1205.2629*, 2012.

Malach, E., Yehudai, G., Shalev-Shwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. *arXiv preprint arXiv:2002.00585*, 2020.

Mardia, K. V., Kent, J. T., and Laha, A. K. Score matching estimators for directional distributions, 2016.

Mei, S. and Montanari, A. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

Montgomery, D. C. *Design and Analysis of Experiments*. John Wiley & Sons, 2006. ISBN 0470088109.

Monti, R. P. and Hyvärinen, A. A unified probabilistic model for learning latent factors and their connectivities from high-dimensional data, 2018.

Munkhoeva, M., Kapushev, Y., Burnaev, E., and Oseledets, I. Quadrature-based features for kernel approximation. In *Advances in Neural Information Processing Systems*, pp. 9147–9156, 2018.

Mur-Artal, R. and Tardós, J. D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

Nocedal, J. and Wright, S. J. Numerical optimization, second edition. *Numerical optimization*, pp. 497–528, 2006.

Oseledets, I. and Tyrtyshnikov, E. Tt-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

Oseledets, I. V. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33 (5):2295–2317, 2011.

Owen, A. B. Latin supercube sampling for very high-dimensional simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):71–102, 1998.

Phien, H. N., Tuan, H. D., Bengua, J. A., and Do, M. N. Efficient tensor completion: Low-rank tensor train. *arXiv preprint arXiv:1601.01083*, 2016.

Potechin, A. and Steurer, D. Exact tensor completion with sum-of-squares. *arXiv preprint arXiv:1702.06237*, 2017.

Qin, M., Li, Z., Chen, S., Guan, Q., and Zheng, J. Low-rank tensor completion and total variation minimization for color image inpainting. *IEEE Access*, 8:53049–53061, 2020.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pp. 1177–1184, 2008.

Rasmussen, C. E. and Ghahramani, Z. Infinite mixtures of gaussian process experts. In *In Advances in Neural Information Processing Systems 14*, pp. 881–888. MIT Press, 2001.

Rasmussen, C. E. and Nickisch, H. Gaussian processes for machine learning (gpml) toolbox. *J. Mach. Learn. Res.*, 11:3011–3015, December 2010. ISSN 1532-4435.

Rasmussen, C. E. and Williams, C. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Recht, B., Fazel, M., and Parrilo, P. A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

Reehorst, E. T. and Schniter, P. Regularization by denoising: Clarifications and new interpretations. *IEEE Transactions on Computational Imaging*, 5(1):52–67, Mar 2019. ISSN 2573-0436. doi: 10.1109/tci.2018.2880326. URL http://dx.doi.org/10.1109/TCI.2018.2880326.

Rendall, T. and Allen, C. Multi-dimensional aircraft surface pressure interpolation using radial basis functions. *Proc. IMechE Part G: Aerospace Engineering*, 222:483 – 495, 2008. Publisher: IMechE.

Roberts, G. and Rosenthal, J. Optimal scaling for various metropolis-hastings algorithms. *Statistical Science*, 16, 11 2001. doi: 10.1214/ss/1015346320.

Rossi, S., Heinonen, M., Bonilla, E. V., Shen, Z., and Filippone, M. Sparse gaussian processes revisited: Bayesian approaches to inducing-variable approximations, 2020.

Roth, K., Lucchi, A., Nowozin, S., and Hofmann, T. Stabilizing training of generative adversarial networks through regularization. 05 2017.

Rudi, A. and Rosasco, L. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pp. 3215–3225, 2017.

Rudi, A., Carratino, L., and Rosasco, L. Falkon: An optimal large scale kernel method. In *Advances in Neural Information Processing Systems*, pp. 3888–3898, 2017.

Smola, A. J. and Schölkopf, B. Sparse greedy matrix approximation for machine learning. 2000.

Snelson, E. and Ghahramani, Z. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pp. 1257–1264. MIT press, 2005.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution, 2019.

Sriperumbudur, B., Fukumizu, K., Gretton, A., Hyvärinen, A., and Kumar, R. Density estimation in infinite dimensional exponential families, 2013.

Steinlechner, M. Riemannian optimization for high-dimensional tensor completion. *SIAM Journal on Scientific Computing*, 38(5):S461–S484, 2016.

Stone, C. J., Hansen, M., Kooperberg, C., and Truong, Y. K. Polynomial splines and their tensor products in extended linear modeling. *Ann. Statist*, 25:1371–1470, 1997.

Strathmann, H., Sejdinovic, D., Livingstone, S., Szabo, Z., and Gretton, A. Gradient-free hamiltonian monte carlo with efficient kernel exponential families, 2015.

Strömberg, E. Smoothing and mapping of an unmanned aerial vehicle using ultra-wideband sensors, 2017.

Sutherland, D. J. and Schneider, J. On the error of random fourier features. *arXiv preprint arXiv:1506.02785*, 2015.

Sutherland, D. J., Strathmann, H., Arbel, M., and Gretton, A. Efficient and principled score estimation with nystr\" om kernel exponential families. *arXiv preprint arXiv:1705.08360*, 2017.

Suzuki, T. Convergence rate of bayesian tensor estimator and its minimax optimality. In *International Conference on Machine Learning*, pp. 1273–1282, 2015.

System optimization — testroblems. *Swiss International Institute of Technology*. URL http://www.tik.ee.ethz.ch/sop/download/supplementary/testproblems/.

Teng, T., Chen, J., Zhang, Y., and Low, B. K. H. Scalable variational bayesian kernel selection for sparse gaussian process regression. In *AAAI*, pp. 5997–6004, 2020.

Thrun, S., Burgard, W., and Fox, D. *Probabilistic robotics*. MIT press, 2005.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Tobler, C. *Low-rank tensor methods for linear systems and eigenvalue problems*. PhD thesis, ETH Zurich, 2012.

Tong, C. H., Furgale, P., and Barfoot, T. D. Gaussian process gauss–newton for non-parametric simultaneous localization and mapping. *The International Journal of Robotics Research*, 32(5):507–525, 2013.

Tropp, J. A. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015. ISSN 1935-8245. doi: 10.1561/2200000048. URL http://dx.doi.org/10.1561/2200000048.

Tyrtyshnikov, E. Incomplete cross approximation in the mosaic-skeleton method. *Computing*, 64(4):367–380, 2000.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, pp. 14622–14632, 2019.

Wang, X., Marcotte, R., Ferrer, G., and Olson, E. Apriisam: real-time smoothing and mapping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2486–2493. IEEE, 2018.

Wenliang, L., Sutherland, D., Strathmann, H., and Gretton, A. Learning deep kernels for exponential family densities, 2018.

Williams, C. Computing with infinite networks. *Advances in neural information processing systems*, 9:295–301, 1996.

Williams, C. K. Computing with infinite networks. In *Advances in Neural Information Processing Systems*, pp. 295–301, 1997.

Williams, C. K. and Seeger, M. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pp. 682–688, 2001.

Wilson, A. G., Gilboa, E., Cunningham, J. P., and Nehorai, A. Fast kernel learning for multidimensional pattern extrapolation. In *NIPS*, pp. 3626–3634, 2014.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. Stochastic variational deep kernel learning. *Advances in Neural Information Processing Systems*, 29:2586–2594, 2016.

Yan, X., Indelman, V., and Boots, B. Incremental sparse gp regression for continuous-time trajectory estimation and mapping. *Robotics and Autonomous Systems*, 87:120–132, 2017.

Yang, J., Sindhwani, V., Avron, H., and Mahoney, M. Quasi-Monte Carlo feature maps for shift-invariant kernels. In *Proceedings of The 31st International Conference on Machine Learning (ICML-14)*, pp. 485–493, 2014.

Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. In *Advances in Neural Information Processing Systems*, pp. 476–484, 2012.

Yokota, T., Zhao, Q., and Cichocki, A. Smooth parafac decomposition for tensor completion. *IEEE Transactions on Signal Processing*, 64(20):5423–5436, 2016.

Yu, F. X., Kumar, S., Rowley, H., and Chang, S.-F. Compact nonlinear maps and circulant extensions. *arXiv preprint arXiv:1503.03893*, 2015.

Yu, S., Drton, M., and Shojaie, A. Graphical models for non-negative data using generalized score matching, 2018.

Yuan, L., Zhao, Q., and Cao, J. Completion of high order tensor data with missing entries via tensor-train decomposition. In *International Conference on Neural Information Processing*, pp. 222–229. Springer, 2017.

Yuan, M. and Zhang, C.-H. On tensor completion via nuclear norm minimization. *Foundations of Computational Mathematics*, 16(4):1031–1068, 2016.

Zhang, Z., Weng, T.-W., and Daniel, L. Big-data tensor recovery for high-dimensional uncertainty quantification of process variations. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 7(5):687–697, 2017.

Zhao, L. Y., Cao, J., et al. High-dimension tensor completion via gradient-based optimization under tensor-train format. *arXiv preprint arXiv:1804.01983*, 2018.

Zhao, Q., Zhang, L., and Cichocki, A. Bayesian cp factorization of incomplete tensors with automatic rank determination. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1751–1763, 2015.