Gaussian Processes on Distributions based on Regularized Optimal Transport

Students:
Louis ALLAIN
Léonard GOUSSET
Julien HEURTIN

Referents :
Brian Staber
Sébastien Da Veiga

Graduation Project

March 2024

Abstract

After giving reminders on kernel methods, optimal transport, regularized optimal transport and the Sinkhorn algorithm, we show that the kernel proposed in [1] works in a kernel ridge regression setting. We achieve similar performances as the Gaussian Processes model and explore a little beyond the toy experiment. After that, we present a public dataset, Rotor37, on which we are going to use the Sinkhorn kernel. After presenting a reference kernel that works really well for this specific dataste, we study the impact of multiple hyperparameters of the model. We show that with the correct hyperparameters, one can achieve really good performances with the Sinkhorn kernel.

Glossary

- **Algorithm**:
  An algorithm is a systematic and precise set of step-by-step instructions designed to perform a specific task or solve a particular problem. It serves as a computational procedure, outlining the sequence of operations required to achieve a desired outcome.

- **Machine Learning**:
  This a field of science where we give a problem to a machine and it learns the optimal solution to this problem. Usually this is an *optimization* problem. The machine learns the best parameters for a problem.

- **Regression**:
  Regression is a specific task in machine learning. It involves the knowledge of input and output data, and makes the machine learns how to predict the output from the input data.

- **Distribution**:
  A distribution refers to a set of all possible values and their corresponding probabilities. It describes how the values of a variable are spread or distributed across different outcomes.

- **Kernel**:
  The word kernel is used in a variety of mathematics domains. In our use case it is simply a (positive definite) function that takes two arguments and outputs a real number. It can be seen as a similarity measure between those two objects.

- **Transport**:
  If you have two histograms, you want to know how to "transform" one into the other. They are both density measures, therefore their integral is one. You slice the histograms (or density) into units and the transport correspond to the way you map each unit of the first hsitogram to units of the second. There are infinite ways to do so.

- **Optimal Transport**:
  The *optimal* transport is the way you map one histogram to another that minimizes a given cost function. This function essentially tells you how much does it costs to map every unit of the first density to the second.

- **Regularization**:
  In machine learning, the word regularization refers to the reformulation of the initial optimization problem that adds a constraint (usually on the models parameter). It is very useful to bypass computational difficulties one may encounter in machine learning.

- **Sinkhorn**:
  The Sinkhorn algorithm, is an iterative numerical method used for regularizing and solving optimal transport problems. It greatly helps with computational performances.

- **Dual formulation**:
  In a constraint optimization problem, the dual formulation introduces "dual" variables associated with the constraint. Dual formulations are often useful in optimization theory and algorithms, providing insights into the problem structure and facilitating efficient solution techniques.

- **Complexity**:
  The complexity of an algorithm is the way the computation time scales with the data the algorithm receives. Usually, one aims for the smallest complexity, such as a linear or logarithmic complexity.

Contents

Part I
Methodological Part

1 Introduction

In recent years, the intersection of Gaussian Processes (GPs) and Optimal Transport (OT) has emerged as a fertile ground for innovation in statistical learning. Gaussian Processes, known for their flexibility and robustness, are a cornerstone in the field of machine learning, providing a probabilistic approach to learning in kernel-based models. On the other hand, Optimal Transport offers a powerful framework for comparing probability distributions, with applications ranging from economics to image processing.

The development presented in François Bachoc et al.'s paper[1] introduces a significant advancement in statistical analysis and algorithm design through the creation of a new kernel using Gaussian Processes, enhanced by techniques from Regularized Optimal Transport. This report offers a comprehensive overview of this novel kernel, detailing its conceptual foundation and potential applications in various scientific and technological fields.

In this project, we are collaborating with Safran, a leading aerospace manufacturer, to pioneer innovative approaches in designing aircraft engine components, specifically focusing on the blades of a propeller. Our objective is to leverage advanced machine learning techniques, particularly in the realm of regression analysis, to optimize the efficiency of these critical components. The target variable in our machine learning model is the efficiency metric of the blade, a quantifiable measure of performance under various operational conditions. Intriguingly, the feature variable is the probabilistic distribution of the blade's structure, represented as a point cloud in a three-dimensional space ($\mathbb{R}^3$). This representation is not just a mere geometrical depiction but encapsulates the intricate variability and physical characteristics of the blade. Through the analysis of this relationship between the shape distribution and efficiency, our goal is to identify underlying patterns and gain insights that could inform the engineering of propeller blades with enhanced efficiency and durability. This research represents a significant step in aerospace engineering, merging advanced statistical methods with engineering challenges, and contributes to the ongoing evolution of aircraft propulsion technology.

To approach François Bachoc et al.'s paper with a comprehensive understanding, we will first elucidate the methods of kernels in statistical learning, with a specific focus on kernel ridge regression (see sec. 2). Additionally, we will revisit the genesis of optimal transport and its more recent formulation in an entropic framework, along with its resolution using the Sinkhorn algorithm (see sec. 3). Subsequently, we will provide a detailed exposition of the kernel developed within the paper under examination (see sec. 4).

2 Kernel methods

2.1 Framework

2.1.1 Regression

Regression is a fundamental task in machine learning. Let $\mathcal{X}$ be a nonempty set, often called the *input space* and let $f\colon \mathcal{X} \to \mathcal{Y}$. Because we are focusing on a regression problem, the *target* or *input space* is going to be continuous, meaning $\mathcal{Y} \subset \mathbb{R}$. Assume, for $n \in \mathbb{N}$, that we are given the following training data $\mathcal{D}_n = ((x_i, y_i))_{1 \leq i \leq n} \in (\mathcal{X} \times \mathbb{R})^n$ such that:

$$y_i = f(x_i) + \varepsilon_i, \quad i = 1, \dots n$$

where ε_i are random variable of mean 0 and variance σ^2. They represent the "noise" of the data, the variability of y_i that is not explained by the input data. The regression task is to estimate the unknown function f from the training data $\mathcal{D}_n$. The function f is called the regression function and is the conditional mean of the output knowing an input:

$$f(x) = \mathbb{E}\left[y|x\right]$$

2.1.2 Positive definite kernel

A primary constituent of the methods discussed here is the positive definite kernel. The kernel is a way to measure similarity in the input space $\mathcal{X}$. We remind that this input space can be anything, $\mathbb{R}$ for tabular data, the set of words for natural language processing, images for computer vision or the set of distributions as in our context. Here we give a definition of a positive definite kernel and a matrix interpretation of it. Then we move on to properties and examples.

Definition 1. *Let $\mathcal{X}$ be a nonempty set. A symmetric function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **positive definite kernel**, if for any $n \in \mathbb{N}, (x_1, \dots x_n) \subset \mathcal{X}$ and $\forall\, (c_1, \dots c_n) \subset \mathbb{R}$,*

$$\sum_{i=1}^{n} c_i c_j k\left(x_i, x_j\right) \geq 0$$

Remark 1. *We consider here only real-valued kernel, which is why k needs to be symmetric. Results exists for complex-valued kernels but they are out of scope here.*

A few example of kernels are given below.

Example 1 (Linear kernel)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$. The linear kernel $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the most simple kernel and is defined by*

$$k\left(x, x'\right) = x^\intercal x', \quad x, x' \in \mathcal{X}$$

Example 2 (Polynomial kernel)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$. For $c > 0$ and $m \in \mathbb{N}$, the polynomial kernel $k_{c,m}\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$k_{c,m}\left(x, x'\right) = \left(x^\intercal x' + c\right)^m, \quad x, x' \in \mathcal{X}$$

Example 3 (Gaussian kernel)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$. For $\gamma > 0$, the Gaussian kernel $k_\gamma\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$k_\gamma\left(x, x'\right) = e^{-\frac{\|x - x'\|^2}{\gamma^2}}, \quad x, x' \in \mathcal{X}$$

where $\|.\|$ can be any norm on $\mathbb{R}^d$.

Remark 2. *For simplicity, examples here are only given for kernels on $\mathcal{X} \subset \mathbb{R}^d$. But one strength of kernel methods is that the kernel can be defined over any nonempty set $\mathcal{X}$, for instance the space of distribution as it used here. From there on, any model can be used with the defined kernel. The kernel's role is to measure the similarity in the input space $\mathcal{X}$.*

This definite positive property is essential as it transforms complex machine learning problems into convex optimisation problems, for which we have tools to solve. Consequently they are widely used in numerous machine learning methods. Quite surprisingly they arise in different machine learning framework. In the Bayesian paradigm they appear as covariance matrices in Gaussian Processes. But they also emerge in the more traditional frequentist paradigm as a way to take into account non linearity without adding too much complexity, subject of the next section 2.2.

2.2 Reproducing Kernel Hilbert Space

To estimate the regression function, numerous method have been developed. Most of them works well for the linear case. However, in the real world arise dependencies and relations that are non linear. Reproducing Kernel Hilbert Spaces (RKHS) methods combines the two worlds, allowing us to analyse non linear relationships in a linear setting. The kernel (a positive definite kernel as mentioned in paragraph 2.1.2) corresponds to a dot product in a high dimensional space, often called the *feature space*. In this space, the algorithms and methods used are linear. As long as we can express the calculations using the kernel, none of the computations has to be performed in the complex feature space. We assume from now on that we are given a nonempty input space $\mathcal{X}$ and a positive definite kernel k. We start be defining the *feature space* and *feature map* notions.

Definition 2. *Let $\mathcal{X}$ be a nonempty set. We call $\mathcal{H} \coloneqq \{f \colon \mathcal{X} \to \mathbb{R}\}$ the **feature space**.*

Definition 3. *A **feature map** is an application mapping input space objects into the feature space:*

$$\Phi \colon \mathcal{X} \to \mathcal{H}$$

To execute such a computation, we need the kernel to satisfy, for a given feature map Φ:

$$\forall x, x' \in \mathcal{X}^2, \quad k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}} \tag{1}$$

It is shown in Schölkopf et al.[2] that the set of kernels satisfying relation 1, meaning kernels that admit a representation as a dot product in a feature space, is equal to the set of positive definite kernels studied in 2.1.2. This shows that the right class of kernel to study is the definite positive ones. The construction of kernels from feature maps is developped in Schölkopf et al.[2]. We now present spaces related to such a kernel and expose strong theorem about those spaces and kernels.

Definition 4. *Let $\mathcal{X}$ a nonempty set and k a positive definite kernel on $\mathcal{X}$. A Hilbert space $\mathcal{H}_k$ of function over $\mathcal{X}$ endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_k}$ is a **Reproducing Kernel Hilbert Space** of reproducing kernel k if:*

1. $\forall x \in \mathcal{X},\ k(\cdot, x) \in \mathcal{H}_k$
2. $\forall f \in \mathcal{H}_k,\ \forall x \in \mathcal{X},\ \ f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}_k}$ (Reproducing property)

Remark 3. *For all $x \in \mathcal{X},\ k(\cdot, x) : \mathcal{X} \to \mathbb{R}$ is the canonical feature map of x, because inside $\mathcal{H}_k$, k writes as a scalar product: $k(x, y) = \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}_k}$*

Next comes a powerful theorem that tight things up between a RKHS and a kernel:

Theorem 1 (Moore-Aronszajn)**.** *The association between a kernel and a RKHS is unique:*

- *For every positive definite kernel k there exist a uniquely associated RKHS $\mathcal{H}_k$.*
- *In the other way, for every RKHS $\mathcal{H}$, there exist a unique kernel k with the reproducing property. And this kernel is positive definite.*

The application of this theorem in a machine learning context is extremely powerful, it applies in optimization problem that arises in machine learning:

Theorem 2 (Representer Theorem)**.** *We consider the following machine learning optimization problem:*

- *Let $\mathcal{X}$ a nonempty set and k a positive definite kernel on $\mathcal{X}$. We denote by $\mathcal{H}_k$ the (unique) associated Reproducing Kernel Hilbert Space.*

- *We keep the same notation for the data:* $\mathcal{D}_n = ((x_i, y_i))_{1 \leq i \leq n} \in (\mathcal{X} \times \mathbb{R})^n$
- *Let* $\Omega \colon [0, +\infty[\to \mathbb{R}$ *a strictly monotonic increasing function.*
- *Let* l *be a loss function.*
- *We also consider* $\lambda > 0$ *a real positive constant, which plays the role of weight of the regularization.*

The Representer Theorem *states that for any solution* f^* *of the optimization problem that is minimizing the empirical risk associated to the loss function, regularized with the function* Ω*, over the space of function* $\mathcal{H}_k$*:*

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} l\left(y_i, f(x_i)\right) + \lambda \Omega\left(\|f\|_{\mathcal{H}_k}\right)$$

there exists $\alpha_i \in \mathbb{R}$ *for* $i = 1, \ldots, n$ *such that for any* $x \in \mathcal{X}$*:*

$$f^*(x) = \sum_{i=1}^{n} \alpha_i k\left(x, x_i\right)$$

Remark 4. *Kernel methods and RKHS were made very popular because of the* ***kernel trick****. This* trick *consists in replacing in all computations the use of the dot product and the feature map with the kernel.*

Therefore, kernel methods enable to compute linear algorithm on non linear data without having to compute anything in the Hilbert space itself, thanks to the kernel trick. It makes such methods very powerful and useful in modern machine learning. Another huge benefit of such methods is given by the Riesz Representation theorem.

Theorem 3 (Riesz Representation Theorem)**.** *Let* $\mathcal{H}$ *be a Hilbert space and* f *a continuous linear functional defined over* $\mathcal{H}$*. Then, there exist a unique element* y *of* $\mathcal{H}$ *such that for all* $x \in \mathcal{H}$*:*

$$f\left(x\right) = \langle x, y \rangle_{\mathcal{H}}$$

The consequences of this theorem are of great magnitude. By specifying the three terms, being the loss function, the regularization function and the positive definite kernel, you can perform a lot of known models such as logistic regression, ridge regression and even kernel-PCA and kernel-Nearest Neighbour. To a certain extend Gaussian Process and Neural Networks such as CNN, RNN, Transformers and GPTs can be linked to RKHS and more broadly kernel methods. In the following section we will develop on the Kernel Ridge Regression.

2.3 Kernel Ridge Regression

Ridge Regression is a usual linear regression performed with a regularization on the coefficient of the regression. Therefore the input space is a subset of $\mathbb{R}^d$. The **Kernel** Ridge Regression is an extension of the Ridge Regression. Indeed, the Ridge Regression is using the linear kernel seen here 1. Generalizing this with any kernel gives us the general formulation of the Kernel Ridge Regression. This generalization allows for this method to be used in a broader range of settings such as the one that interests us here, a regression over the set of distribution.

Example 4. *We present the results of the* ***Kernel Ridge Regression****. We keep the same notions as above and consider the quadratic loss function. The optimization problem thus writes:*

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \|f\|_{\mathcal{H}_k}^2$$

We know with the Representer Theorem that the optimization problem writes in a matrix formulation:

$$\min_{\alpha \in \mathbb{R}^n} \|Y - K\alpha\|^2 + \lambda \alpha^\intercal K \alpha$$

To solve this optimization problem, one must derive this expression with regard to α*. Writing* $\hat{\alpha} =$

$(\hat{\alpha}_i, \ldots, \hat{\alpha}_n) \in \mathbb{R}^n$ *we have the Kernel Ridge estimator for all* x *in* $\mathcal{X}$*:*

$$\hat{f} = \sum_{i=1}^{n} \hat{\alpha}_i k(\cdot, x_i)$$

This estimator is extremely powerful. Indeed, the regression can be performed on any set of data $\mathcal{X}$ and all the benefits of using the Ridge Regression applies. A few remark must be made on this model.

Remark 5. *The first remark there is to make is that this Kernel Ridge Regression is an extension of the regression with ridge regularization model. If we use the linear kernel we immediately have* $K = X^\intercal X$*. We found the exact same formulation as the regression with ridge regularization.*

Remark 6. *When* λ *tends to* 0 *(with the linear kernel) we found the usual linear regression. When* λ *tends to* ∞*, we found* $\hat{\alpha} = 0_{\mathbb{R}^d}$*. The regularization allows for a solution that is unique and that always exists.*

The Kernel Ridge Regression is both a versatile and easy model to use in a regression setting. Nevertheless, it is sometime necessary to turn towards more complex and high performance models. Specifically, using Bayesian model allows one to use prior knowledge to enhance the model. Gaussian Processes [3, 4] are such models. They also rely on a positive definite kernel as covariance matrix. Therefore the benefit of kernel methods, that is, be usable in any context if one can find the right kernel, still applies for Gaussian Processes.

3 Optimal Transport

3.1 Introduction to Optimal Transport : The Monge formulation

The Monge Transport Problem, named after 18th-century mathematician Gaspard Monge, represents a seminal development in Optimal Transport. Originally conceptualized to address practical challenges in mass transportation, Monge's formulation has evolved into a comprehensive theoretical framework. It involves finding the most cost-effective strategy to transport mass from one distribution to another.

Consider a scenario with n distinct sources and n distinct destinations, each source-destination pair incurs a specific transportation cost, represented in a cost matrix $(C_{i,j})_{i\in[\![n]\!],j\in[\![n]\!]}$, where i and j index the sources and destinations, respectively. If we denote σ a bijection from the set of sources to the set of destinations, and $\mathrm{Perm}(n)$ the set of all possible permutations of n elements, the Monge Transport Problem seeks to minimize the total transportation cost, formalized as:

$$\min_{\sigma\in\mathrm{Perm}(n)} \sum_{i=1}^{n} C_{i,\sigma(i)} \tag{2}$$

Optimal Transport (OT) can be conceptualized across various scenarios, including continuous-continuous, discrete-continuous, and discrete-discrete distributions. Each scenario presents unique challenges and methodologies within the framework of OT. However, for the purpose of our study, we will concentrate on the discrete-discrete case. This focus aligns with the scenarios where both the source and destination distributions are discrete, as is often encountered in practical applications relevant to our field of study.

Monge problem between discrete measures : Considering two discretes measures

$$\alpha = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \text{and} \quad \beta = \sum_{j=1}^{m} b_j \delta_{y_j},$$

the Monge problem seeks a map $T : \{x_1, \ldots, x_n\} \to \{y_1, \ldots, y_m\}$ that associates to each point x_i a single point y_j, and which must push the mass of α toward the mass of β. The map must satisfy the following condition for every $j \in m$:

$$\forall j \in m, \quad b_j = \sum_{i:T(x_i)=y_j} a_i \tag{3}$$

which can be written in compact form as $T\#\alpha = \beta$. This map must minimize a transportation cost parameterized by a function $c(x, y)$ defined for points $(x, y) \in X \times Y$, expressed as:

$$\min_T \sum_i c(x_i, T(x_i)) \quad : \quad T\#\alpha = \beta. \tag{4}$$

Remark 7. *This problem, however, does not always have a unique solution. Particularly, when $n = m$ and weights are uniform ($a_i = b_j = 1/n$), it simplifies to an optimal matching problem 2, representable with a cost matrix C_{ij}. On the other hand, if $n \neq m$ or weights vary, a Monge map might not exist, necessitating more complex or generalized approaches.*

Such variations underscore the challenges in addressing the Monge problem for diverse distribution scenarios. It is in this context that the Kantorovich formulation becomes particularly relevant.

3.2 Kantorovich's Relaxation

Leonid Kantorovich, a mathematician in the 20-th century has developed a new formulation for the problem of optimal transport. His formulation was conceived to address critical constraints in the Monge problem, notably the challenge of finding feasible solutions that adhere to mass conservation. Kantorovich's method effectively manages the combinatorial complexity of the assignment problem and overcomes the nonconvex nature of feasible sets in the Monge formulation. It accomplishes this by introduces 'mass splitting', allowing the distribution of source mass to multiple destinations, enhancing both flexibility and computational feasibility.

This flexibility is encoded using a coupling matrix $\mathbf{P} \in \mathbb{R}_+^{n \times m}$, where $\mathbf{P}_{i,j}$ describes the amount of mass flowing from bin i toward bin j, or from mass at x_i toward y_j in discrete measures. Admissible couplings are characterized as:

$$\mathbf{U}(\mathbf{a}, \mathbf{b}) \coloneqq \left\{\mathbf{P} \in \mathbb{R}_+^{n \times m} : \mathbf{P}\mathbb{1}_m = \mathbf{a} \text{ and } P^T\mathbb{1}_n = \mathbf{b}\right\}$$

with matrix-vector notation:

$$\mathbf{a} = \left(\sum_i \mathbf{P}_{i,j}\right)_i \in \mathbb{R}^n \text{ and } \mathbf{b} = \left(\sum_i \mathbf{P}_{i,j}\right)_j \in \mathbb{R}^m.$$

The set $U(a, b)$ is a convex polytope, defined by $n + m$ equality constraints.

In contrast to the Monge formulation's asymmetric nature, the Kantorovich formulation exhibits inherent symmetry. This is demonstrated by the relationship between the coupling matrix P and its transpose P^T: P is a member of $U(a, b)$ if and only if P^T is included in $U(b, a)$. This lead to the Kantorovich optimal transport problem defined as :

$$L_{\mathbf{C}(\mathbf{a},\mathbf{b})} \coloneqq \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle \coloneqq \sum_{i,j} \mathbf{C}_{i,j}\mathbf{P}_{i,j}. \tag{5}$$

Remark 8. *In the context of Optimal Transport, the linear program in Equation 5 is solved to derive a transportation plan $\mathbf{P}^*$, quantifying the goods $\mathbf{P}_{i,j}$ to be transported from the sources i to the destinations j. For permutation matrices as couplings, $\mathbf{P}_\sigma$ represents the permutation matrix for $\sigma \in Perm(n)$, with $(\mathbf{P}_\sigma)_{i,j} = \frac{1}{n}$ if $j = \sigma(i)$, else 0. This formulation allows recasting the assignment problem as a Kantorovich problem, where couplings are restricted to permutation matrices. The set of permutation matrices is a subset of the Birkhoff polytope $U(\frac{1}{n}, \frac{1}{n})$, illustrating that the Kantorovich problem's minimum is achieved with permutation matrices when dealing with uniform measures.*

Remark 9 (Kantorovich problem between discrete measures)**.** *For discrete measures α, β of the form 4, we store in the matrix C all pairwise costs between points in the supports of α, β, namely $C_{i,j} \coloneqq c(x_i, y_j)$, to define*

$$L_c(\alpha, \beta) \coloneqq L_C(\mathbf{a}, \mathbf{b}). \tag{6}$$

Therefore, the Kantorovich formulation of optimal transport between discrete measures is the same as the problem between their associated probability weight vectors $\mathbf{a}, \mathbf{b}$ except that the cost matrix C depends on the support of α and β.

Building upon our focus on the discrete measure in the Kantorovich formulation, our project further evolves to embrace Regularized Optimal Transport. This advanced variation of Optimal Transport is particularly suited to the practical constraints and computational demands we face in analyzing high-dimensional data, such as point clouds in $\mathbb{R}^d$. Regularized Optimal Transport offers improved computational efficiency and robustness, crucial for handling the complexity of probability distributions in our study. The integration of regularization techniques into Optimal Transport thus aligns seamlessly with our objective of utilizing the kernel for efficient and accurate comparison of these distributions.

3.3 Regularized Optimal Transport

To resolve this optimal transport problem between two discrete measures, several algorithms are available, including the Hungarian algorithm and the simplex algorithm. However, these algorithms can become very time-consuming, with a worst-case complexity of $O(n^3 log(n))$ when $n = m$. To overcome those computational issue, the regularized OT introduces an entropic regularization penalty (see sec. 3.3.1). To solve this minimization problem, we're going to use the Sinkhorn algorithm (see sec. 3.3.2).

3.3.1 Entropic Regularization and formulation

Let $\mathbf{P}$ be a coupling matrix, its discrete entropy is defined as follows :

$$H(\mathbf{P}) \stackrel{\text{def}}{=} -\sum_{i,j} \mathbf{P}_{i,j}(\log(\mathbf{P}_{i,j}) - 1)$$

The term $-H(\mathbf{P})$ is thus utilized here as a regularization term to obtain approximate solutions for the optimal transport problem. We then obtain the following minimization problem:

$$L_C^\varepsilon(\mathbf{a}, \mathbf{b}) \stackrel{\text{def}}{=} \min_{\mathbf{P} \in U(a,b)} \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon H(\mathbf{P})$$

Since this minimizing transport is an ε-strongly convex function, it has a unique optimal solution. We can write it solution $\mathbf{P}_\varepsilon$. One can verify the following convergence:

$$L_C^\varepsilon(\mathbf{a}, \mathbf{b}) \xrightarrow[\varepsilon \to 0]{} L_C(\mathbf{a}, \mathbf{b})$$

As ε approaches 0, we approach Kantorovich's formulation. The goal is to determine an optimal value for ε. When ε is small, the solution tends to converge to the maximum entropy optimal transport coupling. However, with increasing ε, the optimal transport coupling becomes denser, signifying a greater density of non-zero entries. This densification leads to accelerated computational algorithms and improved statistical convergence[5]. We can observe this densification as ε increases in the following chart:

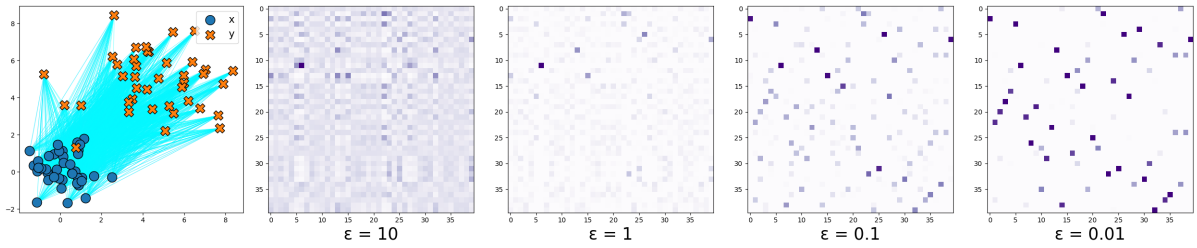


Figure 2: The influence of ε on the transport plan is illustrated. On the left are the two distributions targeted for transport optimization, while the right displays four distinct plans corresponding to different ε values.

The Kullback–Leibler divergence permit to defines the unique solution $\mathbf{P}_\varepsilon$ as a projection onto $\mathbf{U}(\mathbf{a}, \mathbf{b})$ using the Gibbs kernel associated with the cost matrix $\mathbf{C}$. The Kullback-Leibler divergence between the coupling matrix $\mathbf{P}$ and a kernel $\mathbf{K}$ is defined as:

$$\mathrm{KL}(\mathbf{P}|\mathbf{K}) \stackrel{\text{def}}{=} \sum_{i,j} \mathbf{P}_{i,j} \log\left(\frac{\mathbf{P}_{i,j}}{\mathbf{K}_{i,j}}\right) - \mathbf{P}_{i,j} + \mathbf{K}_{i,j}$$

where

$$\mathbf{K}_{i,j} \stackrel{\text{def}}{=} \exp -\frac{\mathbf{C}_{i,j}}{\varepsilon}\,.$$

We then have the unique solution as:

$$\mathbf{P}_\varepsilon = \text{Proj}^{\text{KL}}_{\mathbf{U}(\mathbf{a},\mathbf{b})}(\mathbf{K}) \stackrel{\text{def}}{=} \underset{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})}{\arg\min}\ \text{KL}(\mathbf{P}|\mathbf{K})$$

In the next section, the Sinkhorn algorithm used to solve this optimization problem is presented.

3.3.2 Sinkhorn's Algorithm

Another way of finding the solution is to write the Lagrangian Λ^ε_C associated to the problem $L^\varepsilon_C(\mathbf{a},\mathbf{b})$. The two constraints are the mass conservation of $\mathbf{a}$ and $\mathbf{b}$, inherent to $\mathbf{U}(\mathbf{a},\mathbf{b})$. We then obtain the following Lagrangian formulation:

$$\Lambda^\varepsilon_{\mathbf{C}}(\mathbf{P},\mathbf{f},\mathbf{g}) = \langle \mathbf{P},\mathbf{C}\rangle - \varepsilon H(\mathbf{P}) - \langle \mathbf{f}, \mathbf{P}\mathbb{1}_m - \mathbf{a}\rangle - \langle \mathbf{g}, \mathbf{P}^\intercal \mathbb{1}_n - \mathbf{b}\rangle$$

And the unique solution has the form:

$$\mathbf{P}_{i,j} = e^{\frac{\mathbf{f}_i}{\varepsilon}} e^{-\frac{\mathbf{C}_{i,j}}{\varepsilon}} e^{\frac{\mathbf{g}_j}{\varepsilon}}, \qquad \forall (i,j) \in [\![n]\!] \times [\![m]\!]$$

By using nonnegative vectors u and v and the Gibbs kernel, we can write this solution as:

$$\mathbf{P}_{i,j} = \mathbf{u}_i \mathbf{K}_{i,j} \mathbf{v}_i, \qquad \forall (i,j) \in [\![n]\!] \times [\![m]\!]$$

And we can write this as a matrix scaling:

$$\mathbf{P}_{i,j} = \text{diag}(\mathbf{u})\mathbf{K}\text{diag}(\mathbf{v})$$

Under the mass conservation constraints, with $\circ$ the Hadamard-product:

$$\mathbf{u} \circ (\mathbf{K}\mathbf{v}) = \mathbf{a}\,,$$

$$\mathbf{v} \circ (\mathbf{K}^\intercal \mathbf{u}) = \mathbf{b}\,.$$

Finally, to find the solution, we are going to use the Sinkhorn algorithm, described as follows:

$\mathbf{u} \leftarrow \mathbf{1}_n$
$\mathbf{v} \leftarrow \mathbf{1}_m$
$\mathbf{P} \leftarrow \text{diag}(\mathbf{u})\mathbf{K}\text{diag}(\mathbf{v})$
while $\mathbf{P}$ change **do**
  $\mathbf{u} \leftarrow \frac{\mathbf{a}}{\mathbf{K}\mathbf{v}}$
  $\mathbf{v} \leftarrow \frac{\mathbf{b}}{\mathbf{K}^T\mathbf{u}}$
  $\mathbf{P} \leftarrow \text{diag}(\mathbf{u})\mathbf{K}\text{diag}(\mathbf{v})$
end while

The convergence of the Sinkhorn algorithm is ensured, as demonstrated by Franklin and Lorenz[6]. The complexity of each iteration of this algorithm is $O(n^2)$.

To conclude, by incorporating the entropy of the transport plan as a regularization factor within the Kantorovich minimization problem, it emerges that solving the discrete optimal transport problem involves utilizing iterative matrix multiplications and elementwise divisions. This algorithm is therefore very efficient because it is easily parallelizable and can be run on a GPU.

4 Suggested Kernel

4.1 A kernel based on optimal transport

The authors in the paper [1] use the Sinkhorn potentials to measure a distance between distributions and then create a kernel using this distance.

First of all, lets consider a measure $\mathcal{U}$ on Ω and two distributions P and Q. We then consider two transport problems, using the regularized optimal transport framework developed in 3.3. The regularized optimal transport from P to the reference measure $\mathcal{U}$ and from Q to $\mathcal{U}$. Lets denote by $\pi_{\mathcal{U}}^P$ and $\pi_{\mathcal{U}}^Q$ the optimal entropic plans of both problems. The Sinkhorn algorithm, then, gives us the optimal entropic potentials of both problems: $\left(f_{\mathcal{U}}^P, g_{\mathcal{U}}^P\right)$ for the first problem and $\left(f_{\mathcal{U}}^Q, g_{\mathcal{U}}^Q\right)$ for the second one. This can be summarized in the following way:

$$P \xrightarrow[Transport]{Reg.Opt.} \mathcal{U} \Longrightarrow \pi_{\mathcal{U}}^P \Longrightarrow \left(f_{\mathcal{U}}^P, g_{\mathcal{U}}^P\right) \tag{7}$$

$$Q \xrightarrow[Transport]{Reg.Opt.} \mathcal{U} \Longrightarrow \pi_{\mathcal{U}}^Q \Longrightarrow \left(f_{\mathcal{U}}^Q, g_{\mathcal{U}}^Q\right) \tag{8}$$

Those potentials are unique up to an additive constant. Therefore we can decide to use the centered potential. We redefine the potential as $g_{\mathcal{U}}^P = g_{\mathcal{U}}^P - \mathbb{E}\left(g_{\mathcal{U}}^P(U)\right)$ and also $g_{\mathcal{U}}^Q = g_{\mathcal{U}}^Q - \mathbb{E}\left(g_{\mathcal{U}}^Q(U)\right)$. This leads to the following equality:

$$Var_{U\sim\mathcal{U}}\left(g_{\mathcal{U}}^P(U) - g_{\mathcal{U}}^Q(U)\right) = \|g_{\mathcal{U}}^P - g_{\mathcal{U}}^Q\|_{L^2(\mathcal{U})}^2 \tag{9}$$

This equality is used to introduce the following kernel:

Theorem 4. *Let $F\colon [0,+\infty[\to \mathbb{R}$ be a continuous function and $\mathcal{U} \in \mathcal{P}_{SG}(\Omega)$. If:*

1. *$F \circ \sqrt{\cdot}$ is completely monotonous on $[0,+\infty[$*
2. *There exist a nonnegative Borel measure ν on $[0,+\infty[$ such that for $t>0, F(t) = \int_0^{+\infty} e^{-ut^2}\, d\nu(u)$*

Then

$$\begin{aligned} K\colon \mathcal{P}_{SG}(\Omega) \times \mathcal{P}_{SG}(\Omega) &\longrightarrow \mathbb{R} \\ (P,Q) &\longmapsto F\left(\|g_{\mathcal{U}}^P - g_{\mathcal{U}}^Q\|_{L^2(\mathcal{U})}\right) \end{aligned}$$

is a positive definite kernel on $\mathcal{P}_{SG}(\Omega)$.

The function F can be the usual kernel functions such as the Radial Basis Function kernel, the power exponential kernel or the Matérn kernel.

4.2 Theoretical proprieties

The paper then presents a few propositions. The first one assures that the distance on Sinkhorn potential can be dominated by a quantity that is controlled by the distance between the distributions P and Q.

Propositon 1. *Let $s \in \mathbb{N}$. Suppose Ω is compact and let $P, Q \in \mathcal{P}(\Omega)$. There exist a constant c, that depends on the dimension of Ω such that*

$$\|g_{\mathcal{U}}^P - g_{\mathcal{U}}^Q\|_{L^2(\mathcal{U})} \leq c \times \|P - Q\|_s$$

The second proposition ties the potentials to the probability distributions. It guarantees that the entropic potentials $g_{\mathcal{U}}^P$ and $g_{\mathcal{U}}^Q$ can be used to characterize the distributions P and Q.

We know that in practice we do not have access to real distributions such as P and Q. We only have a sample of those distributions, therefore:

$$P_n = \frac{1}{n}\sum_{i=1}^n \delta_{X_i} \quad \text{and} \quad Q_m = \frac{1}{m}\sum_{i=1}^m \delta_{Y_i} \tag{10}$$

It is shown that the kernel verifies the following proposition that states the consistency of the empirical kernel.

Propositon 2. *If F is continuous, then*

$$F\left(\|g_{\mathcal{U}}^{P_n} - g_{\mathcal{U}}^{Q_m}\|_{L^2(\mathcal{U})}\right) \xrightarrow[n,m\to+\infty]{a.s} F\left(\|g_{\mathcal{U}}^{P} - g_{\mathcal{U}}^{Q}\|_{L^2(\mathcal{U})}\right) \tag{11}$$

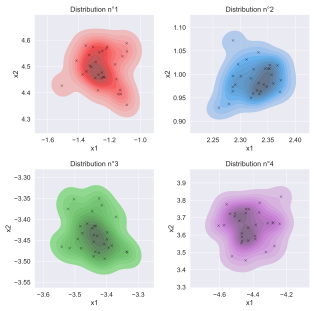
This last theorem is very important as it enable us to use this kernel in practice.

4.3 Sinkhorn kernel in practice

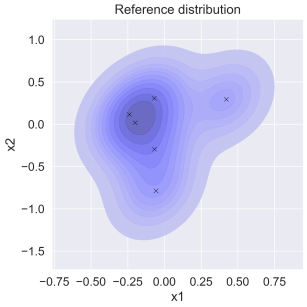
4.3.1 Reproducing the first experiment

To understand a little bit our such kernel works we will use it on a toy example from the paper. First of all lets understand our simulated data. We are going to consider a 100 two-dimensional normal distribution. Those distribution are going to have a mean vector (m_1, m_2) uniformly drawn from $[-0.3, 0.3]^2$ and the variance uniformly drawn from $[0.01^2, 0.02^2]$. We consider only isotropic distribution, therefore the covariance matrix of our distribution are going to be of the form $\sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. For each of those distribution we compute the value $Y = \frac{(m_1+0.5-(m_2+0.5)^2)}{1+\sigma}$. Those values associated to each and every distributions are living in the target space $\mathcal{Y}$ in our regression framework.

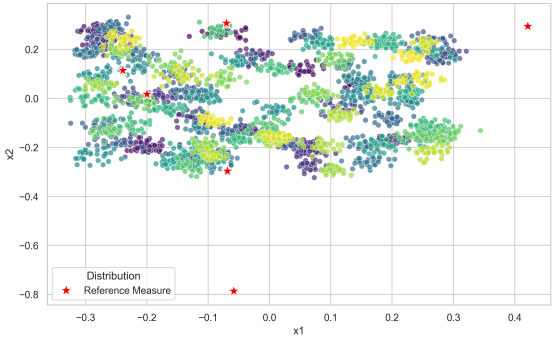
The data is going to be a random sample of 30 points from all of those 100 distributions. Figure 3a shows four of those distributions. All those sample are split into a training set 50% and a test set 50%. We need to specify our reference measure $\mathcal{U}$. For this toy experiment we are going to consider a centered two-dimensional isotropic Gaussian distribution, of variance 0.1. Once again, we are going to work with a sample of this distribution, a sample of size 6. We provide the plot of the reference measure sample in figure 3b. Note that this reference distribution can greatly influence our regression. We will see later how it impacts it.



(a) Samples from 4 different distributions.



(b) Sample points of the reference distribution.



(c) All the training samples and the reference measure sample.

Figure 3: Exploring all the distributions used in the toy experiment.

Next, we are going to perform a discrete-discrete optimal transport between our 100 samples and our reference measure. This is done using the Sinkhorn algorithm developed in section 3.3.2. We obtain the

Sinkhorn potentials of all those problems, that is, we recover for each sample a vector of size 6 (the number of sample in the reference measure). From here, those vectors define our observations and we need to compute a "classical" kernel ridge regression on those data points, using the Radial Basis Function kernel. We have two hyper-parameters to optimize in this kernel ridge regression, that are the parameter of the kernel and the strength of the regularization.

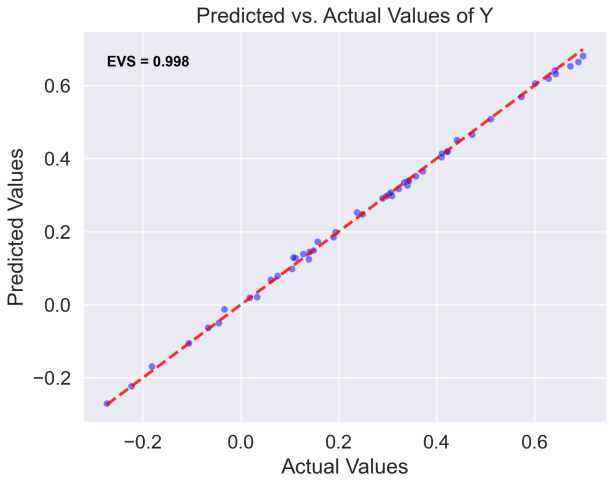


Figure 4: Comparing real objective values against predicted values for the test set.

The model is performing very well on the test set. We obtain an explained variance score of 0.998, figure 4, which is slightly greater than what is obtain with Gaussian Processes in the paper, 0.997. This can be explained by the randomness of our samples, the cross-validation performed to optimize the hyper-parameters of the kernel ridge regression and the fact that we did not optimize the choice of the samples of the reference distribution as in the paper (see algorithm 1 in the paper). Nevertheless, our results are really strong. Taking a step back on this results we have multiple remarks to make. First of all, the reference measure chosen is slightly different from the one in the paper. Ours is certainly better as it is in the center of all the distributions considered, as we can see in figure 3c. The second point we want to make is that the problem is not that hard, the variances of the distribution are very small and thus it is not very hard to differentiate them. Multiple modifications that could be done to make the problem harder are changing the reference measure, changing the parameters of the training distributions and adding dimensions to the problem.

One last thing we want to talk about is the value of the optimal transport optimization parameter ϵ. Indeed, as we stated before, the bigger ϵ, the easier the problem but also the worst the solution is. In opposite, the smaller ϵ is, the harder the problem and the better the solution is. In the figure 5, we show the performance of the model against the value of ϵ for the first experiment case. We see that the best performing models are with the smallest ϵ. We do not encounter specific computing difficulties because we believe our problem is not difficult enough. Therefore, for all experiments we are going to use $\epsilon = 0.01$.

4.3.2 Changing the dimension of the problem

Because our applied case is for 3D models of hardware pieces, we thought trying with at least a 3D model would be interesting. We thus tested for 2, 3, 4 and 8 dimensions problems. All the rest of the experiment's parameter are being kept the same. To take into account the bigger dimensions in the problem we need to change our target variable, we did this as follow:

$$\sum_{i=1}^{\text{dimension}} \frac{(-1)^i (m_i + 0.5)^i}{1 + \sigma} \tag{12}$$

As we can see in figure 6, the performance of the kernel ridge regression diminishes greatly with the

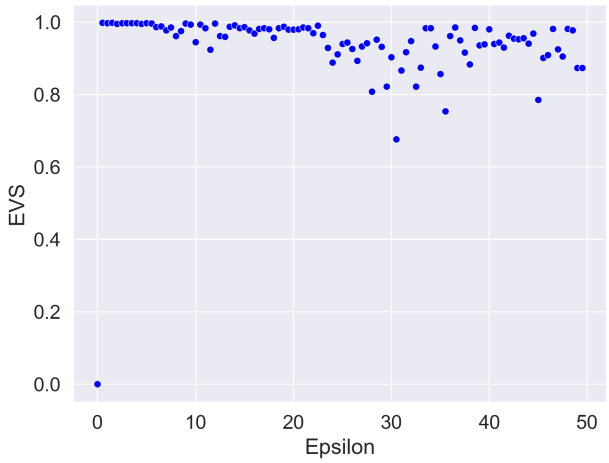


Figure 5: Explained Variance Score with regard to ϵ

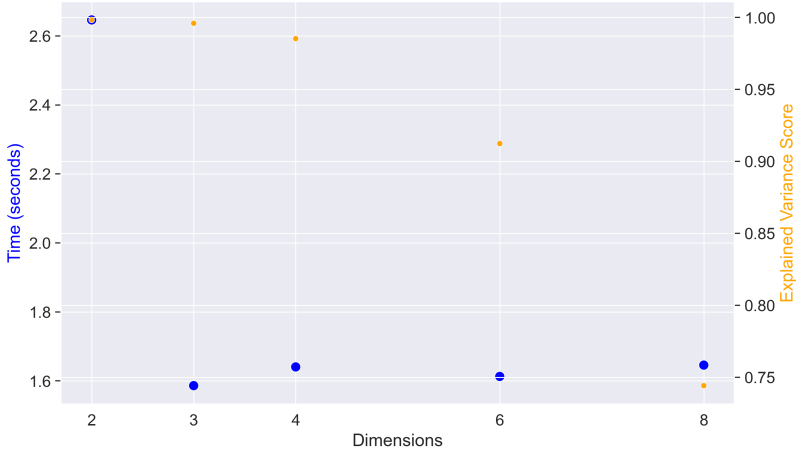


Figure 6: Scatter Plot of Time and Explained Variance Score vs. Dimensions

dimensions. Fortunately, the problem in dimension 3 still achieves a good explained variance score. We provide the performances on the test split for each problem in the appendix. The computing time increases a lot from 2 to 3 dimensions but not so much from there on. Again, our problem is very simple and the computing time might not be representative of the difficulty of the problem. More investigation is needed on this side but we can be quite hopeful for dimension 3.

4.3.3 Changing the reference measure

First, we are going to use a reference measure that is far from our data. Lets consider a 2-dimensional uniform distribution on $[-20, -10] \times [-20, -10]$. In the figure 7 we can see how different is this reference measure against our data. We obtain an expected variance score of 0.997. Which is really good and close

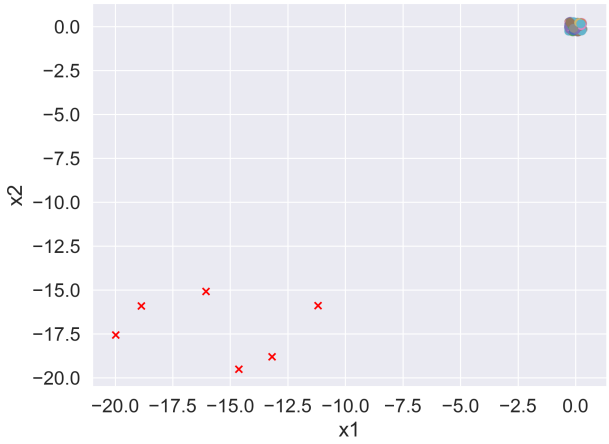
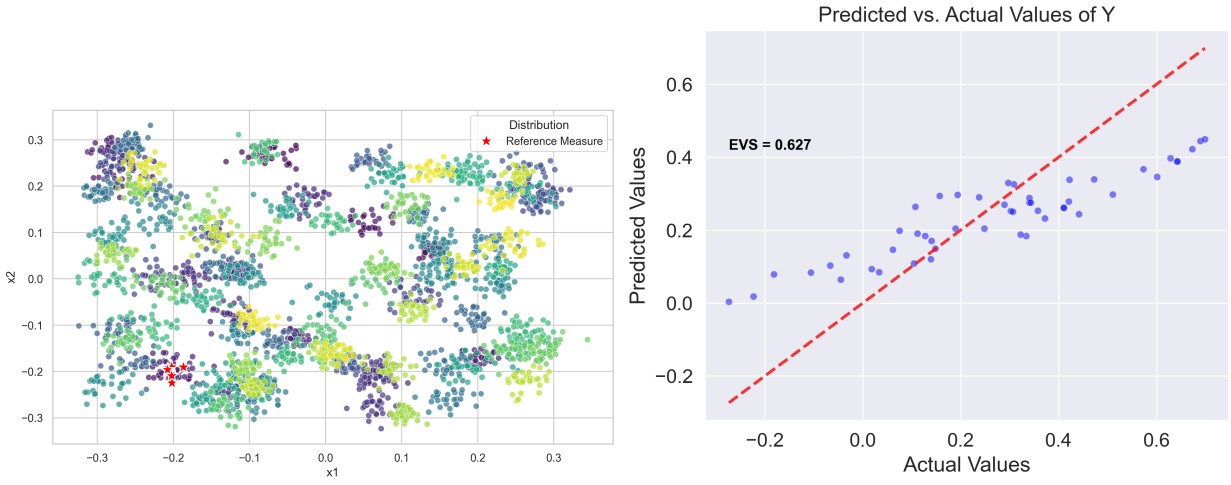


Figure 7: Uniform Reference Measure.

to the score obtained with the previous reference measure. This is surprising to us as we expected the performance to deteriorate. What we can say is that this measure is far from the data but "equally" far, and thus the optimal transport problem is "equally" bad for all the data. Lets try to use a reference measure that is going to favor some distributions.

The next reference measure we are going to use is a normal distribution with mean vector $(-0.2, -0.2)$ and a very small variance of 0.0001. We show figure 8a the reference measure in the middle of our data. This



(a) "Specific" Reference measure.

(b) Test set results with "specific" reference measure.

Figure 8: Tackling the problem with a "specific" reference measure.

time the performance decreases to an expected variance score of 0.627. The optimal transport is here not

equal for all the distributions considered. We show figure 8b the performance of the kernel ridge regression on the test set. This achieve to show that the reference measure is an important "parameter" to optimize in this problem. Our believe is that it should be "equally" distant from the training data, as in it should not favor some of the distributions, like seen in our last example. The paper suggests a way to optimize this reference measure, something we did not achieve to replicate here.

5 Conclusion

Kernel methods are incredibly powerful methods that present lots of advantages against more popular methods such as Neural Networks. They rely on a strong mathematical background and are versatile enough to be used in most cases. Kernel methods are known for some time now. On the other hand, the Optimal Transport theory is quite a recent field and not all possibilities of this theory have been explored. Recently the use of optimal transport between two distributions have been used to compare two distribution together.

The paper "Gaussian Processes on Distributions based on Regularized Optimal Transport" proposes a kernel based on the Regularized Optimal Transport between distributions. The optimal transport theory provides a nice framework to compare distributions and kernel methods allows to capitalise on this knowledge to perform machine learning tasks. We were able to reproduce a small toy experiment achieving similar performances as shown in the paper, even if we did not performed the optimization on the choice of the reference measure. We explored a little beyond this simple experiment and concluded that it may be too simple to showcase the difficulties one may encounter in practice using this method.

Nevertheless, using optimal transport to compare distributions in a kernel method is promising, although it needs more experiment. We are eager to perform more experiment and maybe find an application on real world data to see if the computation scales. It would also be really interesting to succeed into performing Gaussian Processes instead of Kernel Ridge Regression as it provides a nice framework, especially to compute confidence intervals.

Part II
Experimental Part

6 Gaussian Processes

Gaussian Process is a popular model in machine learning, which belongs to the class of Bayesian non-parametric methods. It perform well with a small amount of data, leverage the power of kernel methods through its covariance function and enables to compute confidence intervals for each predictions. It is a very powerful model, and its formulation under some prior assumption is closely related to the Kernel Ridge Regression. We will give a brief presentation of the Gaussian process model in this section. We will used them, section 9, in association with the Sinkhorn kernel and we will observe that it performs better than the Kernel Ridge Regression.

6.1 Definition and training

A Gaussian process is a collection of random variables, which any finite number of them have a joint Gaussian distribution.

Definition 5. *Let $\mathcal{X}$ be a nonempty set. $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel, and $m : \mathcal{X} \to \mathbb{R}$ be any real-valued function. Then, we call a random function $f : \mathcal{X} \to \mathbb{R}$ to be a Gaussian process with mean function m and covariance kernel k, if the following preperty holds:*
For any finite set $X = (x_1, \dots, x_n) \subset \mathcal{X}$ of any size $n \in \mathbb{N}$, the random vector

$$f_X = (f(x_1), \dots, f(x_n))^\top \in \mathbb{R}^n$$

follows the multivariate normal distribution $N(m_X, k_{XX})$ with covariance matrix $k_{XX} = (k(x_i, x_j))_{i,j=1}^n \in \mathbb{R}^{n \times n}$ and mean vector $m_X = (m(x_1), \dots, m(x_n))^\top$. We write

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$$

For notational simplicity we will take the mean function to be zero. Given some observation data $\{\mathbf{X}, \mathbf{y}\}$, one wants to maximise the likelihood that the underlying distribution, from which the data emerged, is a Gaussian process of given parameters. This likelihood is called the posterior and writes:

$$p\left(\mathbf{f}|\mathbf{X}, \mathbf{y}\right) \tag{13}$$

Using the Bayesian framework, one can write the posterior from the likelihood of the data, the prior and the marginal likelihood.

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \qquad p\left(\mathbf{f}|\mathbf{X}, \mathbf{y}\right) = \frac{p\left(\mathbf{y}|\mathbf{X}, \mathbf{f}\right) p\left(\mathbf{f}|\mathbf{X}\right)}{p\left(\mathbf{y}|\mathbf{X}\right)} \tag{14}$$

The prior is the easiest to understand. It represents the Gaussian process observed on all poinst $\mathbf{X}$. According the the definition 5, the prior follows a multi-dimentionnal but finite Gaussian distribution.

$$p\left(\mathbf{f}|\mathbf{X}\right) \sim \mathcal{N}\left(\mathbf{m}(\mathbf{X}), k(\mathbf{X}, \mathbf{X}')\right) \tag{15}$$

Next, the likelihood represents the plausibility that the outputs $\mathbf{y}$ are coming from a Gaussian process $\mathbf{f}$ and inputs $\mathbf{X}$. Acknowledging that we can observe those outputs with some errors, the likelihood follows a normal distribution:

$$p\left(\mathbf{y}|\mathbf{X}, \mathbf{f}\right) \sim \mathcal{N}\left(\mathbf{f}, \sigma_n^2 I\right) \tag{16}$$

Finally, the marginal likelihood is given by:

$$p\left(\mathbf{y}|\mathbf{X}\right) = \int p\left(\mathbf{y}|\mathbf{X}, \mathbf{f}\right) p\left(\mathbf{f}|\mathbf{X}\right) \, d\mathbf{f} \tag{17}$$

The fact that the product of two Gaussian distribution gives another Gaussian distribution (we refer to [3], especially equation A.7 and A.8) allows us to compute the marginal likelihood and more precisely the log-marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K} + \sigma_n^2 \mathbf{I}| - \frac{n}{2} \log(2\pi) \tag{18}$$

Where $\mathbf{K}$ is the kernel matrix or covariance matrix, computed on all the observation's inputs $\mathbf{X}$. And $|\cdot|$ computes the determinant. The first term represents a quadratic form measuring the fit of the model to the data. The second term, penalizes the complexity of the model, it prevents overfitting. The third and last term is simply the normalization constant that appears in the multivariate Gaussian density.

The Gaussian process model is a non-parametric one, it does not assume any form about the regression function. But the model still has parameters, they are called hyperparameters and they appear in the expression of the covariance function, which is also called a kernel. This kernel must be positive definite for the good definition of the Gaussian process. Here is the link to the kernel methods. The covariance function in Gaussian process must have the same propriety as the kernel in kernel methods. Therefore training a Gaussian process boils down to finding the best hyperparameters. This can be done by either cross-validation or optimizing the log-marginal likelihood with respect to the parameters of the kernel (by computing the gradient of the log-marginal likelihood with respect to the parameters of the kernel).

6.2 Inference

We now consider $\mathbf{X}_*$ the test inputs. According the the prior, the joint distribution of the observed of the observed outputs value $\mathbf{y}$ and the function values at the test locations $\mathbf{f}_*$ is given by

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}K(\mathbf{X},\mathbf{X})+\sigma_n^2 I & K(\mathbf{X},\mathbf{X}_*)\\ K(\mathbf{X}_*,\mathbf{X}) & K(\mathbf{X}_*,\mathbf{X}_*)\end{bmatrix}\right) \tag{19}$$

Looking for the conditionnal distribution of the function values at the test inputs we obtain (we refer to [3], equation A.5):

$$\mathbf{f}_*|\mathbf{X},\mathbf{y},\mathbf{X}_* \sim \mathcal{N}\left(\bar{\mathbf{f}_*}, \text{cov}(\mathbf{f}_*)\right) \tag{20}$$

With $\bar{\mathbf{f}_*} = K(X_*,X)\left[K(X,X)+\sigma_n^2 I\right]^{-1}\mathbf{y}$ and $\text{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)\left[K(X,X)+\sigma_n^2 I\right]^{-1}K(X,X_*)$

The point prediction of the Gaussian process at point $\mathbf{x}_*$ is given by the mean of the distribution because we consider the mean squared error loss and our prior is supposed Gaussian. We can write it in a similar fashion to the Kernel Ridge Regression:

$$\bar{f}(\mathbf{x}_*) = \sum_{i=1}^{n} \hat{\alpha}_i k\left(\mathbf{x_i}, \mathbf{x}_*\right) \tag{21}$$

With $\alpha = \left[K(X,X)+\sigma_n^2 I\right]^{-1}\mathbf{y}$.

One key feature of the Gaussian process model, is that the predictions is not a single point but rather the parameters of a normal distribution. This means that the point prediction can be the mean or the median of this distribution. But more importantly, because we are predicting the parameters of a *Gaussian* distribution, we also have access to the variance of this distribution. This enables one to quantify the uncertainty of the point prediction made by the model.

6.3 On the covariance function

The covariance plays a very important role in the performance of the Gaussian process regression model. Choosing the right one is thus fundamental. The paper, *Gaussian Processes on Distributions based on Regularized Optimal Transport* is suggesting a kernel based on Sinkhorn potentials that can be used in the Gaussian process model. More precisely, it suggest a norm between those potentials, and use a wrapper function around this norm. They also provide theoretical guarantees that, one, the Sinkhorn potentials caracterise the distributions, and two, that this norm wrapped in a specific function gives a positive definite kernel. We will review here two different functions that can be used as wrapper around a norm, as long as the norm is defined on a Hilbert space.

Radial Basis Function. The Radial Basis Function (RBF) kernel is one of the most popular because of its simplicity and its good performances.

Definition 6 (Radial Basis Function)**.** *Let $\mathcal{X} \subset \mathcal{H}$, $\mathcal{H}$ a Hilbert space. Let $l > 0$ be the length scale, the RBF kernel $k_l \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$k_l(x, x') = exp\left(-\frac{\|x - x'\|_{\mathcal{H}}^2}{2l^2}\right), \quad \forall x, x' \in \mathcal{X}$$

where $\|.\|_{\mathcal{H}}$ is the norm associated to the inner product of the Hilbert space $\mathcal{H}$.

This kernel has a property unique, which is that it is *universal* ([1], proposition 3.6). In a word this property means that this kernel can, up to a sufficient amount of data, with sufficient compute power and good hyperparameters tuning, can approximate any target function to any desired level of accuracy.

The Matérn family. The Matérn kernel has two parameter, one is the length scale, similar to the RBF kernel and the other one is often called ν and controls the smoothness of the kernel.

Definition 7 (Matérn kernel)**.** *Let $\mathcal{X} \subset \mathcal{H}$, $\mathcal{H}$ a Hilbert space. Let $l > 0$ be the length scale and $\nu > 0$, the Matérn kernel $k_{l,\nu} \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$k_{l,\nu}(x, x') = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{l}\|x - x'\|_{\mathcal{H}}\right)^{\nu} B_\nu \left(\frac{\sqrt{2\nu}}{l}\|x - x'\|_{\mathcal{H}}\right)$$

where $\|.\|_{\mathcal{H}}$ is the norm associated to the inner product of the Hilbert space $\mathcal{H}$ and B_ν is a modified Bessel function.

This is a very famous kernel. Specific values for the ν parameter gives other well known kernels. $\nu = 1/2$ is the absolute exponential kernel and $\nu \to \infty$ is the RBF kernel. Then two values of ν are interpretable and ease the computation of the kernel:

Definition 8 (Matérn 3/2)**.** *For $\nu = 3/2$, the Matérn kernel is made of once differentiable functions is given by:*

$$k_{l,\frac{3}{2}}(x, x') = \left(1 + \frac{\sqrt{3}\|x - x'\|_{\mathcal{H}}}{l}\right) \exp\left(-\frac{\sqrt{3}\|x - x'\|_{\mathcal{H}}}{l}\right)$$

Definition 9 (Matérn 5/2)**.** *For $\nu = 5/2$, the Matérn kernel is made of twice differentiable functions is given by:*

$$k_{l,\frac{5}{2}}(x, x') = \left(1 + \frac{\sqrt{5}\|x - x'\|_{\mathcal{H}}}{l} + \frac{5\|x - x'\|_{\mathcal{H}}^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}\|x - x'\|_{\mathcal{H}}}{l}\right)$$

In the next section we will experiment with the RBF and Matérn 5/2 kernels associated with the Sinkhorn norm proposed by the paper.

7 On the Rotor37 dataset

The Rotor37 dataset is a publicly available dataset published by Safran Group. Physics Learning AI Datamodel (`PLAID`) is a Python package developed by Safran, which aims at providing a nice interface and a new file format for physics informed machine learning tasks.

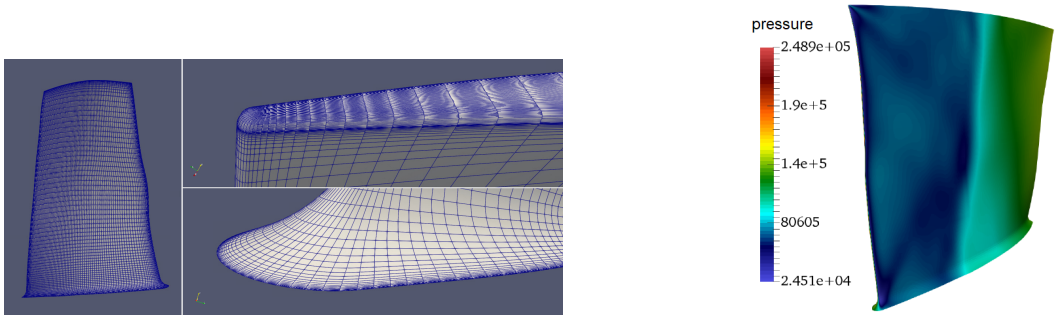
7.1 The dataset

A compressor blade is found an numerous compressors, including gas turbines, jet engines and even refregirators. The blades are designed to efficiently convert kinetic energy onto pressure energy. They have an aerodynamic shape like an airfoil. As the fluid flows throught them, it gains energy and is compressed. The shape of such blades plays an important role into the efficiency of the compressor and its performances. One very important part of the research in blades design is to reduce the fuel consumption in aircraft engines. The Rotor37 dataset contains computational fluid dynamics solutions over different shape of blades. Specifically the Reynolds-Averaged Navier-Stokes (RANS) equations were used over three dimentionnal compressor blades. The goal is to be able to predict the performances and other physics informations from the shape of the blade.

Figure 9: Example of a real compressor blade made by Safran.

This dataset provides a very specific machine learning dataset for a regression task. The dataset is composed of 1000 observations in a train set and 200 observations for a test set. The train set is available in multiple sizes, from 8 to 1000 observations.

The inputs variables: First of all, we have the blade's shape as a mesh. The user disposes of a discrete representation of this mesh, which an example is given in figure 10a. Therefore the blade's shapes



(a) A compressor blade and its mesh superposed.

(b) A crompressor blade and its pressure field.

Figure 10: Example of a mesh and pressure field for a compressor blade.

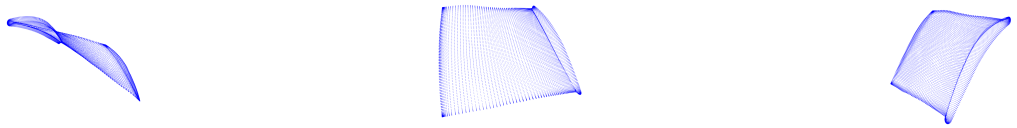
are represented by a cloud point. The problem also has two scalars inputs to complement the mesh. Which

are the pressure P, which refers to the pressure of the incoming fluid, and omega, which represents the velocity of the incoming fluid.

The outputs variables: This problem has vector valued outputs of three different scalars. The efficiency of the blade which is essentially, how much of the input energy (mechanical or electrical) is coneverted, used to accelerate and increase the pressure of the fluid. The massflow which represents how much fluid passes throught per unit of time. The compression ratio measures by how much has the fluid increased its pressure. The problem also provides fields outputs but those will not be covered here.

7.2 Optimal transport

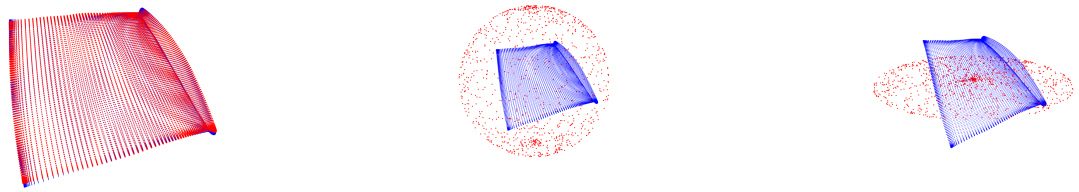
The idea to tackle this regression task is to use the Sinkhorn kernel on the cloud points that represents the shape of the blade and a more classic kernel to take into account the pressure and the velocity of the incoming fluid. The key part is to perform regularized optimal transport between the cloud point of a blade and another cloud point, the reference measure. Here we present what such a cloud point looks like and present regularized optimal transport between a blade and multiple reference measure. Every single blade's



(a) First perspective. (b) Second perspective (c) Third perspective.

Figure 11: Different perspective of the cloud point representing a blade in the dataset.

mesh is made out of 29773 pointsan example is given figure 11. Three ideas emerges quickly when choosing a reference measure to perform regularized optimal transport. The first one, figure 12a, is to consider another



(a) Another blade. (b) The bounding sphere. (c) A disk.

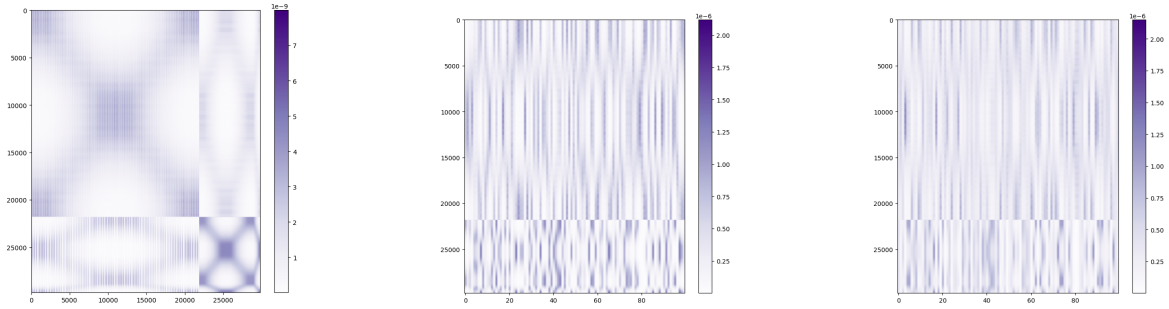
Figure 12: The point cloud of the blade 0 (blue) and different reference measures (red).

blade. This is the easiest to implement, but it is also very costly as both cloud points are of size 29773. Then, one can consider the bounding sphere around the blade as a reference measure. The center and radius are to be chosen (we will see how when discussing our implementation). This is a lot cheaper than to use another blade, as one can choose how many points to sample of the sphere. Finally, the third option that came to our mind, is to consider a disk that cuts the blade in half. We simply used the points sampled on the sphere and set the z-coordinate to zero. We quickly present the results of the Sinkhorn algorithm for regularized optimal transport problem between the blade 0 and each of the three reference measure just discussed. For all the three regularized optimal transport problem presented below, we consider epsilon equal to 1×10^{-3}. As expected, using the blade as a reference measure is a lot more resourceful than using a reference measure with a lot less sample points. We also observe that, even thought the blade is a much denser reference measure, it has a bigger cost than using the sphere or a disk. The sphere has the smallest

Reference Measure (sample)	Entropy regularized cost (E_{RC})	Time to compute (s)
Blade (29773)	-0.01935494	54.7
Sphere (1000)	-0.01531382	2.5
Disk (1000)	-0.01580824	1.8

Table 1: Comparaison of time and entropy regularized cost against the reference measue.

entropy regularized cost of all, but it is hard to draw conclusion from such a little experiment. We can see

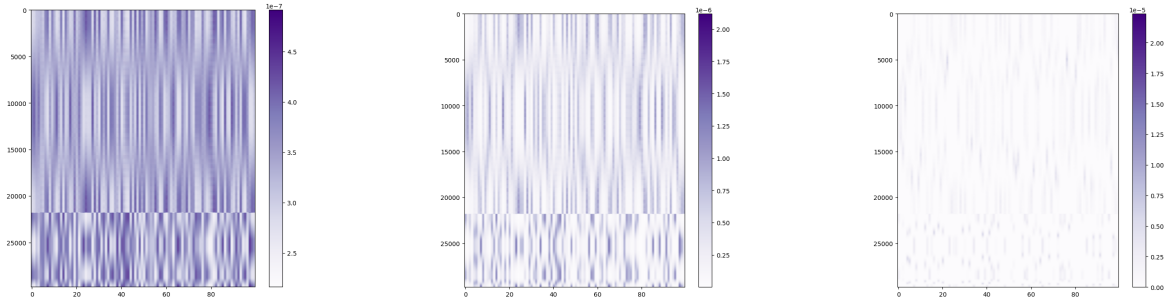


(a) Another blade. (b) The bounding sphere. (c) A disk.

Figure 13: The transport cost matrix for different reference measures. The scale is for the cost.

that the matrices for the disk and the sphere are not easelly interpretable. On the other hand, the transport plan matrix for the optimal transport between two blades, figure 13a, shows a lot of symmetry. This might be due to the fact that the two shapes ressemble a lot each other. Therefore when points are really close, they fully transport their masse to only one point.

On the value of epsilon. The value of ε greatly affects the transport from one distribution to another. Figure 14 are the transport cost matrices for the problem between the blade 0 and the sphere reference measure for different values of ε. We can observe that as expected the matrix unblurs itself as the value of ε diminishes.



(a) $\varepsilon = 0.01$ and $E_{RC} = -0.169$ (b) $\varepsilon = 0.001$ and $E_{RC} = -0.015$ (c) $\varepsilon = 0.0001$ and $E_{RC} = -0.0007$

Figure 14: The transport cost matrix for different values of ε.

We have three significant insights to draw from this. Firstly, the Sinkhorn algorithm takes too much time to compute between two blades. Secondly, it seems to perform worth than using a simple sphere or disk with 1000 sample points. This will arise again when evaluating our regression model. Finally, we observe that the transport cost matrix seems clear enough for $\varepsilon = 0.0001$. Therefore we expect in the regression task to obtain acceptable results starting with $\varepsilon = 0.0001$.

7.3 Subsampling the blades

The time complexity of the Sinkhorn algorithm is $\mathcal{O}(\frac{n|\mathbf{u}|log(n|\mathbf{u}|)}{\varepsilon^2})$ for n the maximum number of points per blade and $|\mathbf{u}|$ the number of points in the reference measure. Our blades have $n = 29773$ points. It is obvious that the algorithm is not going to scale well on this problem. Consequently, wee looked at the possibilities to downscale the problem. One way of doing that is to use a smaller training set, which we explore in the section 9. Another way of doing so is to subsample the points of the blades. We explore here three ways of doing it.

Optimized subsampling. One can choose the points of the blade to keep by by greedily minimizing the maximum mean discrepancy. This should, in theory, select the points that are the most important in the blade. This saves computation for the Sinkhorn algorithm but adds to the total computation because the user must run the subsampling algorithm on all blades. Thankfully this is a greedy algorithm. You only have to run the algorithm once, for all blades, for n points to be able to optimally subsample $m \leq n$ points from the blades. We precomputed the 2000 first optimal indices for both the training set and test set.

Independent random subsampling. One very inefficient way to subsample the blades is, for each blade to select randomly the points to consider for a given number of points n. This is not costly, has choosing n points among 29773 is cheap. One consideration is that it can alter the results of regression task.

One random subsampling. Similar to the previous one, expect that the indices of the subsampling points are the same for all blades. This can be useful if the points of the blades are sorted in some way. This method comes with a small consideration. Should the indices of the points be the same for all training and test observations ? We decided to consider that the indices were a parameter of the optimal transport problem, similarly to the optimized subsampling indices. We thus used the same indices for the train set and test set.

8 A reference model

We present in this section a model, that we will call *reference model*, that do not make use of the optimal transport theory. We will use its results an upper bound of what can be achieved by the Sinkhorn kernel models.

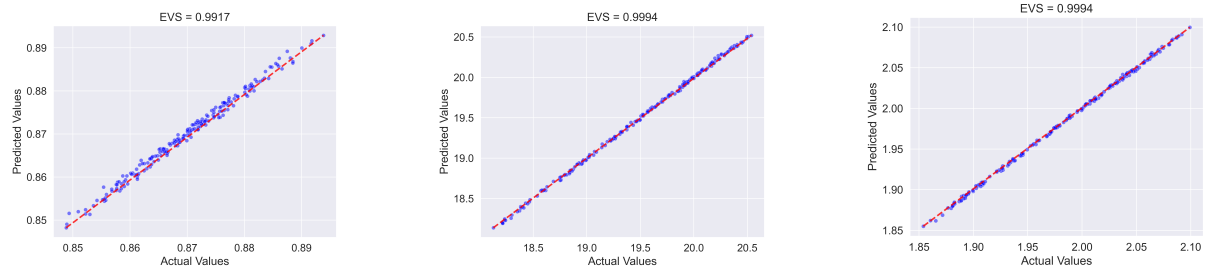
8.1 The full reference model

This model is quite naive, yet achieves amazing perfomances. The idea if very simple, for every blade we will consider one point's coordinate to be one input variable. The blades are represented by 29773 three dimentionnal points, therefore the *reference model* has $3 * 29773 = 89319$ input variables. Equation 22 shows how one blade's array is transformed into a row of the input data of the model.

$$\text{Original array of shape } (29773, 3): \quad \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_{29773} & y_{29773} & z_{29773} \end{bmatrix} \longrightarrow \underset{\text{Transformed array of shape } (1, 89319):}{\begin{bmatrix} x_1 & x_2 & \dots & x_{29773} & y_1 & \dots & z_{29772} & z_{29773} \end{bmatrix}} \tag{22}$$

With 89319 variables, one has te reduce the dimension of the input data. We handle the dimension reduction with a Principal Component Analysis (PCA) of dimensions 32. The choice of the PCA dimension is arbitrary. It is one we found works well, both in term of performance of the *reference model* and in term of computation time, regarding the training of the Gaussian Process.

Thus we consider three Gaussian Process models to train on the output of the PCA to predict the efficiency, the massflow and the compression ratio of the blade.



(a) Efficiency, EVS = 0.9917 (b) Massflow, EVS = 0.9994 (c) Compress Rate, EVS = 0.9994

Figure 15: Predicted vs. true values for all three outputs.

Trained model is a Gaussian Process with anisotropic Matérn 5/2 kernel.

Figure 15 presents the results on the test set for all three scalars outputs. The results are indisputable with an Explained variance score that is never below 0.99 for any of the scalars.

8.2 The subsampled reference model

For all the subsampling method presented in the subsection 7.3, we trained the reference model and evaluated it with the test set.

Optimized subsampling. Because we only precomputed the first 2000 optimal indices for all train and test observations, we will only present the result for the blades subsampled to 2000 points. Figure 17 shows that the results are not great. Therefore we did not investigate smaller size subsample. We only present the prediction of the efficiency here, has we found it is the hardest to predict accurately. The performance of the model drops drastically. 2000 optimally drown points do not seem to be enough to correctly represent the blade.

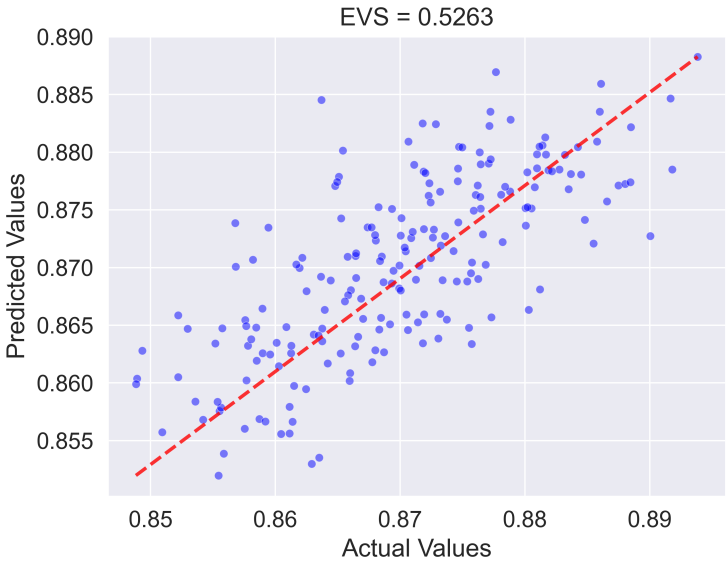


Figure 17: Predicted vs. true values for the efficiency, when optimally subsampling the blade with 2000 points. EVS = 0.5263

Trained model is a Gaussian Process with anisotropic Matérn 5/2 kernel.

Subsampling size	29772	2000
Explained Variance Score	0.0002	0.0001

Table 2: Explained variance score predicting the efficiency for multiple independent random subsampling sizes.

Independent random subsampling. Here we again focus ourselves on predicting the efficiency of the blade and we subsample each and every blade randomly. The results are presented in table 2. The performance of this model is even worse than the optimally subsampled model. Because randomly subsampling a blade is not expensive computationnaly, we wanted to investigate the number of random points at which the performances started to seriously decrease. What we observe is that simply shuffling the points of the blades completely breaks the *reference model*.

One random subsampling. Here the indices of the points are the same for all blades. Those indices are randomly selected upstream and used for the training and test sample. Table 3 showcases the results for different subsampling sizes. Surprisingly, at 2000 points the model performs very well and the performance holds on even down to 50 points per blade. This was quite surprising to us. The explaination of that lies in

Subsampling size	2000	200	50
Explained Variance Score	0.9993	0.9992	0.9988

Table 3: Explained variance score predicting the efficiency for one random subsampling of the indices for multiple sizes.

the construction of the Rotor37 dataset. Each blade's cloud point is constructed from a "reference" blade. Each points of this blade is then slightly modified to investigate different shapes of compressor blade. This means that the blades are not entierly independent. Both the optimized subsampling and the independent and random one broke this dependencie between the blades and explains why those methods performs poorly compared to this subsampling method.

9 The regression task: optimal transport and the Sinkhorn kernel

In this section we present our results of using the Sinkhorn kernel on the Rotor37 dataset. We first expand ourselves into the complications we encountered performing regularized optimal transport on such a large dataset. Then we discuss multiple hyperparameters of the regression task and present their impact on the performance of the regression. Finally we present the model that performed the best in our findings.

9.1 Performing optimal transport

We recall that the regularized optimal transport problem has a time complexity of $\mathcal{O}(\frac{n|\mathbf{u}|log(n|\mathbf{u}|)}{\varepsilon^2})$. Because our dataset is composed of a 1000 train sample and 200 test sample, this meant that to be able to fit the regression Gaussian process we had to perform 1200 times the Sinkhorn algorithm.

Moreover, the Sinkhorn algorithm takes up a lot of GPU memory. We did not found any information on the complexity of the Sinkhorn algorithm memory wise. Therefore this discussion will only be about what be observed. We used the package `ott − jax`, which relies heavely on `JAX`. First of all, we tried a function called `vmap` which is supposed to compute the potentials simultaneously. Unfortunately, the memory usage exploded. Therefore we decided to compute the potentials sequentially. A great way to speed up the computation is to use Just In Time (JIT) compilation. The first computation takes some time, in order to load the cache to the GPU, then the next computations are quick. Figure 18 presents the computation time for the train and test samples for different values of ε. We find that the computation time for ε between

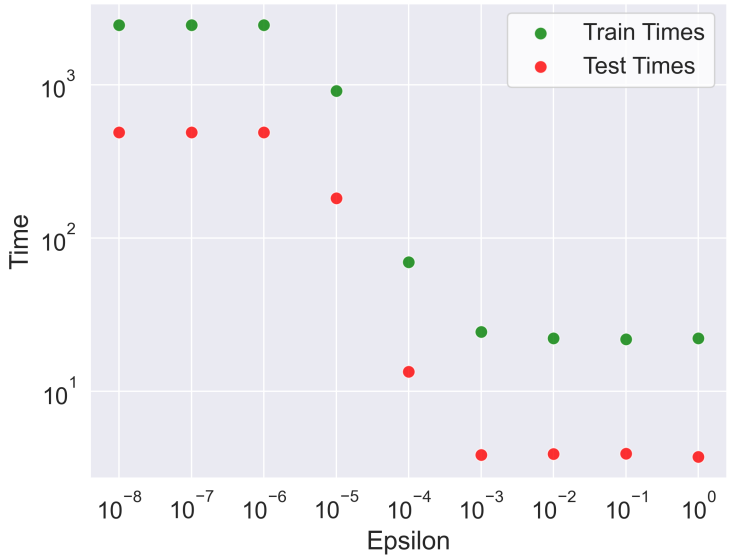
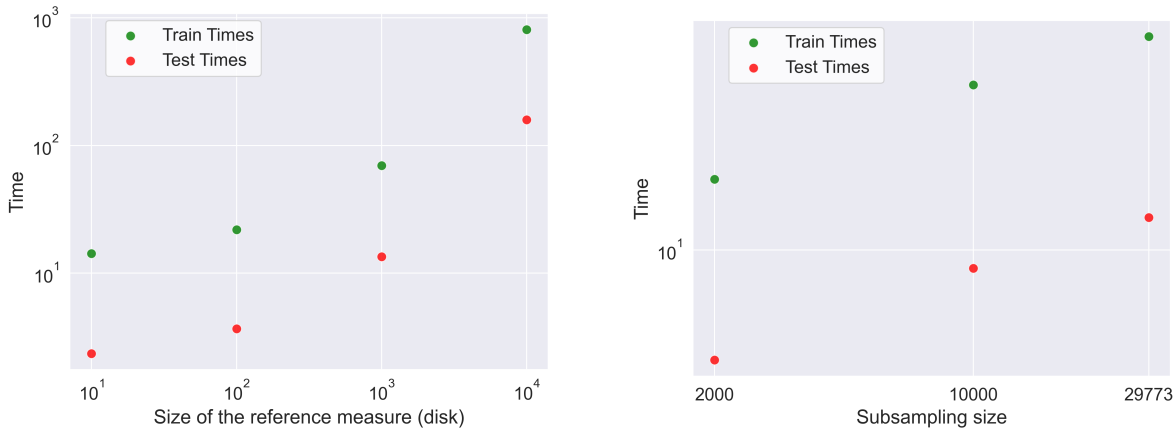


Figure 18: Computation time in seconds with regard to ε.

Logarithmic scale is used for both axis. Computation was made with the split train 1000 and for the reference measure we used the disk with a 1000 points.

1 and 10^{-3} are similar. It goes up between 10^{-4} and 10^{-6} and then is constant for ε up to 10^{-8}. This is in agreement with what we found in figure 14. The optimal transport between a blade and a reference measure seemed acceptable from $\varepsilon = 10^{-4}$. Unfortunately, we found the computation to be too expensive from $\varepsilon = 10^{-4}$ and up. Therefore most of our comparaison for multiple hyperparameters will be conducted with $\varepsilon = 10^{-4}$.

Next, we recoreded the time needed to perform the Sinkhorn algorithm for different reference measure sizes and different subsampling sizes. The results are in figure 19. It is nice to notice that the trends follow the theoretical complexity of the Sinkhorn algorithm. We observe both an augmentation of the computation time with the size of the reference measure and the blade's cloud. Surprisingly, the computation time for the full blades is not as expensive as anticipated. Depending on the hyperparameters choice, performing the Sinkhorn algorithm with the full blades was doable. Regarding the size of the reference measure, the



(a) With different disk reference measure sizes.

(b) With two subsampling sizes and the full blades.

Figure 19: Computation time of the Sinkhorn algorithm.

Logarithmic scale is used for both axis. Computation was made with the train 1000 sample and with $\varepsilon = 10^{-4}$.

computation time increases a lot, to a point where it was not possible to evaluate a lot of hyperparameters with a huge reference measure.

9.2 Hyperparameters tuning

Hyperparameter tuning is really important as it can dramatically change the performance of a model. The fact that we need to perform the Sinkhorn algorithm to create our input variables increases the number of hyperparameters. It is thus essential to have a good knowledge of the behavior of the model according to those hyperparameters. Here we try to understand such behaviors in order to select the parameters that have the best compromise between performance and efficiency. All the regression performances showned in this section are for predicting the efficiency of the blades.

9.2.1 Epsilon

This hyperparameter is essential for the Sinkhorn algorithm. It can quickly make the computation too heavy or if it is not small enough, the potentials will not possess enough information. This choice is thus a compromise between usability of the model and feasability. Figure 21 showcases the performance of the regression model to predict the efficiency of the blades using the Sinkhorn kernel. Curiously we do not observe a steady augmentation in regression performance by diminushing the ε. Clearly the performances are worst for $\varepsilon > 10^{-2}$. However smaller values do not seem to help the regression model to perform. Because of this result, we decided to settle with $\varepsilon = 10^{-4}$ as it seems to perform well and not perticularly worst than smaller values. Moreover, the computation time with this ε was acceptable.

A parameter that could have played a role in not showing greater performances for smaller ε values is the reference measure. Indeed for each different model we drew a 1000 points on the disk to make the reference measure. This could be adressed either by using the *exact* same reference measure or optimize the reference measure. The later is something we attempted but did not succeed.

9.2.2 Subsampling the blades

Subsampling the blades is necessary only if the number of points per blade is too high to be tractable by the Sinkhorn algorithm. Fortunately, we had access to enough computation power to perform the Sinkhorn algorithm with the full blades. We still wanted to investigate the drop in performance if one had to subsample the blades. We present the results of the regression task to predict the efficiency of the blades for the three different subsampling methods we considered. For all the following models, we used $\varepsilon = 10^{-4}$, the disk reference measure with 1000 points and a PCA with 32 dimensions. We used the Matérn 5/2 kernel and trained 24 random starts. We considered only a subsampling of 2000 points for each methods.

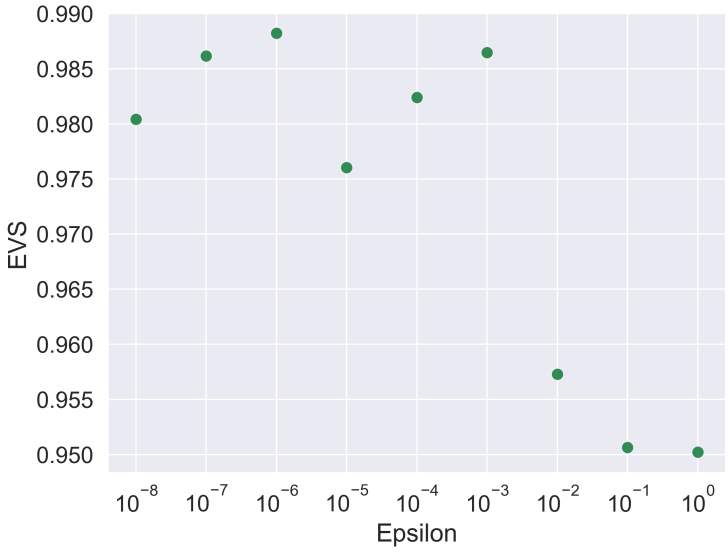


Figure 21: Explained variance score for various ε.

Logarithmic scale is used for the x-axis. Sinkhorn algorithm was performed with the split train 1000 and for the reference measure we used the disk with a 1000 points. The Gaussian process used the Sinkhorn kernel with the Matérn 5/2 wrapper function. To train the model, we used the `GPy` package, performed 24 random start and a 10000 iterations limit.

Optimized subsampling. The reference model achieved a explained variance score of 0.526, which is not very convincing. Using the Sinkhorn kernel we obtained $EVS = 0.5858$. Which is very similar to the reference model. We believe that the Sinkhorn kernel is sensitive to the independance of the blades.

Independent random subsampling. Here the reference model obtained really poor results. It could not predict the efficiency at all. The Sinkhorn kernel did not help this model to perform better. It seems that subsampling each the blades randomly kills every link that two blades could have together.

One random subsampling. For this method the reference model performed incredibly well, achieving explained variance score of 0.99 even with as few as 50 points per blade. Once again, the Sinkhorn kernel performs really closely to the reference model. With a subsampling of 2000 points the model achieves an explained variance score of 0.978.

This means that the Sinkhorn kernel is affected by the way this dataset has been constructed. Therefore the performance of the kernel of this dataset will not reflect the performance one might get on real data. Nevertheless, using the Sinkhorn kernel on this dataset will allow us to invsetigate the behaviour of the kernel under different hyperparameters. But to truly investigate the performance of the Sinkhorn kernel, we should ue it on multiple other datasets.

9.2.3 The reference measure

The reference measure is a key hyperparameter of the Sinkhorn kernel. It is not obvious what the impact of one reference measure or the other might have on the performance of the regression model. Thus choosing the right one is very important. Moreover, we can choose the size of this reference measure, which is a compromise between performance and computation time. In section 7.2 we discussed three different options for the reference measure choice.

Figure 22 presents the explained variance score for those different reference measure and different sizes of those reference measure. The first one is the most straight forward, it uses the very first blade (blade0) of the training sample as the reference measure. This is very computationnaly inefficient because the reference

measure as thus as much points as the observations. The performance did not lived up to the computation time. It is the worst performing reference measure out of all the one we tested.

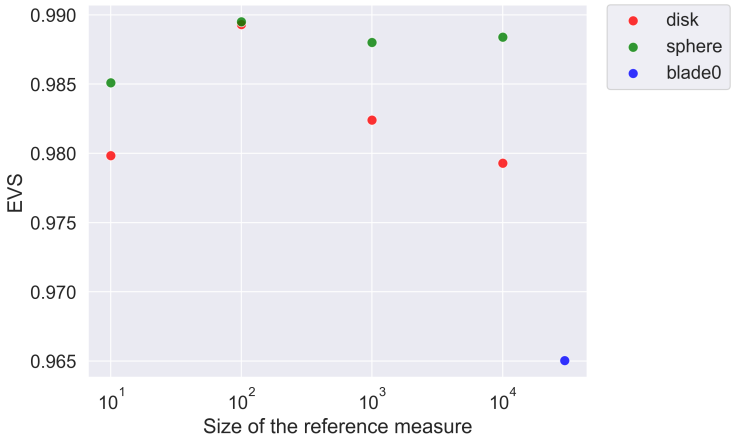


Figure 22: Explained variance score for various reference measures.

Logarithmic scale is used for the x-axis. Sinkhorn algorithm was performed with the split train 1000 and $\varepsilon = 10^{-4}$. The Gaussian process used the Sinkhorn kernel with the Matérn 5/2 wrapper function. To train the model, we used the `GPy` package, performed 24 random start and a 10000 iterations limit.

Then we thought about using the circumscribed sphere of the training blades. The center and radius of this sphere is computed as follows. We compute the mean center of all the training blades, this becomes the center of the sphere. The radius is then simply the maximum distance between this center point and all the points of all the training blades. Therefore different training sample have different center and radius for the sphere reference measure. This reference measure performed really well for all the sizes we tested. Increasing the size of the reference measure improved the regression model up to 100, but above do not seems to enhance the model. This is a great news for computation complexity. It is not necessary to use a huge reference measure to capture all the information of the blades during the Sinkhorn algorithm.

Lastly, we considered the disk reference measure. This disk is essentially computed the same way as the sphere, but the z-coordinate of the sampled points are set to 0. The performances of this reference measure follow a similar path as these of the sphere reference measure, but always a little lower. It is still worse noting that the disk reference measure performed better than the blade0 reference measure.

On the optimization of the reference measure. As we saw, choosing the right reference measure is a key part of the Sinkhorn kernel regression model. The paper, [1], proposes an algorithm to optimize this reference measure as an hyperparameter of the model, see Algorithm 1 from the paper. We tried to optimize the sphere and disk reference measures by adding the coordinates of the points as parameters of the log marginal likelihood of the Gaussian process. Unfortunately we were not able to succeed. Indeed, the gradients (computed with the Auto Diff function of `JAX`) of the log marginal likelihood with respect to the coordinate of those reference measure points were always null. We explored a lot of different optimization algorithm such as `BFGS` and `Adam`, but we never manage to get ride of the vanishing gradients. This is defenitely something to explore to fully unleash the performance of the Sinkhorn kernel.

9.2.4 The PCA dimension

Using the Sinkhorn kernel essentially replaces the blades by a vector of potentials that results from the Sinkhorn kernel in the regression model. Inevitably, this vector is of length the size of the reference measure. As we saw, the reference measures we used are most of the time of size 100 or more. The Gaussian process does not performs well at all with too much variable. Therefore, we decided to reduce the dimensions of those vectors. We did that by using a Principal Component Analysis (PCA). The number of dimensions to keep from the PCA is thus a hyperparameter of the model. It is worth noting that the more dimensions we keep the longer the training of the Gaussian process takes. Figure 23 showcases the performance of the model with different number of PCA dimensions. We clearly observe a plateau from 32 dimensions. This is the number of dimensions we will keep for all of the models we consider.

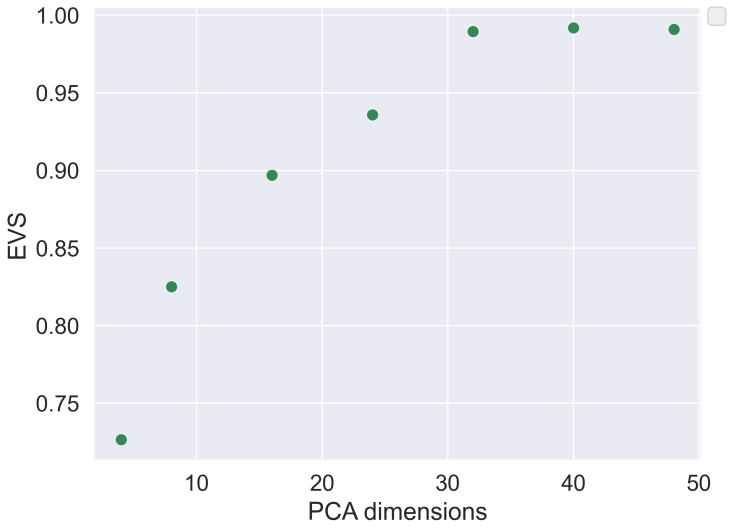


Figure 23: Explained variance score vs. PCA dimensions.

Sinkhorn algorithm was performed with the split train 1000, $\varepsilon = 10^{-4}$ and the sphere reference measure with 100 points. The Gaussian process used the Sinkhorn kernel with the Matérn 5/2 wrapper function. To train the model, we used the `GPy` package, performed 24 random start and a 10000 iterations limit.

9.2.5 Smaller training problems

The Rotor37 provides multiple training sample sizes, that we will call train splits. Those train splits allow for training a machine learning model with a specific set of observations, in order to make the problem harder. This can be usefull to evaluate if a model is still capable when dealing with a small amount of data. We used the train splits from 8 to 1000. We recall that the test split contains 200 observations. As expected the explained variance score goes up as the number of observations goes up. The train split 8 do not performs at all. We observe the biggest leap occurs between tha train splits 32 and 64. The first train split to reach an EVS above 0.9 is the split 125, which shows that the model can perform well without the most number of observations.

9.2.6 The regression method

In this part, we are going to explore the performances of the Kernel Ridge Regression and the Gaussian process models with multiple kernels (wrapper function around the Sinkhorn distance). For this we are going to consider the following Sinkhorn problem: for the train split 1000, we used $\varepsilon = 10^{-4}$, the reference measure is the sphere with 100 points and with used a PCA of dimension 32. To train the Kernel Ridge Regression, we performed cross-validation for the three hyperparameters of the kernel (length scale for the potentials, length scale for the pression, length scale for omega and the penalization coefficient). We used 800 blades to train the model and 200 to validate the cross-validation. The grid we used is presented in table 4.

Penalization parameter	$0.001, 0.01, 0.1, 1, 10, 100$
Length Scale potentials	$0.1, 1, 10, 100, 1000$
Length Scale pression	$0.1, 1, 10, 100$
Length Scale omega	$0.1, 1, 10, 100$

Table 4: The parameters grid we used for cross-validation when training the Kernel Ridge Regression.

Next, in table 5 we present the explained variance score for the Gaussian process and the Kernel Ridge Regression for the three kernels we presented in the section 6. The performances of all models are really good. The best model seems to be the Gaussian process with the Matérn 5/2 kernel.

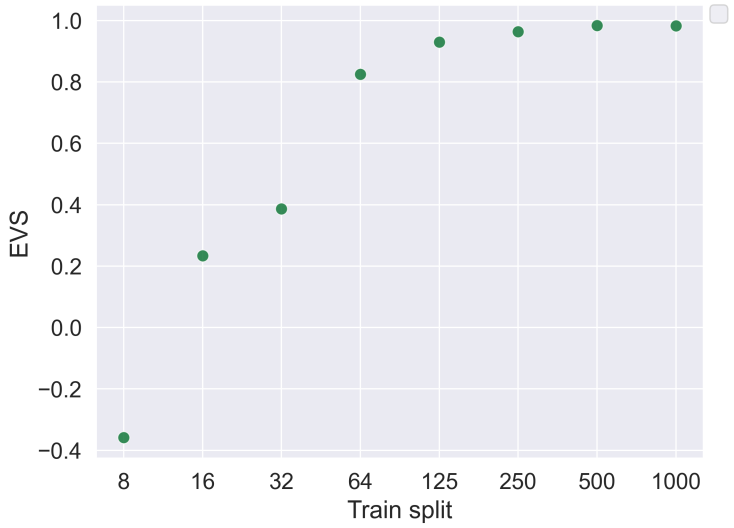


Figure 24: Explained variance score vs. train splits.

Sinkhorn algorithm was performed with $\varepsilon = 10^{-4}$ and the disk reference measure with 1000 points. The Gaussian process used the Sinkhorn kernel with the Matérn 5/2 wrapper function. To train the model, we used the `GPy` package, performed 24 random start and a 10000 iterations limit. For the first 3 train splits, we did not use a PCA for dimensions reduction. For all the others, we used a 32 dimensions PCA.

	RBF	Matérn 3/2	Matérn 5/2
Gaussian process	0.9882	0.9885	0.9895
Kernel Ridge Regression	0.9807	0.9812	0.9834

Table 5: The explained variance score for different models and kernels.

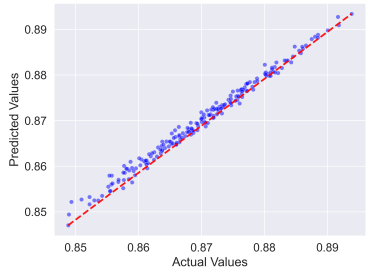
9.3 Analysis of the best model

Now we are going to take a deeper look at the best model (highest EVS) we obtained. We consider the train split 1000 with no subsampling of the blades. The hyperparameters of the Sinkhorn algorithm are the following: $\varepsilon = 10^{-4}$ and we use the sphere reference measure with 100 points. We will use a Gaussian process model with the Matérn 5/2 kernel. The computation of the Sinkhorn potentials took 26 secondes for the training sample and 5 secondes for the test sample.

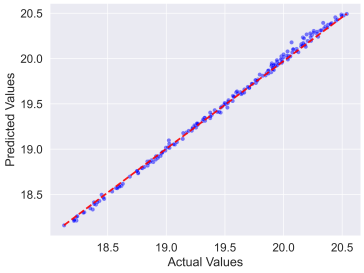
We trained three models, one for each target value, the efficiency, the massflow and the compression ratio. Figure 25 shows the true target values of the test sample against the predictions our model makes of those values. For the efficiency the mean confidence intervals given by the Gaussian process are in th order of 10^{-6}. The massflow as a mean confidence interval of 0.001, but the typical value of a massflow is around 20, whereas the efficiency is between 0 and 1. The compression ratio has a mean confidence interval of order 10^{-5}. The model seems to perform really well for all three outputs variables. Figure 27 exposes the residuals for the three targets and table 6 gives a few statistics about those residuals. We observe that the efficiency and the compression ratio do not reject the the null hypothesis of the Shapiro-Wilk test. However, the model predicting the massflow reject the null hypothesis of the test.

	Efficiency	Massflow	Compression ratio
Mean	3.1^{-5}	1.5^{-3}	1.59^{-4}
Standard deviation	9.7^{-4}	2.6^{-2}	2.5^{-3}
Shapiro-Wilk test statistic	0.989	0.971	0.972
p-value	0.16	0.0004	0.00054

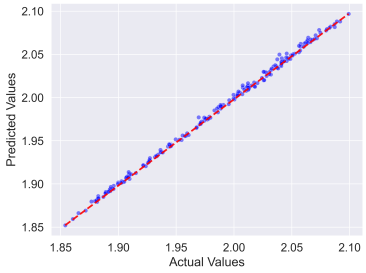
Table 6: Statistics about the residuals of all three models.



(a) Efficiency, EVS $= 0.9895$, $\bar{CI} = 1.7^{-6}$



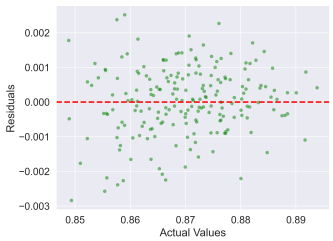
(b) Massflow, EVS $= 0.9983$, $\bar{CI} = 1.1^{-3}$



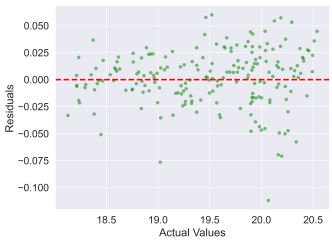
(c) Compression Rate, EVS $= 0.9983$, $\bar{CI} = 1.05^{-5}$

Figure 25: Predicted vs. true values for the "best" model we found.

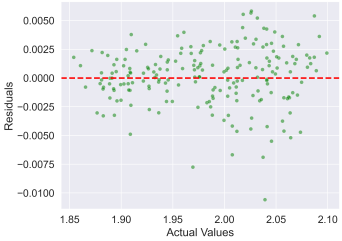
We trained the Gaussian processes with 24 random restarts and a limit iterations of 10000, we also used a PCA of 32 dimensions. $\bar{CI}$ refers to the mean confidence interval over all test sample.



(a) Efficiency



(b) Massflow



(c) Compression Rate

Figure 27: Residuals vs. true values for the "best" model we found.

This Sinkhorn kernel model comes really close to the simple yet powerful reference model presented in section 8. In addition our models provid confidence intervals around the predictions. Moreover, the reference model only works when the blades all have the same number of points, which is not always the case. Because we use a reference measure we could perform the Sinkhorn algorithm between the reference measure and blades that have different number of points. That would not be an issue.

The Sinkhorn algorithm is very time consuming, especially when ε diminushes. Fortunately, a small reference measure is enough to obtain very good performances. This pulls down the time needed to compute the potentials.

10 Conclusion

The paper "Gaussian Processes on Distributions based on Regularized Optimal Transport" proposes powerful kernel for distributions. It is based on Regularized Optimal Transport and especially the Sinkhorn algorithm, leveraing the Sinkhorn potentials, which contains a lot of informations about the two distributions involved. Using a reference measure, those potentials become a way to compare two distributions.

The author shows that by wrapping the Sinkhorn distance between two distibutions, into classical kernels such as the Radial Basis Function and the Matérn kernel, the function is still a positive definite kernel. This allows then, to use this kernel in all kernel methods and Gaussain process. We demonstrated the power of this kernel on a public dataset provided by Safran that aims at recovering the efficency of a compressor blade from its shape. We showed that the Gaussian process model is really powerful and comes really close to a reference model that we presented.

The time complexity is an important bottleneck of the Sinkhorn algorithm. We showed that this can be bypass by using small reference measures. It is not necessary to use complex with numerous points reference measure to achieve great performances. We found the memory complexityto be a bigger obstacle. This kernel only works when powerful (with a lot of memory) are used. Otherwise the computation is not feasable for such a dataset as the Rotor37 one. We also explored the impact of multiple hyperparameter of the regression model. The number of possible combinations for hyperparameters makes it impossible to investigate them all. But by studying them one by one, we were able to precise how one should choose the hyperparameters. One important parameter is the regularization one ε. We noticed that from a threshold value the performances of the model were great, but would not get better has the value decreased. The computation time skyrocketed without an increase consistent increase in regression performance.

There are still a lot of areas to investigate to evalute the performance of the Sinkhorn kernel. The first one is the optimization of the reference measure done in the paper we studied. We encountered vanishing gradient that we could not resolve within the given time. We believe that this could greatly improve the performance of the kernel. The second thing we wanted to investigate was coregionalized kernels. The Rotor37 dataset problems has vector valued outputs and the coregionalized kernel is one way to tackle that in the Gaussian process and kernel methods models. Finally, we wanted to compare the Sinkhorn kernel to other well established kernels for distributions, especially the Sliced-Wasserstein one. It is a very ressourceful kernel to implement and we, unfortunately, did not found the time to do it.

References

[1] François Bachoc et al. *Gaussian Processes on Distributions based on Regularized Optimal Transport*. 2022. arXiv: 2210.06574 [stat.ML].

[2] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press, 2002. URL: http://www.worldcat.org/oclc/48970254.

[3] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006, pp. I–XVIII, 1–248. ISBN: 026218253X.

[4] Motonobu Kanagawa et al. *Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences*. 2018. arXiv: 1807.02582 [stat.ML].

[5] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport*. 2020. arXiv: 1803.00567 [stat.ML].

[6] Joel Franklin and Jens Lorenz. "On the scaling of multidimensional matrices". In: *Linear Algebra and Its Applications, pages 717–735* (1989).

[7] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. "Kernel methods in machine learning". In: *The Annals of Statistics* 36.3 (2008), pp. 1171–1220. DOI: 10.1214/009053607000000677. URL: https://doi.org/10.1214/009053607000000677.

A On the kernel methods

We present in this appendix a few details on kernel methods that we could not expose in our report.

We can represent the positive definite kernel as a matrix or a function interchangeably.

Definition 10. *For a given $k\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ and given inputs $(x_1,\dots x_n)\subset\mathcal{X}$, the $n\times n$ matrix:*

$$K \coloneqq \left(k\left(x_i,x_j\right)\right)_{1\leq i,j\leq n}$$

*is called the **Gram matrix** (or kernel matrix) of k with respect to $x_1,\dots x_n$.*

Propositon 3. *A symmetric function $k\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ is a positive definite kernel* if and only if *the associated Gram matrix K is positive definite for $n\in N$ and $(x_1,\dots x_n)\subset\mathcal{X}$.*

An important propriety of kernel is the following.

Propositon 4. *Let $\mathcal{X}$ be a nonempty set. And let $k_1,k_2,\dots$ be arbitrary positive definite kernels on $\mathcal{X}\times\mathcal{X}$.*

(i) The set of positive definite kernels is a closed convex cone:

(a) $\forall\alpha_1,\alpha_2\geq 0$, $\alpha_1k_1+\alpha_2k_2$ is a positive definite kernel.

(b) If $k\left(x,x'\right)\coloneqq\lim_{n\to\infty}k_n\left(x,x'\right)$ exists for all $(x,x')\in\mathcal{X}^2$, then k is a positive definite kernel.

(ii) The pointwise (Hadamard) product k_1k_2 is definite positive

(iii) For $i=1,2$, let k_i be a positive definite kernel on $\mathcal{X}_i\times\mathcal{X}_i$ ($\mathcal{X}_i$ nonempty), the tensor product $k_1\otimes k_2$ and direct sum $k_1\oplus k_2$ are positive definite kernel on the space $(\mathcal{X}_1\times\mathcal{X}_2)\times(\mathcal{X}_1\times\mathcal{X}_2)$

In Hofmann et al.[7] it is proven that those operations are the only one that preserves the positive definite property.

The next remark shows how to construct the RKHS from a given kernel.

Remark 10. *Let $\mathcal{X}$ a nonempty set and k a positive definite kernel on $\mathcal{X}$. We can construct the RKHS associated to k in the following way.*

We define $\mathcal{H}_0$ as the linear span of k. Meaning every function in $\mathcal{H}_0$ can be expressed as a linear combination of k at points $(x_1,\dots x_n)$:

$$\mathcal{H}_0 \coloneqq \operatorname{span}\{k\left(\cdot,x\right), x\in\mathcal{X}\}$$

Let $f=\sum_{i=1}^n a_ik\left(\cdot,x_i\right)\in\mathcal{H}_0$ and $g=\sum_{i=1}^m b_ik\left(\cdot,x_i\right)\in\mathcal{H}_0$ with $n,m\in\mathbb{N}$, $\left(a_1,\dots a_n\right),\left(b_1,\dots b_m\right)\in\mathbb{R}$ and $\left(x_1,\dots x_n\right),\left(y_1,\dots y_m\right)\in\mathbb{X}$. We equip $\mathcal{H}_0$ with the following inner product:

$$\langle f,g\rangle_{\mathcal{H}_0} \coloneqq \sum_{i=1}^n\sum_{j=1}^m a_ib_jk\left(x_i,y_j\right)$$

Therefore $(\mathcal{H}_0,\langle\cdot,\cdot\rangle)$ is a pre-Hilbert space. Considering the associated norm $\|f\|_{\mathcal{H}_0}^2=\langle f,f\rangle_{\mathcal{H}_0}$, one can construct the RKHS associated to k as the closure of $\mathcal{H}_0$ with respect to the norm above:

$$\mathcal{H}_k \coloneqq \overline{\mathcal{H}_0} = \{f=\sum_{i=1}^\infty c_ik\left(\cdot,x_i\right),\ \text{with}\ (c_1,c_2,\dots)\subset\mathbb{N},(x_1,x_2,\dots)\subset\mathcal{X}\ \text{such that}\ \|f\|_{\mathcal{H}_k}^2<\infty\}$$

Example 5. *We present in detail the **Kernel Ridge Regression**:*

- *The data is : $\mathcal{D}_n=\left(\left(x_i,y_i\right)\right)_{1\leq i\leq n}\in\left(\mathcal{X}\times\mathbb{R}\right)^n$*
- *The kernel here is k. We denote by $\mathcal{H}_k$ the feature space associated to k.*

- *The regularization function is*

$$\begin{aligned}\Omega\colon [0,+\infty[&\longrightarrow \mathbb{R}\\ \|f\|_{\mathcal{H}_k} &\longmapsto \|f\|^2_{\mathcal{H}_k}\end{aligned}$$

  which is indeed a strictly monotonic increasing function.

- *Let l be a loss function. In our regression settings:*

$$\begin{aligned}l\colon (\mathcal{X}\times\mathbb{R})\times\mathcal{H}_k &\longrightarrow \mathbb{R}\\ ((x_i,y_i),f) &\longmapsto (y_i-f(x_i))^2\end{aligned}$$

- *Let $\lambda>0$, the weight of the regularization. This parameter should be tuned by cross-validation.*

The optimization problem thus writes:

$$\min_{f\in\mathcal{H}_k}\sum_{i=1}^{n}(y_i-f(x_i))^2+\lambda\|f\|^2_{\mathcal{H}_k}$$

We know with the Representer Theorem seen in 2 that we dispose of $\alpha_i\in\mathbb{R}, i=1,\dots,n$ such that $f=\sum_{i=1}^n\alpha_i k(\cdot,x_i)$ therefore the norm in the feature space writes:

$$\begin{aligned}\|f\|^2_{\mathcal{H}_k} &= \langle f,f\rangle_{\mathcal{H}_k}\\ &= \langle\sum_{i=1}^{n}\alpha_i k(\cdot,x_i),\sum_{i=1}^{n}\alpha_i k(\cdot,x_i)\rangle_{\mathcal{H}_k} && \textit{(Representer theorem)}\\ &= \sum_{i=1}^{n}\sum_{i=j}^{n}\alpha_i\alpha_j\langle k(\cdot,x_i),k(\cdot,x_j)\rangle_{\mathcal{H}_k} && \textit{(bilinearity of dot product)}\\ &= \sum_{i=1}^{n}\sum_{i=j}^{n}\alpha_i\alpha_j k(x_i,x_j) && \textit{(from remark 3)}\\ &= \alpha^\intercal K\alpha && \textit{(as a matrix formulation)}\end{aligned}$$

Writing the total loss in a matrix form we have:

$$\begin{aligned}\sum_{i=1}^{n}(y_i-f(x_i))^2=\sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{n}\alpha_j k(x_j,x_i)\right)^2 && \textit{(Representer theorem)}\\ =\|Y-K\alpha\|^2 && \textit{(as a matrix formulation)}\end{aligned}$$

Therefore, the optimization problem writes in a matrix formulation:

$$\min_{\alpha\in\mathbb{R}^n}\|Y-K\alpha\|^2+\lambda\alpha^\intercal K\alpha$$

To solve this optimization problem, one must derive this expression with regard to α:

$$\begin{aligned}\frac{\partial}{\partial\alpha}\left[\|Y-K\alpha\|^2+\lambda\alpha^\intercal K\alpha\right] &= \frac{\partial}{\partial\alpha}\left[(Y-K\alpha)^\intercal(Y-K\alpha)+\lambda\alpha^\intercal K\alpha\right]\\ &= \frac{\partial}{\partial\alpha}\left[Y^\intercal Y-Y^\intercal K\alpha-(K\alpha)^\intercal Y+(K\alpha)^\intercal(K\alpha)+\lambda\alpha^\intercal K\alpha\right]\\ &= 0-K^\intercal Y-K^\intercal Y+2K^\intercal K\alpha+\lambda(K+K^\intercal)\alpha\end{aligned}$$

In order to find the minimum one must set the derivative equal to zero:

$$\begin{aligned}\frac{\partial}{\partial\alpha}\left[\|Y-K\alpha\|^2+\lambda\alpha^\intercal K\alpha\right]=0 &\iff -K^\intercal Y-K^\intercal Y+2K^\intercal K\alpha+\lambda(K+K^\intercal)\alpha=0\\ &\iff 2K^\intercal K\alpha+2\lambda K\alpha=2K^\intercal Y\\ &\iff (K+\lambda I_n)\alpha=Y\\ &\iff \hat{\alpha}=(K+\lambda I_n)^{-1}Y\end{aligned}$$

Writing $\hat{\alpha} = (\hat{\alpha}_i, \ldots, \hat{\alpha}_n) \in \mathbb{R}^n$ *we have the Kernel Ridge estimator for all* x *in* $\mathcal{X}$*:*

$$\hat{f} = \sum_{i=1}^{n} \hat{\alpha}_i k\left(\cdot, x_i\right)$$