Universidad Autónoma de Madrid

Escuela Politécnica Superior

22|23

Bachelor thesis

Using of Deep Gaussian Processes for Wind Energy Prediction

Rómulo García Mus

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Bachelor as Computer Engineering

BACHELOR THESIS

Using of Deep Gaussian Processes for Wind Energy Prediction

Author: Rómulo García Mus
Advisor: Daniel Hernández Lobato

marzo 2023

Francisco Tomás y Valiente, n^o 1
Madrid, 28049
Spain

Rómulo García Mus
Using of Deep Gaussian Processes for Wind Energy Prediction

Rómulo García Mus
C\ Francisco Tomás y Valiente Nº 11

PRINTED IN SPAIN

To my family.

Agradecimientos

Me gustaria agradecer a mi Familia el apoyo recibido durante mi paso por la Universidad Autónoma de Madrid.

Además, queria ofrecer mi agradecimiento a José Ramón Dorronsoro por proporcionar los datos utilizados en los experimentos y a Eloy Anguiano Rey por la creacion de la plantilla utilizada para la redacción de este trabajo.

Por ultimo, agradecer a Daniel Hernández Lobato, tutor de este TFG, por su apoyo, profesionalidad y paciencia.

Resumen

La obtención de energía es, a día de hoy, una de las principales preocupaciones en términos de sostenibilidad. Los métodos de obtención de energía más efectivos dependen de recursos limitados y altamente contaminantes. Es por esto, que la producción de energía con fuentes renovables es uno de los focos principales de investigación y desarrollo en gran parte de los países.

En este proyecto nos centraremos en usar Procesos Gausianos y Procesos Gausianos Profundos para la predicción de energía eólica. Nuestros experimentos tomarán lugar en Sotavento, un parque eólico localizado en el norte de España. La energía eólica es altamente variable, las turbinas soportan hasta un máximo de velocidad y los molinos ocupan mucho espacio. Por ello, estas predicciones son cruciales, ya que pueden prevenir desabastecimiento, proporcionar métricas para tomar mejores decisiones a la hora de planear la construcción de parques eólicos, mejorar la eficiencia en la producción, etc. Para dicha predicción se propone el uso de modelos de aprendizaje automático sobre las variables meteorológicas. Al ser escalables a grandes cantidades de datos y no paramétricos, los procesos Gaussianos representan una muy buena opción. Además, su generalización, los procesos Gaussianos profundos, aumenta su flexibilidad y precisión.

Tras la realización de experimentos con diversos modelos, obtenemos buenos resultados utilizando Procesos Gausianos en la predicción de energía eólica actual. En cambio, a la hora de predecir a futuro, obtenemos mejores resultados con Vectores de Soporte de Regresión.

Palabras clave

Sostenibilidad, Renovable, Procesos Gausianos Profundos, Energía, Eólica, Inteligencia Artificial, Análisis de datos

Abstract

Energy is one of the main concerns in terms of sustainability nowadays. The most efficient energy production methods depend on limited resources that tend to be harmful for the environment. Therefore, increasing the renewable energy production is one of the main focuses by most countries.

In this project we will focus on testing Gaussian Processes and Deep Gaussian Processes to predict eolian energy production. Our experiments will take place in Sotavento, a wind mill complex located in the North of Spain. Wind energy is not constant, the wind mills don't operate at certain speeds, the infrastructures require a lot of space... Therefore, these predictions are crucial as they can avoid power shortages, provide the basis for making decisions in power system planning and operation, between many other positive applications. We are gong to use Gaussian processes as they can scale to very large datasets, and are non parametric. Moreover, the deep Gaussian processes approach increases their flexibility and precision.

After proceeding with the experiments using different models, we obtain good results using Gaussian Processes for the prediction of production of current wind energy. Instead, for predicting future wind energy production, we obtain better results with Support Vector Regression.

Keywords

Sustainability, Renewable, Deep Gaussian Processes, Energy, Wind Energy, Machine Learning

Table of Contents

1 Introduction **1**

1.1 Machine Learning 1

1.2 Wind Energy Prediction 3

1.3 Kernel Methods and Bayesian Machine Learning 5

1.3.1 Kernel Methods 5

1.3.2 Bayesian Machine Learning 7

1.4 Summary of chapters 8

2 Kernel Methods **11**

2.1 Gaussian Processes 11

2.1.1 The Gaussian probability distribution 11

2.1.2 An Overview to Gaussian Processes 12

2.1.3 Approximate Scalable Gaussian Processes 16

2.2 Deep Gaussian Processes 17

2.3 Support Vector Regression 19

2.3.1 Support Vector Machines 19

2.3.2 Application to regression 20

3 Experiments **23**

3.1 Prediction of current wind energy production 24

3.1.1 Tools used 24

3.1.2 Results Obtained 25

3.2 Prediction of future wind energy production 28

3.2.1 Adapting the data set 28

3.2.2 Results Obtained 30

4 Conclusions and Future Work **37**

4.1 Conclusions 37

4.2 Future Work 38

Bibliography **39**

Acronyms **41**

1

Introduction

In this chapter we will introduce the main concepts that will be developed in the following chapters to obtain a more general understanding of the basis of the project.

1.1. Machine Learning

Machine learning can be described as the ability for a machine to understand a data set to the point that it can observe patterns and understand the insights of the collected data [1]. The general purpose of this understanding - known as training - is allowing the machine to generate new data taking into account the previous acquired knowledge. For example, this is used to transform hand written digits to digital digits, face recognition, or, as detailed in this project, to predict future wind energy production values. We can understand the machine learning process as a simple mathematical function f(x)=y being x our input vector and y our target value. The training phase is mainly, choosing the f function which will be able to handle the input vectors to produce a target value output. The data used during the training phase is known as training set.

We can observe a basic polynomial curve fitting example: Given an input variable x, we wish to predict the values of a target variable y. The data has been generated using the function $sin(2\pi x)$ with random noise in the target values [1].

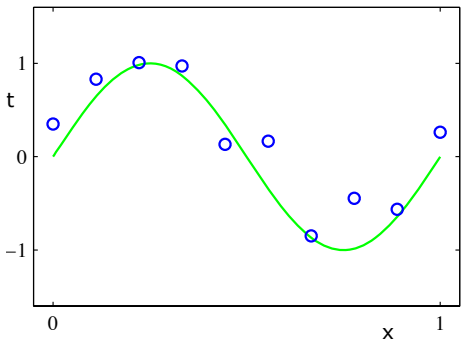


Figure 1.1: Example of polynomial curve fitting problem (Bishop 1.1 [1]).

The green curve shows the function $sin(2\pi x)$, The blue circles are the N = 10 points of observation

of the input variable x along with their corresponding target variable y [1]. Our objective is trying to discover the underlying function $sin(2\pi x)$, but based on a finite data set, it turns out to be a very difficult problem. To predict new data, we need a model explaining the nature of the data set. Therefore we will consider a simple approach based on polynomial curve fitting using a polynomial model of the form:

$$y(x, W) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{1.1}$$

M is the order of the polynomial, The parameters of the model $w_0, \dots w_M$ are collectively denoted by the vector W and need to be estimated by the data.

The values of the vector w will be determined by fitting the polynomial to the training data, this can be done by minimising the error function that measures the error between the function y(x, w) and their corresponding targets. We can use the following widely used error function:

$$E(W) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 \tag{1.2}$$

Moreover we need to choose the order of the polynomial denoted by M.

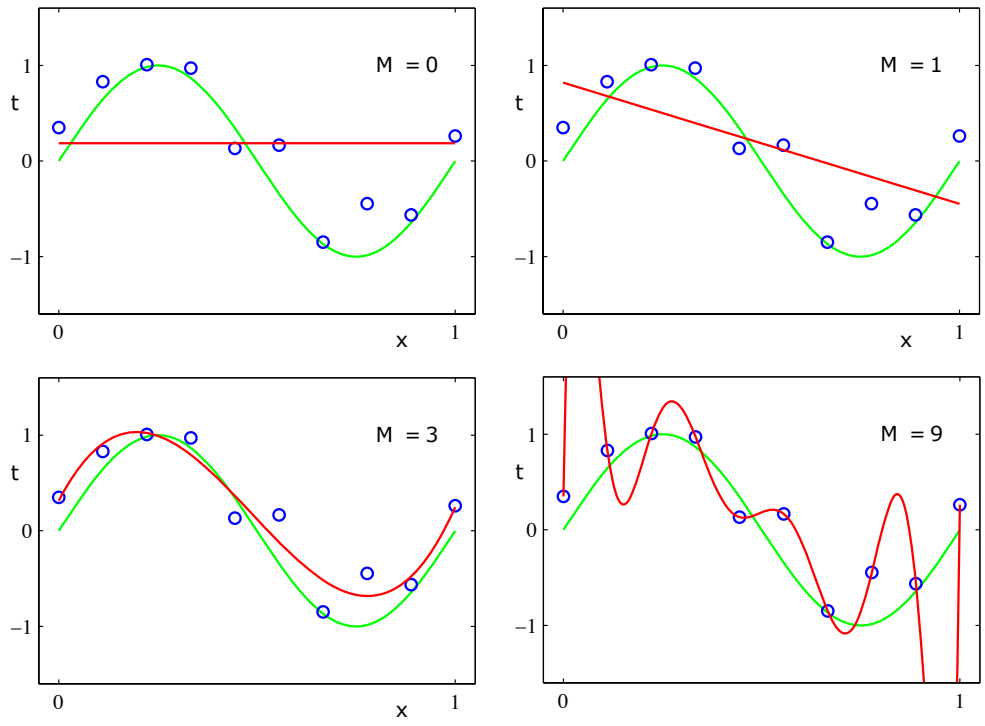


Figure 1.2: Different polynomial orders (Bishop 1.1 [1]).

We can observe that for $M = 0, 1$, the resulting curve -in red- does not approximate well to the data

-in blue-, this is known as under-fitting. For $M = 9$ it goes through all the data points but is very different from the expected curve -in green- $sin(2\pi x)$, this is known as over-fitting [1].

Our objective is to adjust these parameters to have a well trained model for accurate future data predictions. To do so we need to minimize the previously mentioned error function.

Once we have trained the model properly, we should be able to determine new targets for non previously seen input data. These new data are know as the test set. The model being able to predict accurately the values associated to the test set, is known as generalization.

The data set, composed of training and testing data, most of the times, needs some type of pre-processing so that the problem is easier and less computationally demanding. The different parts that compose the data sets are called attributes. This attributes can represent very different things, for instance, speed, force, mass. The variability within each attribute can be very different and can cause the machine learning model to give more importance to some attributes. For this we can scale the data to have the same variability, this is know as standardization. We also have to think if all the attributes are important to obtain the results that we want. If not we can discard them. This are some of the most used preprocessing techniques.

In this project we are going to focus on a supervised learning problem. Supervised learning problems are problems in which the training data conforms values for the input and their corresponding target vectors. A supervised learning algorithm generates an inferred function that we can use to obtain new values from training data.

1.2. Wind Energy Prediction

The use of electricity is increasing exponentially the latest years. Given this scenario, the world is leaning towards renewable energy sources for a more sustainable and long term solution [2].

The European council agreed setting a target of 40 % of renewable energy in the energy mix by 2030 under the 'Fit for 55' package. To reduce the emission under 55 % before 2030, renewable energy will play a very important role as energy is one of the three main pollution production sectors in addition to transport and buildings. One of the counterparts of renewable energy its the big variability on the production depending on the atmospheric conditions and climate. That's the reason why energy forecasting has and will gain more importance during the upcoming years [4].

Wind and Solar energies are the most used renewable energy generation systems.Therefore, they have a remarkable impact on electricity markets. This leads to having a great demand of energy forecasting. Machine learning offers a fast and accurate way of forecasting their production. This approach is based on choosing a model and training it considering the optimal hyper parameters for each energy source to obtain, in consequence, more accurate predictions [5].

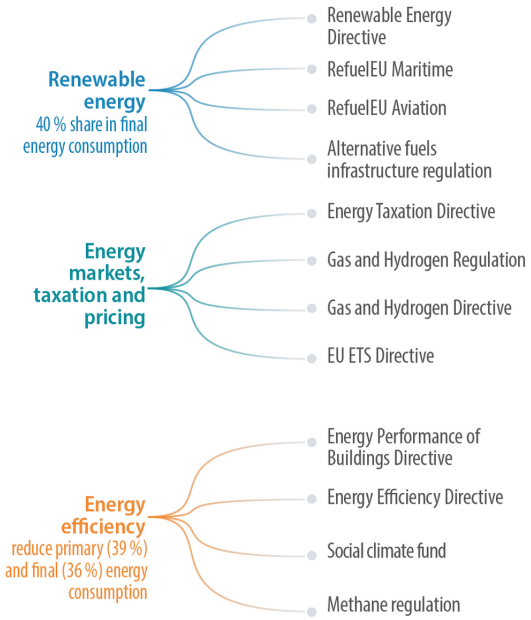


Figure 1.3: Fit for 55 package [3]

We want to make predictions for future wind energy production values in order to optimize the productivity and efficiency of this renewable energy. Our experiment takes place in Galicia, Sotavento [6], a wind park located in the North of Spain. We can obtain real time data and historical data from this park using their public database accessible from their website www.sotaventogalicia.com [6].

Our experiments will be based on the values of 2018. The data is composed of different meteorological variables and the respective wind energy production at that time. We used a geospatial grid consisting of points every 0.125 degrees for latitude and longitude from (44.0, -9.5) to (42.25, -6.0), coordinates situated above the Sotavento Wind Park [6]. From this grid we have values for the meteorological variables with a time difference of 1 hour between them, to a total of 8760 rows, composed of 3479 attributes. Moreover, we have the energy production of the Sotavento wind park for every hour measured in kWh.

The wind energy production data set is formed by diverse attributes such as date and time of the sample, atmospheric pressure, wind components, temperature, etc. Table 1.1 shows some of the meteorological attributes of the wind for a better understanding of the data set [7].

These attributes have a big impact in terms of energy production efficiency. Even if the hour to hour difference is not very significant, the difference between the production at night time and day time is, as it has an impact on the atmospheric boundary layer [8]. Moreover the different directional strengths components and angles can have a great impact as the wind turbines have different angles of operation. The zonal wind component defines the west to east or east to west wind speed, while the meridian's indicates the north-to-south or south-to-north air speed. The power of the wind is the module of the speed of both previously mentioned directional components.

Name	Description
10u_(coordinates)	Zonal wind component at 10 meters of height (m/s)
100u_(coordinates)	Zonal wind component at 100 meters of height (m/s)
10v_(coordinates)	Meridian's wind component at 10 meters of height (m/s)
100v_(coordinates)	Meridian's wind component at 100 meters of height (m/s)
vel10_(coordinates)	Power of the wind at 10 meters of height (m/s)
vel100_(coordinates)	Power of the wind at 100 meters of height (m/s)
2t_(coordinates)	Temperature at 2 meters of height (K)
sp_(coordinates)	Atmospheric pressure (Pa)

Table 1.1: Wind attributes breakdown

In this project we want to use this data to train a model for future wind energy production. We have chosen Deep Gaussian Processes (DGP) as they provide a sophisticated solution for large scale data sets as they use function approximations and can learn the insights of a complex data set using inducing points. We will compare its performance to one of the most used techniques known as Support Vector Regression (SVR) based on Support Vector Machines (SVM) [9].

1.3. Kernel Methods and Bayesian Machine Learning

In this section we are going to describe briefly the kernel methods used and bayesian machine learning.

1.3.1. Kernel Methods

The kernel methods are used to transform scalar product linear models such as $x_1^T * x_2$ to a non linear using kernels, resulting $k(x_1, x_2) = \phi(x_1)^T \phi(x_2)$. They simplify the use of linear models in the extended space $\phi(x)$ without the need of calculating the expansions $\phi(x)$ that could be of infinite dimension.

A kernel can be understood as an inner product in a feature space. In the case of having an algorithm for which the input is a vector in the form of scalar products, we can use the Kernel Trick -also known as Kernel Substitution- to replace the scalar product [1].

Constructing newer kernels can be seen as choosing a transformation to a non-linear feature space $\phi(x)$, defined as:

$$k(x, x') = \phi(x)^T * \phi(x'). \tag{1.3}$$

For example, if we consider the transformation $\phi(x) = x$, we obtain the linear kernel denoted as:

$$k(x, x') = x^T * x'. \tag{1.4}$$

We can extend this definition to cover the polynomial example mentioned in 1.1:

$$k(x, x') = (x^T * x)^2. \tag{1.5}$$

Some kernels consist mainly in the difference of the arguments, named Stationary Kernels

$$k(x, x') = k(x - x'). \tag{1.6}$$

Others such as homogeneous kernels are a specialization of the previous ones and depend on the distance of the arguments named Radial Basis Functions (RBF).

$$k(x, x') = k(||x - x'||). \tag{1.7}$$

Moreover we can create new kernels combining different existing kernels (p.296, chapter 6 Bishop [1]). This technique is useful as we use previous knowledge as building blocks for more complex kernels.

There are a quantitative amount of kernel functions with many interesting possible applications. In this chapter we are going to focus on the Gaussian RBF kernels as are the ones used in SVR and Deep Gaussian Processes.

We start by defining the Gaussian kernel as follows:

$$k(x, x') = exp\left(\frac{-||x - x'||^2}{2l^2}\right). \tag{1.8}$$

Using the previous mentioned method in extent we can deduce [10] the Gaussian RBF kernel:

$$\begin{aligned} k_{RBF}(x, x') &= exp\left(-\frac{||x - x'||^2}{2l^2}\right) = exp\left(-\frac{< x - x', x - x' >}{2l^2}\right) \\ &= exp\left(-\frac{< x - x >}{2l^2}\right) exp\left(\frac{< x - x' >}{l^2}\right) exp\left(-\frac{< x' - x' >}{2l^2}\right) \end{aligned} \tag{1.9}$$

Using $k1(x, x') =< x, x' >= x^T x'$ and $f(x) = exp(-\frac{<x,x>}{2l^2})$ and l being the length-scale denoting the smoothness of the inferred function, we can denote the Gaussian RBF kernel as follows:

$$k_{RBF}(x, x') = f(x)f(x')exp\left(\frac{k_1(x, x')}{l^2}\right). \tag{1.10}$$

1.3.2. Bayesian Machine Learning

From a Bayesian point of view, probabilities provide an amount of uncertainty instead of frequencies of random events. The Bayes theorem has the following form:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}, \tag{1.11}$$

with w the parameters of the model and $p(w|D)$ representing the posterior probability once D has occurred.

In the other side of the equation, $p(D|w)$ is named the likelihood and describes how likely is the observed data for the vector w. And $p(w)$ is the prior probability representing the initial uncertainty.

With this definition, the Bayes' theorem can be described as:

$$posterior \propto likelihood * prior \tag{1.12}$$

Form a frequentest point of view, we consider the parameters w fixed and we determine their values by an estimator.

One of the most used estimators is the maximum likelihood. It consists in choosing a the values for w to maximize the likelihood. This is equivalent to minimize the previously mentioned squared error.

As the maximum likelihood estimator favors more complex models that explain better the training data, it tends to learn also the added noise. Therefore, deciding the model complexity for the particular problem can lead to excessively complex models and over fitting. We can determine model complexity using independent hold-out data, but this can be both computationally expensive and wasteful of valuable data.

Bayesian linear regression avoids the over-fitting problem of maximum likelihood, and enables us to determine the model complexity using only the training data automatically.

We are interested in making predictions of t for new values of x. To do so, we use the predictive distribution over the parameters w defined by:

$$p(t|t, \alpha, \beta) = \int p(t|w, \beta)p(w|t, \alpha, \beta)dw, \tag{1.13}$$

where t is the vector of target values from the training set and β and α precision parameters.

We use different sized data sets to fit into the model using a combination of Gaussian basis functions. The data points have been generated using the function $sin(2\pi x)$ (green curve) with added Gaussian noise. Focusing on the red curve representing the mean of the corresponding Gaussian predictive distribution and the red shaded region being the standard deviation at both sides of the mean,

we can observe that the level of uncertainty decreases as more data points are observed.

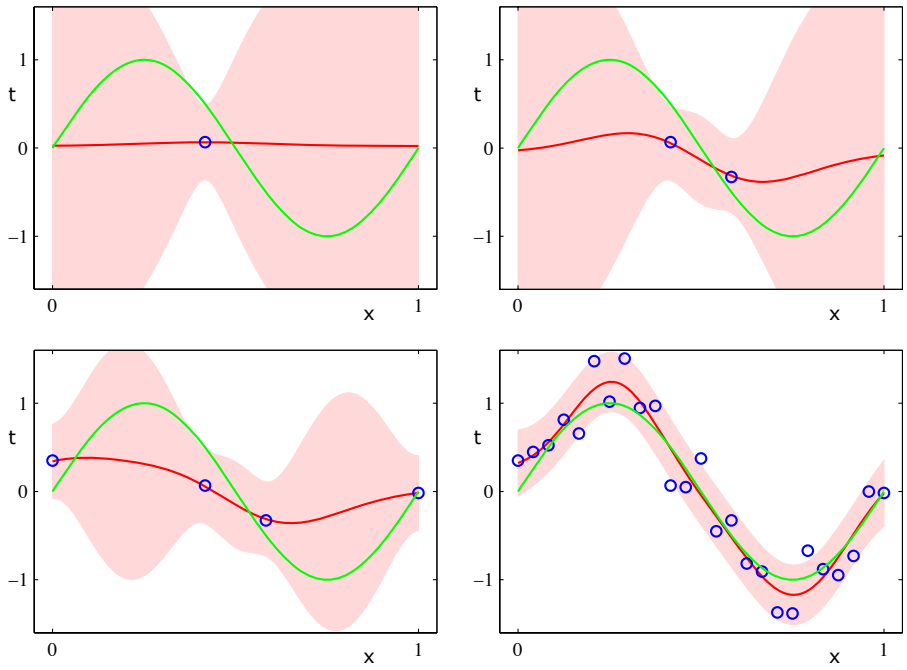


Figure 1.4: Examples of the predictive distribution for a model consisting of 9 Gaussian basis functions (Bishop 3.3) [1]

Using basis functions to model the underlying function leads to high noise contribution in regions away from the function centres. The problem with this approach is that it is easy for the model to become very confident in its predictions, even when the new input data is far from the basis functions centres.

Sometimes it is easier to inference in the function space rather than in the parametric space w. Gaussian processes are an alternative Bayesian approach to regression following this principle. In future sections we are going to discuss their advantages and applications.

1.4. Summary of chapters

Chapter 2: Kernel Methods

In this Section we are going to describe the different kernel methods used afterwards in the experiments subsection consisting of Gaussian Processes, Deep Gaussian Processes and Support Vector Regression. A Gaussian process is a stochastic process, where all the collection of random variables describe a multivariate normal distribution with a distribution over functions with a continuous domain. A deep Gaussian process is a hierarchical stack of Gaussian Processes to gain flexibility. Support Vector regression is the equivalent to support vector machines applied to regression problems that may manage better high noise conditions.

Chapter 3: Experiments

In this section we are going to discuss the experiments using the data corresponding of the Sotavento wind energy park. we are going to distinguish two subsections. The first subsection describes how Gaussian Processes, Deep Gaussian processes and Support Vector Regression, are configured, trained and evaluated over the data for the current energy production values. In the second subsection, we are going to use what we learned and try to predict future eolian energy production values based on the previous ones adding also tests using deep Gaussian processes with non Gaussian noise.

Chapter 4: Conclusions and Future work

In this section we will note the different conclusion obtained from the experiments section and future work discussion.

2

Kernel Methods

In this Section we are going to describe the different kernel methods used afterwards in the experiments subsection consisting of Gaussian Processes, Deep Gaussian Processes and Support Vector Regression.

2.1. Gaussian Processes

In this subsection we are going to focus on explaining the basis of Gaussian Processes. First of all, we are going to describe the Gaussian probability distribution and afterwards we are going to give an overview to Gaussian processes.

2.1.1. The Gaussian probability distribution

The Gaussian probability distribution describes a model used in the domain of continuous variables [11]. It is extensively used and better known as the normal distribution. We can describe it as follows for a single variable:

$$N(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\}, \tag{2.1}$$

where μ is the mean and σ^2 is the variance. If we now consider a D-dimensions vector named x, we obtain the following equation [12]:

$$N(x|\mu,\Sigma) = \frac{1}{(2\pi^{D/2})} \frac{1}{|\Sigma|^{1/2}} exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}, \tag{2.2}$$

where μ is the mean vector, Σ is the covariance matrix of size D^2.

The Gaussian distribution is a very flexible and adaptive solution. It is used in many different scenarios and contexts. In a single real variable, the Gaussian distribution maximizes the entropy. The same happens in to a multivariate Gaussian. Moreover, if we consider the sum of multiple random variables -Giving another random variable-, given the central limit theorem (Laplace) the distribution tends to a

Gaussian distribution as the sum terms increase [11].

If the data set has independent feature-probabilities, finite variance and can grow in size it turns into a Gaussian distribution. The Gaussian distribution has a very big importance as it fits many natural phenomena like height, age, test scores, etc.

2.1.2. An Overview to Gaussian Processes

Using Gaussian processes has multiple benefits. Its complexity grows with the number of observed points to better explain the observed data. It has robustness to over fitting with tolerable error indexes and can model a very rich class taking into account a small number of hyper parameters.

A Gaussian process is a generalization of the Gaussian probability distribution. The use of Gaussian processes for both supervised and unsupervised learning problems was introduced by Rasmussen and Williams in 2005 [11]. They represent an stochastic process such that for any finite set of points $x_1, ..., x_m$, $f(x_1), ..., f(x_m)$ follows a multivariate Gaussian distribution.

A stochastic process defines a group of observed random variables.

Gaussian processes can be understood as a distribution over functions. And therefore we can sample functions and use the influence of the data points observed. This sampled functions constitute the posterior distribution that will be used to determine the predictive distribution.

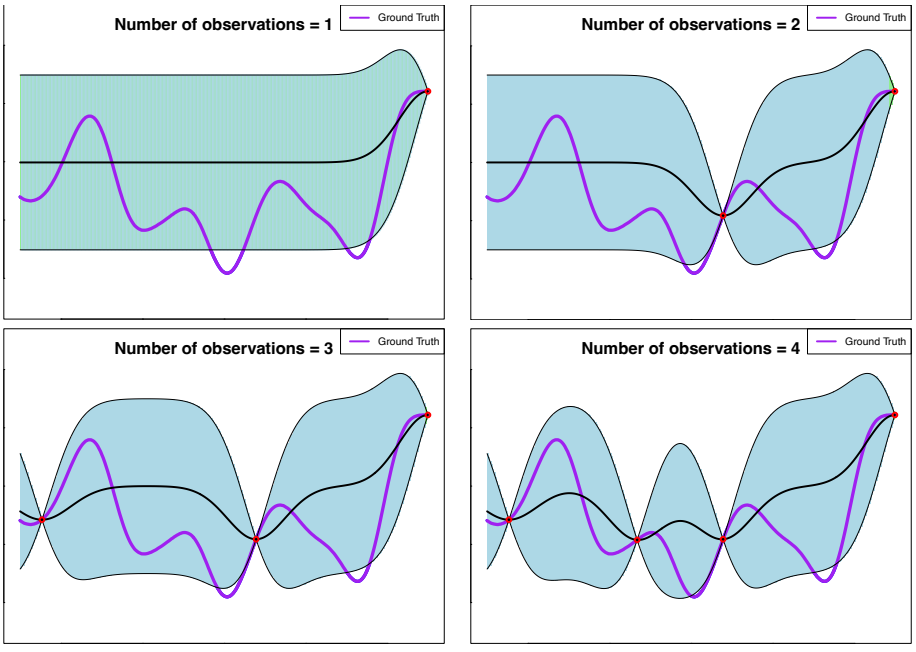


Figure 2.1: We can observe that as the number of observations increase, the mean (black) gets similar to the ground truth (purple) in addition of reducing the uncertainty (blue) [13].

As we observe, increasing the number of data points, lead to better results to predict the objective

function. Our objective is to obtain a predictive distribution from the model that is the closest possible to the objective function, as it is cheaper than evaluating the objective function directly.

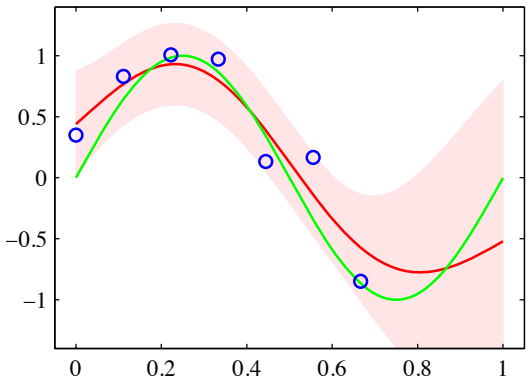


Figure 2.2: Illustration of Gaussian process regression applied to the sinusoidal data set in (Bishop 6.4 [1]). The blue dots represent the data points. The green curve represents the function from where the points are obtained adding Gaussian noise. The red curve represents the mean of the GPs predictive distribution and the red shaded region represents the standard deviation.

To obtain the predictive distribution we need to use the marginalization property of distributions where:

$$\begin{aligned} p(y_1) &= \int p(y_1, y_2) dy_2, \\ p(y_1, y_2) &= \mathcal{N}\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \right), \\ p(y_1) &= \mathcal{N}(y_1 | a, A), \end{aligned} \tag{2.3}$$

to be able to work with a finite set of random variables.

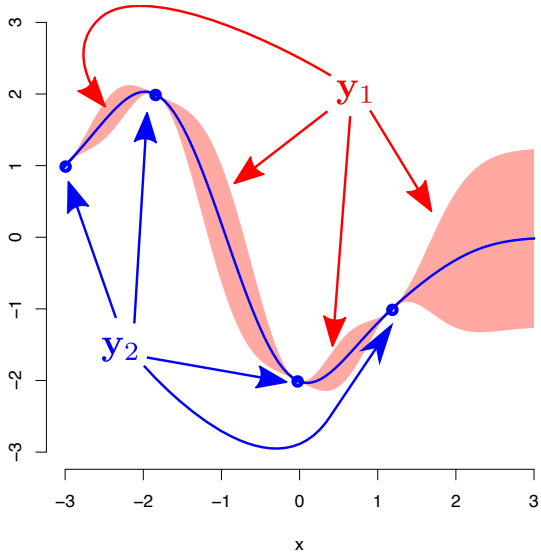


Figure 2.3: Obtention of the predictive distribution as seen in [14].

Now we compute the predictive distribution as follows:

$$\begin{aligned} &p(y_1|y_2) = p(y_1, y_2)/p(y_2), \\ &p(y_1|y_2) = \mathcal{N}(y_1|a + CB^{-1}(y_2 - b), A - CB^{-1}C^T), \end{aligned} \tag{2.4}$$

with a predictive linear mean in $y2$.

Now we have to consider additive noise defined as σ_y^2. Resulting in the following predictive distribution:

$$p(y_1|y_2) = \mathcal{N}(y_1|a + C(B + I\sigma_y^2)^{-1}(y_2 - b), A - C(B + I\sigma_y^2)^{-1}C^T). \tag{2.5}$$

A Gaussian process is defined by its mean $m(x)$ and covariance function $Cov(x, x')$ [15]. The expected value for the random functions from the Gaussian process domain for x is it's prior mean, defined by:

$$m(x) = E[f(x)]. \tag{2.6}$$

As the prior mean determines the global tendency of the latent function before observing the data, we often set it to 0.

The dependencies between two points from a random function contained in the Gaussian domain are reflected in the covariance matrix given by Cov(x,x'). The covariance function has the following shape:

$$Cov(x, x') = E[(f(x) - m(x))(f(x') - m(x'))]. \tag{2.7}$$

The covariance function $Cov(x, x')$ acts as a kernel and it has to provide positive and symmetric matrices. That is why we can see this function also mentioned as $k(x, x')$.

We also need a covariance function to measure how the variables are related to each other and the impacts between them. The covariance function has some hyper-parameters that are defined by the chosen kernel. In this case we are going to use the RBF kernel so our covariance function would look as follows:

$$Cov(x, x') = \sigma_f^2 exp(-\frac{1}{2l^2}||x - x'||^2) + \sigma_n^2 \delta_{xx'}, \tag{2.8}$$

where σ_n^2 represents the additive Gaussian noise and $\delta_{xx'} = 1$ if $x == x'$, otherwise $\delta_{xx'} = 0$.

The previously named hyper-parameters are σ_f^2, σ_n^2 and l. We can optimize them using different methods such as cross validation Leaving-One-Out (LOO) or Bayesian selection.

We can observe a big impact on changing the length-scale in Figure 2.4.

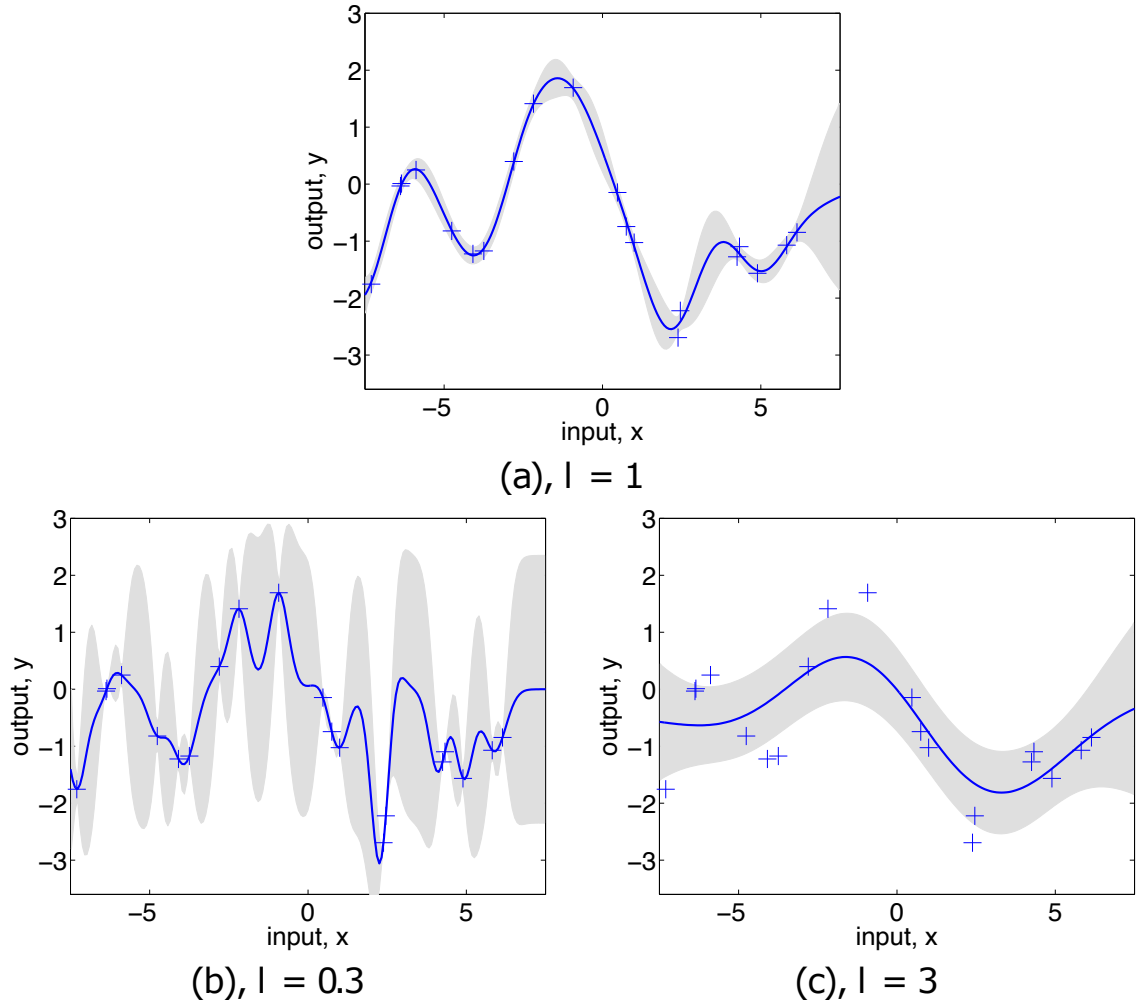


Figure 2.4: The + symbols represent the data, the grey shadow is the 95 % confidence region and the blue line is the underlying function. (Rasmussen 2.3) [11]

We will set the hyper-parameters by optimizing the marginal likelihood distribution using Bayesian selection. Maximizing the marginal likelihood is robust as it tends to favor the right model [1].

The marginal likelihood is how likely is it for data to happen given a model. Given the following marginal likelihood expression (Rasmussen 5.4.1) [11]:

$$logp(y|X,0) = -\frac{1}{2}y^T K_y^{-1} y - \frac{1}{2}log|K_y| - \frac{n}{2}log2\pi \tag{2.9}$$

We can divide it in three parts for a better understanding.

- The first part $-\frac{1}{2}y^T K_y^{-1} y$ is in charge of adjusting the function to the data set.
- The second part $-\frac{1}{2}log|K_y|$ is in charge of penalize the complexity of the model.
- The third part $-\frac{n}{2}log2\pi$ is a normalization constant.

Therefore, if the first term is near 0 it means that the model adjusts well to the data set and the second term will become smaller as the complexity of the data set grows.

The computational cost of Gaussian Process would be of the order of $O(N^3)$. To be able to handle large amounts of data we are going to try to go back to a parametric model but with the possibility to make inference easily.

2.1.3. Approximate Scalable Gaussian Processes

We are going to introduce approximations to our Gaussian Processes to reduce the computational cost using inducing points. This method is known as Sparse Gaussian Processes [15].

There are two main approaches, Full Independent Training Conditional (FITC) and Variational Free Energy (VFE).

- FITC consists in optimizing the marginal likelihood of an approximate GP model, it is easier to optimize but with less accuracy.
- VFE maximizes the fidelity to the exact GP giving a more accurate representation but being more difficult to optimize.

We are going to focus on VFE as it is the option used in the experiments section.

Let's consider the following approximate distribution:

$$q(f,u) = p(f|u)q(u) = p(f|u)\mathcal{N}(u|m,S), \tag{2.10}$$

with f representing a vector with the values of the process in the training data and u a vector with the values of the process in the inducing points M, having always $M\acute{}N$, N being the size of the training data set. We have to tune $\mathcal{N}(u|m,S)$ with the inducing points being parameters of the approximate distribution q.

Now, we use $q(f, u)$ to compute a lower bound on the log marginal likelihood, having:

$$\mathcal{L}(q, \theta) = E_{q(f)}[logp(y|f, \theta)] - KL[q(u)|p(u)], \tag{2.11}$$

with $E_{q(f)}[logp(y|f, \theta)]$ being the mean squared prediction error and $KL[q(u)|p(u)]$ the Kullback-Leibler divergence between Gaussians, measuring how different is the approximate probability distribution to the prior. Being $logp(y|f, \theta)$ the likelihood of the model.

By maximizing this lower bound, we make $q(f, u)$ look close to p(f,u|y) in terms of the KL. The predictions are made marginalizing u in $p(f^*|u)q(u)$. The cost gets reduced to $O(M^2N)$.

We can reduce this cost even further if we introduce the use of minibatch training as follows:

$$\mathcal{L}(q, \theta) = \frac{B}{N} \sum_{i \in B} E_{q(f_i)}[logp(y_i|f_i, \theta)] - KL[q(u)|p(u)], \tag{2.12}$$

with B the batch size and N the total sample size.

Following this approach we can reduce the training cost to $O(M^3)$. Resulting in a Gaussian Processes approach that can scale to large datasets [14].

2.2. Deep Gaussian Processes

Gaussian processes models of a single layer have a limitation by design in terms of expressiveness of the covariance function. Big parameterized space of kernels create a difficulty when proceeding with inference as is very costly in terms of time and can tend to over fit. The combination of kernels through products and sums for thew obtention of compositional kernels is expensive and limited to base kernels.

Deep Learning outperforms other techniques if the data size is large [16]. In this Project we want to use the values extracted in the Sotavento eolian park during the year 2018. This represents a big data source as it is divided in over 3000 attributes mainly consisting of the speed of the wind divided in direction components divided into small angle portions, time, pressure and energy produced. We have over 8000 rows of information consisting of our sample.

Deep Gaussian processes (DGPs) are multi-layer hierarchical generalizations of Gaussian processes (GPs) and were introduced by Damianou and Lawrence in 2013 [17]. In contrast to models with kernels composed of numerous hyper parameters, DGPs learn a representation hierarchy non-parametrically with very few hyper parameters and have better calibrated uncertainty than other deep models [17].

If we consider a DGP consisting of three hidden layers, with three nodes per layer being all of them a GP. The inputs attributes defined as x_i, enter the GPs and obtain a predictive distribution as output for each GP. These distributions act as input for the next layer, and so on. This behavior is named GP

mapping. The last layer outputs the distributions used to compute the observed variables y.

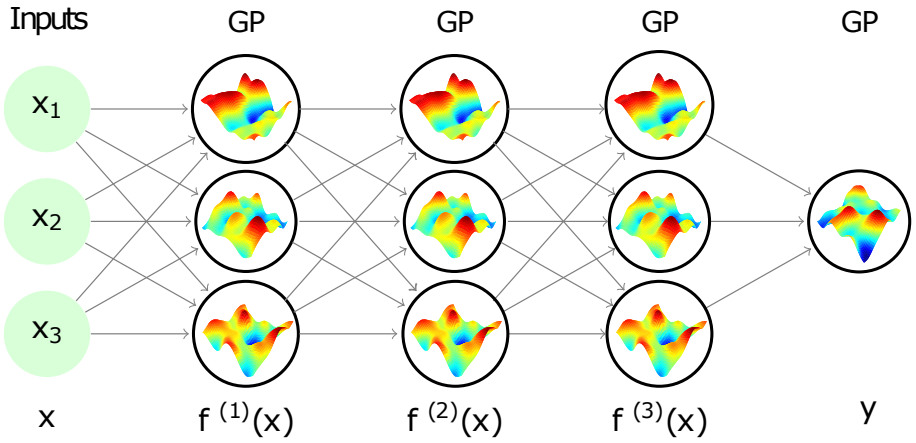


Figure 2.5: DGP visualization consisting of three hidden layers and a GP at the output layer to predict the target value y [14].

The main problem of this approach is the large amount of parameters introduced per layer. We solve this issue using Sparse GPs [15] instead of regular GPs. Using variational inference, we can obtain deep hierarchies of sparse GPs.

To determine the size of the latent space, we can use Automatic relevance determination (ARD). ARD consists of a covariance function that can adapt to the data set and apply a different weight w for each dimension. This can reduce complex models by eliminating the non necessary dimensions by setting zero weight to them.

This covariance function will take the following form:

$$k(x_i, x_j) = \sigma_{ard}^2 e^{-\frac{1}{2}\sum_{q=1}^{Q} w_q (x_{i,q} - x_{j,q})^2} \tag{2.13}$$

The Bayesian treatment of this model can be very challenging due to the covariance function being non linear. Luckily, recent variational inference methods make the approximation of the Bayesian training procedure possible. For a detailed explanation see Bayesian Training (Damianou, Lawrence 2.3) [17] and Doubly Stochastic Variational Inference for Deep Gaussian Processes (Hugh Salimbeni, Marc Deisenroth) [18].

With the introduction of the sparse GPs, we are going to use variational inference to optimize the lower bound on $logp(y)$ [18]. We need to maximize the lower bound on $logp(y)$ defined by:

$$\mathcal{L} = \sum_{i=1}^{N} E_q[logp(y_i|f_i^L)] - \sum_{l=1}^{L} KL[q(u')|p(u')], \tag{2.14}$$

with f_i^L representing the prediction of the model in the last layer and q the variational posterior under the form:

$$q(f,u) = p(f|u;X,Z)\mathcal{N}(u|m,S), \tag{2.15}$$

$p(f|u;X,Z)$ indicating that X and Z are the input locations for f and u.

The expectations can be approximated by Monte Carlo. To obtain the desired estimate, we average the results obtained by assigning multiple values to an uncertain variable, this process is known as repeated random sampling.

To obtain predictions we propagate samples through the DGP hierarchy to compute the predictive distribution as a mixture of Gaussians.

Deep Gaussian processes have several advantages over regular Gaussian processes. They have an automatic non parametric design, are more flexible, they provide better uncertainty estimates and more accurate predictions. As a counterpart they require complicated approximate inference methods and are computationally more expensive [14].

2.3. Support Vector Regression

Support Vector regression is the equivalent to support vector machines applied to regression problems that may manage better high noise conditions.

2.3.1. Support Vector Machines

Sparse kernels have the advantage of only evaluate a subset of the training data points to obtain the kernel function, having a less computational expensive training phase.

First introduced by Vapnik [19] in 1992 as a two-class classifier. Support Vector Machine (SVM) is a popular sparse kernel machine learning classifier that works as a decision machine. Therefore not providing posterior probabilities and using support vector for new input predictions.

After considering the needs to solve problems involving more than two classes, Vapnik proposed to use multiple SVMs under the name one-versus-the-rest. This approach has its limitations as the input can be assigned to multiple classes simultaneously leading to poor results. A lot of methods were proposed to fix this issue until Allwein et al. (2000) Used the approach of Dieterich and Bakiri to generalize the one-versus-the-rest method in which we train the individual classifiers with non specific partitions, giving robustness.

2.3.2. Application to regression

Support Vector Machines can be used for regression under the name Support Vector Regression. The main difference is that they construct an $\varepsilon - tube$ based on the intensity supporting on vectors that lie either on the tube or outside. They represent a great option because they use Gaussian Kernels and an $\varepsilon - intensity$ parameter gives them more flexibility. They are capable to adapt better to specific noise conditions. But, they have an issue, the training cost is high as we need to use a method such as cross validation to optimize the different parameters described as follow [9]:

- gamma = Kernel coefficient for 'RBF'
- c = Regularization parameter.
- epsilon = Distance permitted to enter the epsilon-tube without penalty.

The problem can be seen as an optimization problem in which we have to minimize an error function to leave only this $\varepsilon - tube$. To do so, we introduce two slack variables to each data point $\xi_n >= 0$ and $\hat{\xi}_n >= 0$.

A point t lies inside the $\varepsilon - tube$ if $y(x_n) - \varepsilon <= t <= y(x_n) + \varepsilon$. Giving also the conditions for a point to not be inside the $\varepsilon - tube$ that will need to be minimized.

- $tn <= y(X_n) + \varepsilon + \xi_n$
- $tn >= y(X_n) - \varepsilon - \hat{\xi}_n$

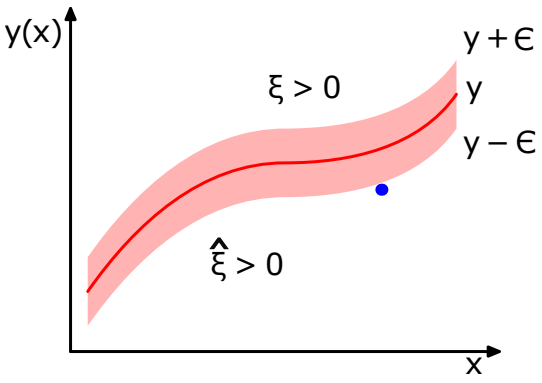


Figure 2.6: Illustration of SVM regression, showing the regression curve together with the insensitive 'tube' (Bishop 7.1 [1]).

The Support Vector Regression error function would look as follows:

$$C\sum_{n=1}^{N}(\epsilon_n + \hat{\epsilon}_n) + \frac{1}{2}||w||^2, \tag{2.16}$$

with $\epsilon_n >= 0$ and $\hat{\epsilon}_n >= 0$, two more constraints to be minimized. w represents the parameters in the extended space $\phi(x)$.

The minimization of all this terms can be achieved using Lagrange multipliers to finally obtain a dual problem that consists in maximizing the following:

$$L(a, \hat{a} = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a_m})k(x_n, x_m) - \varepsilon\sum_{n=1}^{N}(a_n + \hat{a}_n) + \sum_{n=1}^{N}(a_n - \hat{a_n})t_n, \tag{2.17}$$

with $k(x, x') = \phi(x)^T\phi(x')$ being the kernel. This maximization is constrained by:

- $0 <= a_n <= C$
- $0 <= \hat{a}_n <= C$

Allowing us to make predictions for new inputs with:

$$y(x) = \sum_{n=1}^{N}(a_n - \hat{a}_n)k(x, x_n) + b \tag{2.18}$$

The following figure shows the adjustment to a sinusoidal synthetic data set using Gaussian kernels.

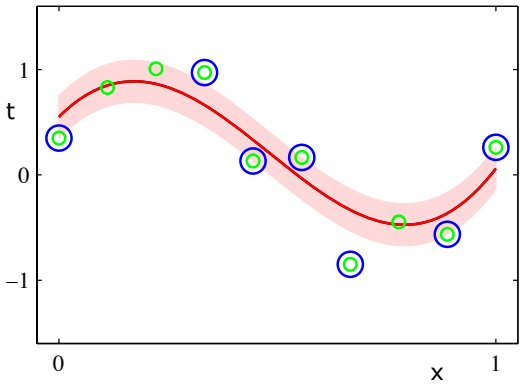


Figure 2.7: The red line shows the prediction regression curve. The green circles represent the data points. The blue circles represent the data points with support vectors. (Bishop 7.1 [1]).

For a detailed explanation see (Bishop 7.1.4 [1]).

3

Experiments

In this section we are going to discuss the experiments using the data corresponding of the Sotavento wind energy park. we are going to distinguish two subsections. The first subsection describes how Gaussian Processes, Deep Gaussian processes and Support Vector Regression, are configured, trained and evaluated over the data for the current energy production values. In the second subsection, we are going to use what we learned and try to predict future eolian energy production values based on the previous ones adding also tests using deep Gaussian processes with non Gaussian noise.

The metrics used in all the experiments to evaluate the different models are the Root Mean Squared Error (RMSE) and the Mean Average Error (MAE). The Negative Log Likelihood is used for Gaussian Processes and Deep Gaussian processes but not for Support Vector Regression as we are unable to compute it as it takes into account the confidence of the prediction.

The Root Mean Squared error is computed as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}. \tag{3.1}$$

The Mean Absolute Error is computed as follows:

$$MAE = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{N}. \tag{3.2}$$

The Negative Log Likelihood is computed as follows:

$$NLL = -logp(y*|x*), \tag{3.3}$$

with p being the predictive distribution of the model. A Gaussian distribution for GPs and a mixture of Gaussian distributions for DGPs.

For all the previous equations $\hat{y}_i$ represents the true values, y_i the predictions and N the size of the sample.

3.1. Prediction of current wind energy production

In this section we are going to describe the procedure to obtain predictions for the current wind energy production and compare the results obtained by the different approaches.

3.1.1. Tools used

The programming language chosen to run the code is Python 3 as it is one of the most popular programming languages in machine learning. Our machine learning library of choice is TensorFlow, as it is also the main machine learning library that provides a big amount of features and flexibility. Moreover, Tensor board is used to obtain metrics for the execution in real time. The likelihood functions used are obtained from the gpflow library. All the experiments were run in the Scientific Computational Centre of the Autonomous University of Madrid using slurm.

We are going to enumerate the libraries containing the the kernel method implementations used in our experiments.

- **Gaussian processes and Deep Gaussian Grocesses:** Hugh Salimbeni and Marc Deisenroth implementation under the name Doubly Stochastic Variational Inference for Deep Gaussian Processes in 2017 [18].
- **Support Vector Regression:** sklearn's library implementation [9].

We will run over the parameters used and a brief explanation of the code functionality.

Doubly Stochastic Variational Inference for Deep Gaussian Processes

First of all we load the data set and preprocess it for training. We divide it into 5 different random splits with a ratio of 70 % for training and 30 % for evaluation proceeding with a normalization phase for all the data set.

Afterwards, we calculate the 100 inducing points to be used with kmeans:

$$Z = kmeans2(X, 100, minit =' points')[0]$$

Then we set up the kernel configuration.

We are going to use the RBF kernel for all the layers with an added white noise function. For the first layer kernel we are going to use the total amount of attributes as the number of dimensions (3479) and for the rest we are going to use 30 dimensions.

The initial value for the lengthscales is computed by calculating the likelihood over the squared distances on a random sample of size 1000. For the rest of the layers, as it is an step down in dimensions, we use the pca projection to obtain the sample to compute the likelihood over the squared distances. This value is constantly being optimized every iteration, but having a good initial value is crucial to ob-

tain better results. This lengthscales, combined with Automatic Relevance Determination, consisting of a lengthscale vector with a lenghtscale per dimension, obtain the best results.

We will split the execution in batches of size 100 to allow parallelization over the batch in TensorFlow and decrease the execution time.

The noise variance for the model is started at 0.05.

The optimizer used for the minimization of the error function is the Adam Optimizer as it is one of the best adaptive optimizers for stochastic gradient descent.

After testing out a lot of different parameters configurations, we decided that the one mentioned is the one with better results at 150.000 iterations.

Support Vector Regression

For support vector regression, we followed the same approach with a ratio of 70 % for training and 30 % for evaluation splits proceeding with the normalization phase for all the data set.

The number of iterations can be set in this case to -1 and the model will stop the learning phase upon the learning loss function converges.

To determine the best values for the parameters C, γ and ϵ we used grid search which uses cross validation and leaves you with the best trained model based on a metric. In this case we use the mean absolute error to determine the best parameter configuration.

By doing some testing with different ranges of parameters we determined that the best values are always among this possibilities:

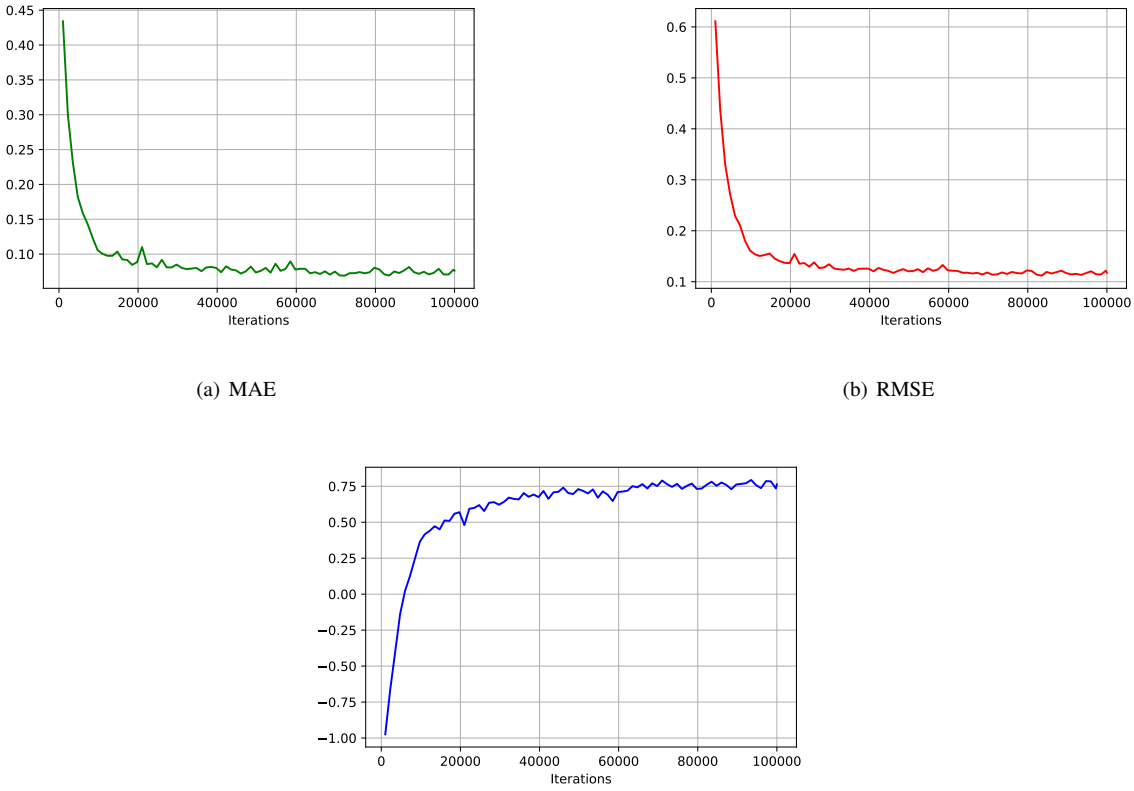
C	γ	ϵ
7.3890	3.3546e4	3.3546e4
20.0855	8.0473e4	8.0473e4
54.5981	1.93045e3	1.93045e3
148.4131	4.63092e3	4.63092e3
403.4287	1.1109e2	1.1109e2

Table 3.1: Grid Search SVR

3.1.2. Results Obtained

After setting everything up, we proceeded with multiple executions, both with Support Vector Regression and Gaussian Processes altering the number of layers to test different depths in Deep Gaussian Processes.

Gaussian Processes



(a) MAE

(b) RMSE

(c) NLL

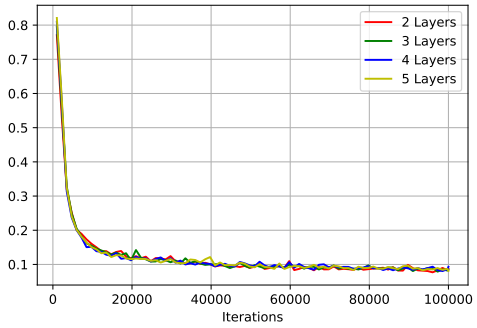
Figure 3.1: GPs MAE, RMSE and NLL for Current Wind Energy Predictions

Time (hr)	RMSE	MAE	NLL
17.2	0.1179	0.0762	0.7602

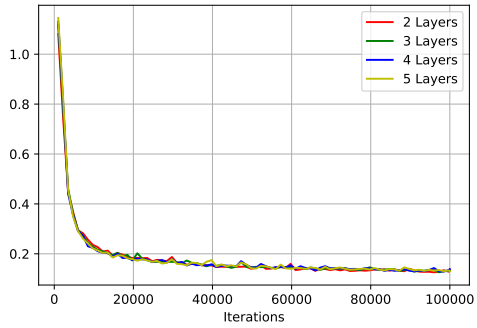
Table 3.2: GP Final Results for Current Predictions

We observe very low RMSE (0.1179) and MAE (0.0762) values which means that the model has a high accuracy at giving energy production predictions for new values. We can observe also that it took 17.2 hours train and (X) hours to validate. We will now proceed to show the results for Deep Gaussian Processes to see if they improve over the one layer Gaussian Process.

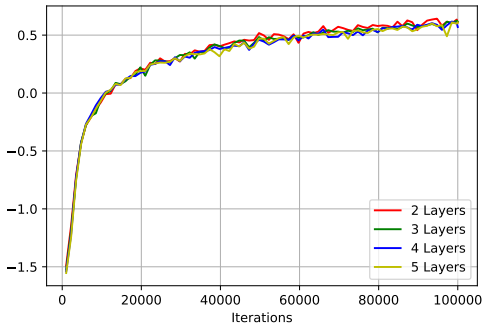
Deep Gaussian Processes



(a) MAE



(b) RMSE



(c) NLL

Figure 3.2: DGPs MAE, RMSE and NLL for Current Wind Energy Predictions

N. Layers	Time (hr)	RMSE	MAE	NLL
2	17.2	0.1320	0.0855	0.6142
3	22.6	0.1312	0.0841	0.6111
4	16.3	0.1355	0.0884	0.5858
5	19.6	0.1321	0.0845	0.5928

Table 3.3: DGP Final Results for Current Predictions

We noticed a slight increase in both RMSE and MAE values which means that Deep Gaussian Processes weren't able to adapt better than a single layered Gaussian process.

Support Vector Regression

Time (hr)	RMSE	MAE	NLL
16.4	0.4711	0.2703	NA

Table 3.4: SVR Final Results for Current Predictions

After observing the three approaches, we can conclude that Sparse GPs are the most optimal. In this case, DGPs are not adding any advantage over GPs and SVR performed poorly in comparison.

3.2. Prediction of future wind energy production

In this section we are going to describe the procedure to obtain predictions for the future wind energy production and compare the results obtained by the different approaches.

3.2.1. Adapting the data set

The environment to run the experiments is the same as the on we used for the current energy production prediction. The main change needed to train the model to predict data in the future is to shift the data set by the amount needed. This means, if we want to predict data 1 hour in the future, and the data set is composed of observations every hour, the energy produced on the second hour is assigned as the result of the first hour and so on. Therefore we will lose one row of data, as the last observation won't have any future production value. As we have 8760 rows, this won't be a significant loss, but we need to take it into consideration for more extensive predictions.

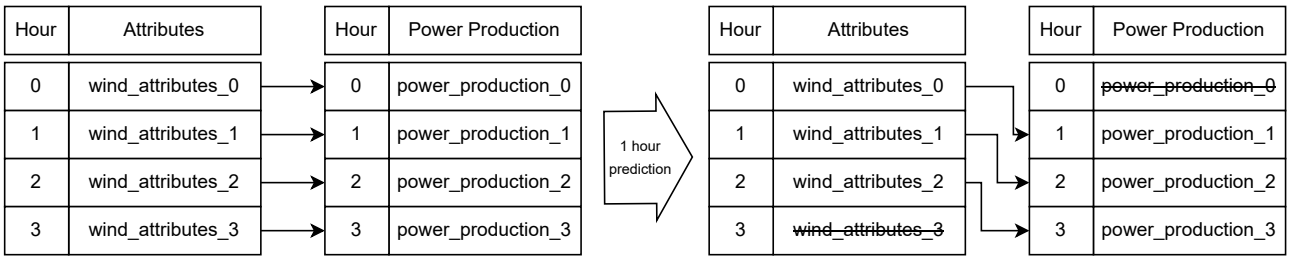


Figure 3.3: Example of data adaptation for predictions in 1 hour

We will repeat this process for 1 hour, 12 hours and 24 hours and use the new data sets in all three models: Gaussian Processes, Deep Gaussian Processes and Support Vector Regression.

Moreover we considered using a Gaussian Processes with Laplace noise instead of Gaussian noise, as Support Vector Regression was using this probability distribution. We are going to describe it briefly and compare it to the Gaussian distribution.

The Laplace probability density function is defined by:

$$f(x|\mu, b) = \frac{1}{2b} exp\left(-\frac{|x-\mu}{b}\right), \tag{3.4}$$

with μ a location parameter and b a scale parameter.

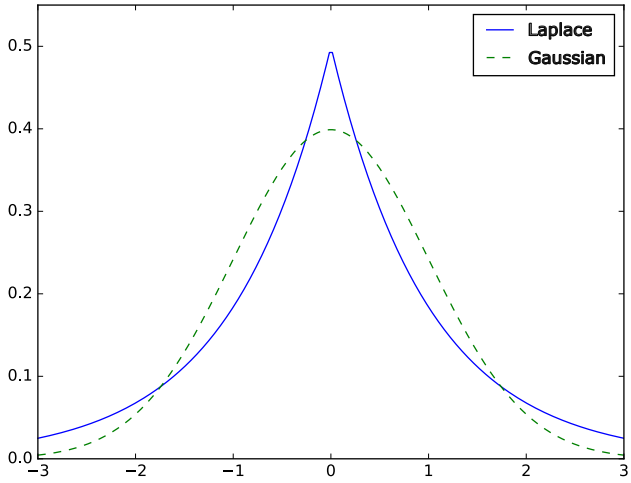


Figure 3.4: Comparison between the Gaussian distribution and Laplace distribution. [20]

The Laplace likelihood has been implemented as follows inheriting from gpflow.likelihoods [21]:

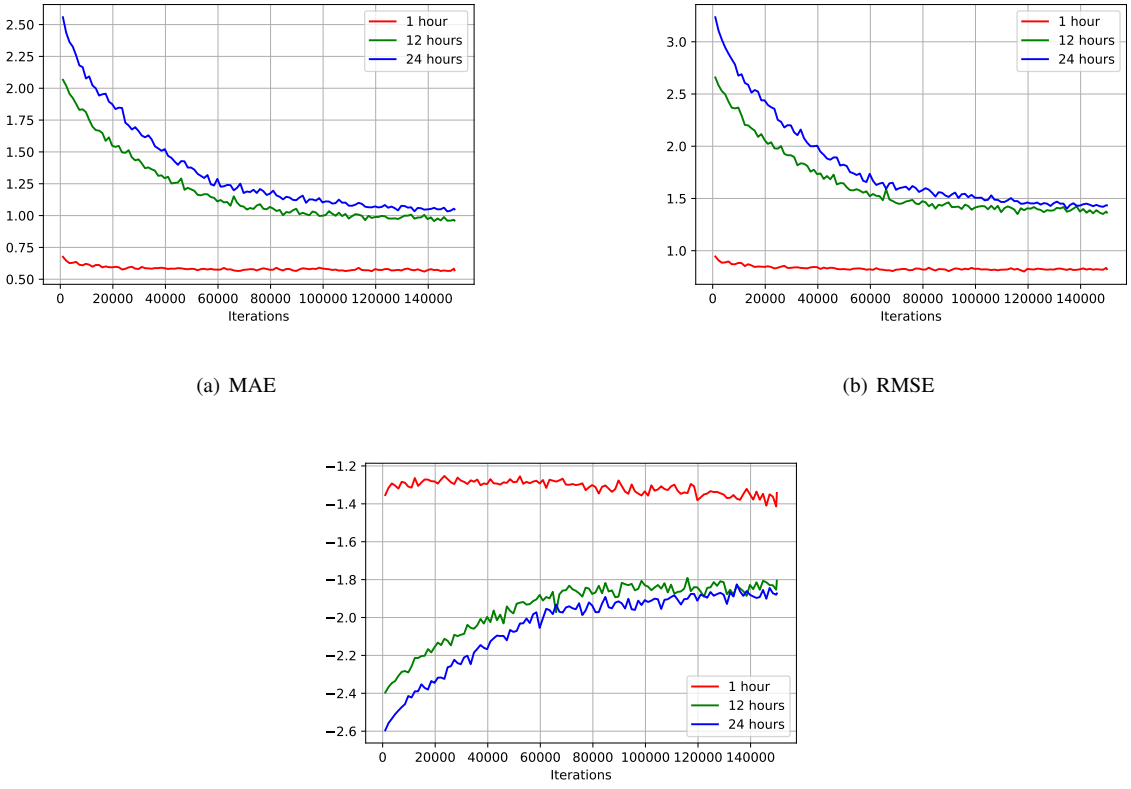
Code 3.1: Laplace Likelihood implementation.

```
class Laplace(Likelihood):
    def __init__(self, b=0.01, **kwargs):
        super().__init__(**kwargs)
        self.b = Parameter(
            b, transform=transforms.positive, dtype=settings.float_type)
    @params_as_tensors
    def logp(self, F, Y):
        return -tf.abs(F -Y) / self.b -tf.log(2. *self.b)
    @params_as_tensors
    def conditional_mean(self, F):
        return tf.identity(F)
    @params_as_tensors
    def conditional_variance(self, F):
        return (2 *self.b)
```

3.2.2. Results Obtained

After adapting the data set for different hour delays, we proceeded with multiple executions, both with Support Vector Regression and Gaussian Processes altering the number of layers to test different depths in Deep Gaussian Processes.

Gaussian Processes



(a) MAE

(b) RMSE

(c) NLL

Figure 3.5: GPs MAE, RMSE and NLL for Wind Energy Predictions

Hour Delay	Time (hr)	RMSE	MAE	NLL
1	15.8	0.8237	0.5707	-1.3242
12	23.2	1.3565	0.9607	-1.806
24	15	1.4320	1.048	-1.8740

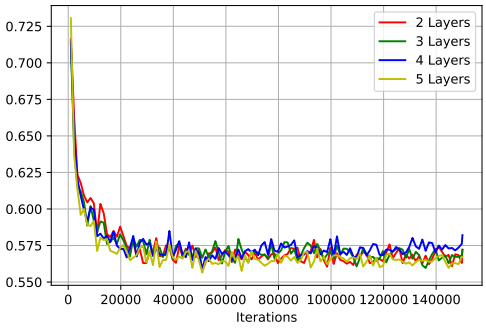
Table 3.5: GP Final Results for Predictions

We can observe an increase of more than 7 times in both RMSE and MAE for predictions of 1 hour of delay compared to the prediction of current production of wind energy. Increasing the delay results in worse results as the complexity grows.

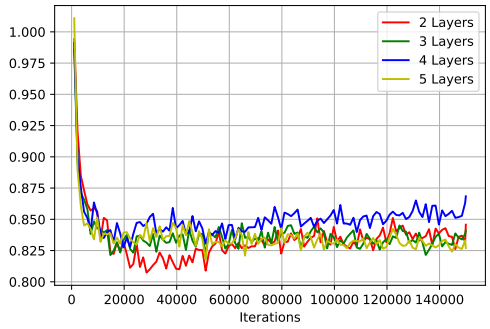
Deep Gaussian Processes

For Deep Gaussian Processes we will divide the results in hours of predictions as we have different depth executions which can reduce the visibility of the graphs.

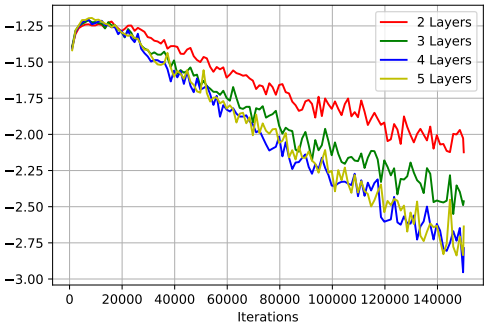
1 hour in the future



(a) MAE



(b) RMSE



(c) NLL

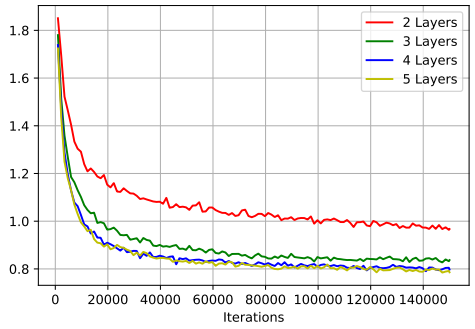
Figure 3.6: DGPs MAE, RMSE and NLL for 1 hour Wind Energy Predictions

N. Layers	Time (hr)	RMSE	MAE	NLL
2	40.4	0.8387	0.5683	-2.0790
3	53.7	0.8379	0.5694	-2.4490
4	93.5	0.8615	0.5774	-2.7930
5	116.4	0.8286	0.5656	-2.7180

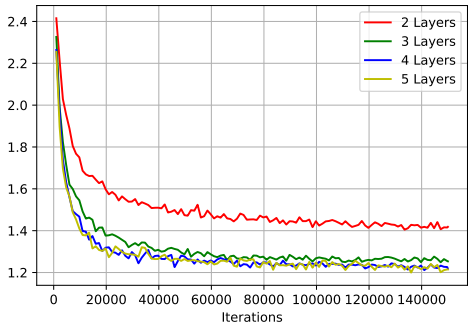
Table 3.6: DGP Final Results for 1 hour Predictions

As seen in GPs, the results are worse predicting future energy production times.

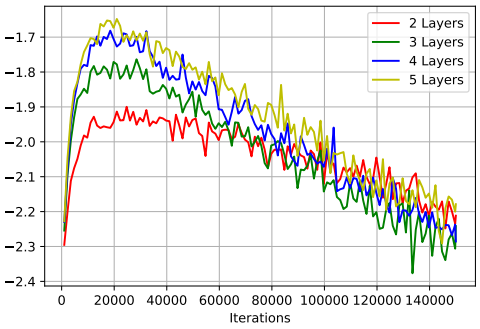
12 hours in the future



(a) MAE



(b) RMSE



(c) NLL

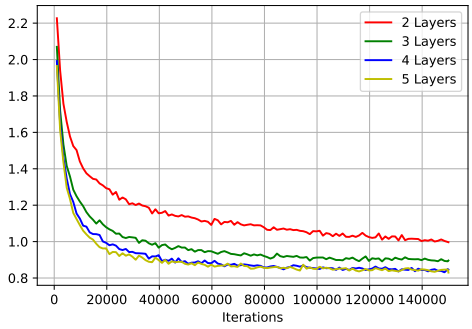
Figure 3.7: DGPs MAE, RMSE and NLL for 12 hours Wind Energy Predictions

N. Layers	Time (hr)	RMSE	MAE	NLL
2	32.6	1.4170	0.9687	-2.2110
3	57.2	1.2570	0.8356	-2.2710
4	97.3	1.2240	0.8005	-2.2610
5	114.9	1.2150	0.7890	-2.1850

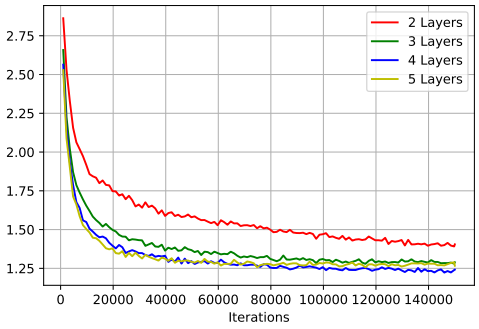
Table 3.7: DGP Final Results for 12 hours Predictions

As we increase the delay, the results are worse.

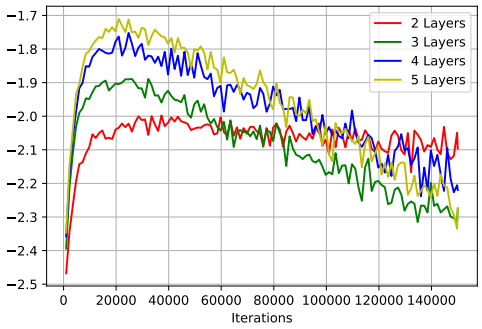
24 hours in the future



(a) MAE



(b) RMSE



(c) NLL

Figure 3.8: DGPs MAE, RMSE and NLL for 24 hours Wind Energy Predictions

N. Layers	Time (hr)	RMSE	MAE	NLL
2	32.8	1.403	0.9966	-2.090
3	49.2	1.286	0.8952	-2.2890
4	97.3	1.236	0.8434	-2.2050
5	114.8	1.276	0.8386	-2.3840

Table 3.8: DGP Final Results for 24 hours Predictions

We can observe that the results are worse than with a single layered GP. Nevertheless, as we increase the number of layers of the DGPs, we can observe that the results tend to be better. The ability to adapt to a more complex data set and the computational cost grow with a higher number of layers.

Support Vector Regression

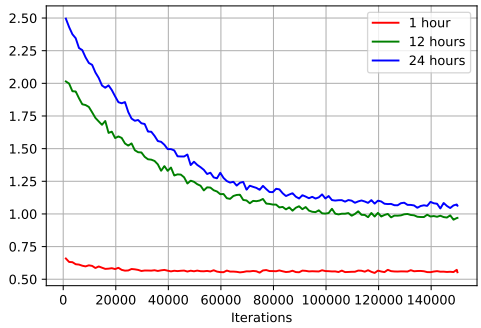
Hour Delay	Time (hr)	RMSE	MAE	NLL
1	16.6	0.7993	0.5292	NA
12	19.9	1.1966	0.7567	NA
24	19.3	1.2528	0.8376	NA

Table 3.9: SVR Final Results for Predictions

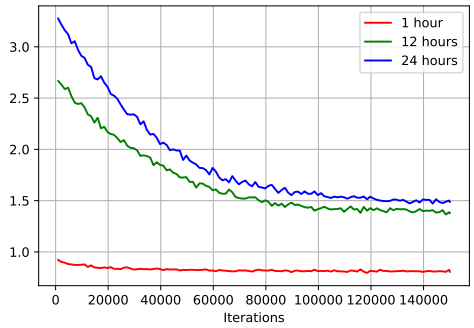
With Support Vector Regression we obtain the best results. Moreover, the execution time is also smaller. This can be due to the ability of SVM to adapt to high noise conditions as it is more flexible than the gaussian processes approach with inducing points.

GP with Laplace noise

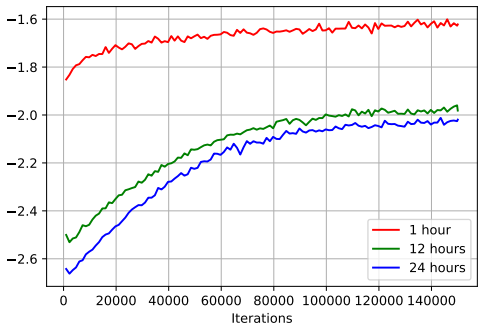
We tried to use Gaussian Processes with Laplace noise instead of Gaussian noise as it is the one used in SVR to try to improve the results. We use a one layered Gaussian Process to measure the possible improvement.



(a) MAE



(b) RMSE



(c) NLL

Figure 3.9: Laplace GPs MAE, RMSE and NLL for Wind Energy Predictions

Hour Delay	Time (hr)	RMSE	MAE	NLL
1	15	0.8072	0.5536	-1.6220
12	16.1	1.3840	0.9698	-1.9730
24	16.1	1.4910	1.0630	-2.0230

Table 3.10: Laplace GP Final Results for Predictions

We observe very similar metrics to Gaussian Processes with Gaussian noise or in some cases slightly worse as we increment the prediction delay.

4

Conclusions and Future Work

In this section we will note the different conclusion obtained from the experiments section and future work discussion.

4.1. Conclusions

The constant expansion of renewable energy production has increased the resources and time invested on the subject. Wind energy is one of the main renewable energy production strategies used by most countries. The main issue with wind energy is the high variability depending on the meteorological variables. For this reason, the prediction of wind energy is very impactful on the possible benefits to be obtained by this source of energy.

In this project we focused on three main approaches consisting of Gaussian Processes, Deep Gaussian Processes and Support Vector Regression.

After observing the results, we conclude that Gaussian Processes perform better for current wind energy production. For future energy production, Support Vector Regression resulted in better results. A possible reason can be the higher flexibility of SVR as GPs use inducing points limiting their flexibility. Moreover, the loss of the $\epsilon - intensity$ parameter makes SVR more robust adapting to more specific noise conditions. Deep Gaussian Processes offered decent results for the prediction of future energy production. Adding more layers could result in even better results. The main issue is the computational cost as it is highly affected by the number of layers used. We also tried assuming Laplacian noise in the context of GPs. We observed very similar metrics to Gaussian Processes with Gaussian noise or in some cases slightly worse as we increment the prediction delay.

Overall, Gaussian Processes are a robust model to predict current wind energy production, but for future energy production, other approaches such as Support Vector Regression are more effective.

4.2. Future Work

There are many other machine learning models that could be tested in this scenario to obtain a more extensive comparison, for example Random Forest or Convolutional Neural Networks. Also, we could retrieve different metrics to have multiple comparisons at once and decide which model to use in more specific situations.

Moreover, we could consider gathering data from various parks located in multiple positions of the earth, with different climates, atmospheric pressures and weather conditions. This will give more reliability to our results and would be more useful for a wider range of applications. In addition, having a larger data set could give more precision to our model.

Ultimately, this field is open to further analysis and development of machine learning techniques to optimize the chain of production of renewable energy.

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[2] B. D. M. F. M. N. X. L. Y. Z. Raphael Anaadumba, Qi Liu, "A renewable energy forecasting and control approach to secured edge-level efficiency in a distributed micro-grid," 2021. (source).

[3] E. P. R. Service, "Fit for 55 package," 2021. (source).

[4] C. of the EU, "Fit for 55: Council agrees on higher targets for renewables and energy efficiency," 2022. (source).

[5] J. R. D. Carlos Ruiz, Carlos M. Alaíz, "Multitask support vector regression for solar and wind energy prediction," 2020. (source).

[6] S. G. Foundation, "Sotavento - parque eólico experimental." (source).

[7] C. A. María, "Modelos de aprendizaje automático en la predicción de viento a corto plazo," 2020. (source).

[8] Y.-T. W. Fernando Porté-Agel, Hao Lu, "Interaction between large wind farms and the atmospheric boundary layer," 2014. (source).

[9] S. learn, "Sklearn epsilon-support vector regression." (source).

[10] D. la pompa porras Víctor, "Procesos gaussianos para problemas de regresión y estimacioón de la incertidumbre," 2018.

[11] C. K. I. W. C. E. Rasmussen, *Gaussian Processes for Machine Learning*. the MIT Press, 2006.

[12] S. Roweis, "Gaussian identities," 1999. (source).

[13] E. C. G. Merchán, "Advanced methods for bayesian optimization in complex scenarios," 2021. (source).

[14] D. H. Lobato, "Slides gps."

[15] M. K. Titsias, "Variational learning of inducing variables in sparse gaussian processes," 2011. (source).

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] N. D. L. Andreas C. Damianou, "Deep gaussian processes," 2012. (source).

[18] M. P. D. Hugh Salimbeni, "Doubly stochastic variational inference for deep gaussian processes," 2017. (source).

[19] V. N. V. Bernhard E. Boser, Isabelle M. Guyon, "A training algorithm for optimal margin classifiers," 1992. (source).

[20] J. D. Cook, "Normal approximation to laplace distribution?." (source).

[21] Secondmind, "Gpflow documentation." (source).

Acronyms

ARD Automatic relevance determination.

DGP Deep Gaussian Processes.

FITC Full Independent Training Conditional.

LOO Leaving-One-Out.

MAE Mean Average Error.

RBF Radial Basis Functions.

RMSE Root Mean Squared Error.

SVM Support Vector Machines.

SVR Support Vector Regression.

VFE Variational Free Energy.


UÁM
Universidad Autónoma
de Madrid