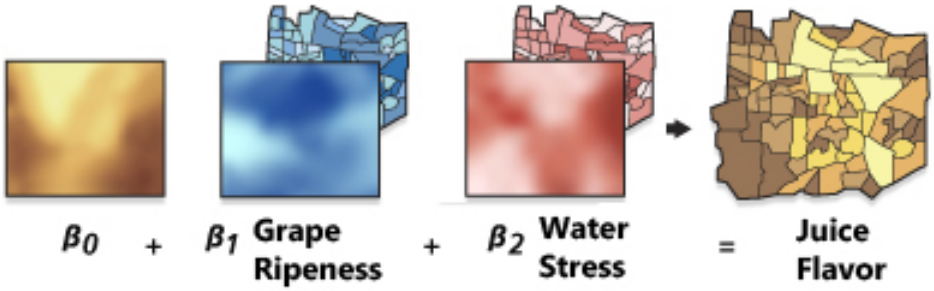
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
MSC IN DATA SCIENCE




# A spatial analytics approach for the ripeness estimation inside a vineyard

*Student Name:*
**Karvouniaris Alexander**

*Supervisor:*
**Mihalis Titsias**

October 15, 2017

# Abstract

Spatial analysis can perhaps be considered to have arisen with early attempts at cartography and surveying but many fields have contributed to its rise in its modern form. Biology contributed through botanical studies of global plant distributions and local plant locations, ethological studies of animal movement, landscape ecological studies of vegetation blocks, ecological studies of spatial population dynamics, and the study of biogeography. Epidemiology contributed with early work on disease mapping, notably John Snow's work of mapping an outbreak of cholera (Figure 1), with research on mapping the spread of disease and with location studies for health care delivery. Statistics has contributed greatly through work in spatial statistics. Economics has contributed notably through spatial econometrics. Geographic information system is currently a major contributor due to the importance of geographic software in the modern analytic toolbox. Remote sensing has contributed extensively in morphometric and clustering analysis. Computer science has contributed extensively through the study of algorithms, notably in computational geometry. Mathematics continues to provide the fundamental tools for analysis and to reveal the complexity of the spatial realm, with machine learning being the cutting edge of this arsenal of tools. Lastly, this thesis is a product of Owleyes' professional imaging through Unmanned Aerial Vehicle, Mr. Zoumpoulis' viticultural expertise, prof. Titsias' never-ending assistance along the way and author's personal interest to blend machine learning with real world tasks.

Figure 1: London Cholera Map, Dr. John Snow, 1854, one of the first uses of map-based spatial analysis.

# Contents

*"If agriculture is to continue to serve the world, it needs to become more like manufacturing."*

Geoffrey Carr,
Economist-Science Editor

# 1

# Vineyard and the need for accuracy

Agriculture is the first and foremost industry that humanoid developed in order to support its very own existence and evolution. Technology has aided this endeavor since the very beginning. Handheld tools were the standard hundreds of years ago, then the industrial revolution brought about the cotton gin (a machine to separate cotton from its seeds). The 1800's consequently brought grain elevators, chemical fertilizers and the first gas-powered tractor, while the late 1900's introduced exploitation of satellite metrics by farmers to plan their work. As in any historical example of this nature, the evolution of agricultural know-how is going faster and wider too. Since lately technology is again setting off to reshape both science and business sector, agriculture is steadily embracing the advances that emerge from bioscience, data science, as well as automation and robotics sector.

Among the goods that agriculture produces, grapes aimed for vinification display some interesting and uncommon characteristics. Inside a vineyard there is a lot of spatial variability. Two vines located 3 meters far apart can behave significantly differently in terms of vigor, grape color, ripeness etc. Ripeness, the main theme of this thesis, is one of the most debatable characteristics of grape. What exactly constitutes ripeness will vary depending on what style of wine is being produced (sparkling, still, fortified, rose, dessert wine, etc.) and what the winemaker and viticulturist personally believe constitutes ripeness. Once the grapes are harvested, the physical and

chemical components of the grape which will influence a wine's quality are essentially set, so struggling to achieve optimal ripeness may be considered the most crucial part in winemaking.

There are several factors that contribute to the ripeness of the grape. As the grapes go through veraison, sugars in the grapes will continue to rise as acid levels fall. The balance between sugar (as well as the potential alcohol level) and acids is considered one of the most critical aspects of producing quality wine so both the must weight and "total acidity", as well as the pH of the grapes, are evaluated to determine ripeness.

In order to track the reasons behind these differences in ripeness, one should search for the variables that affect the behavior of a vine. One of the primary factors influencing the ripening process of grapevine is the climate and weather. Sunlight and temperature are vital to the physiological functions of the grapevine (such as photosynthesis). Soil composition and humidity are also part of the set of all environmental factors that affect a crop's phenotype, the so called "Terroir".

After veraison has begun (grapes start to change their color), viticulturists are sampling several hundred individual berries picked from clusters throughout the vineyard in increasing intervals as the harvest draws closer. Regardless of whether we want to measure must weight, acid level or pH level, most of the techniques are destructives (we have to destroy the berry to measure the variable we need). As a consequence, viticulturists cannot measure but only a small portion of the vines. Using the results of this portion, they take decisions about the readiness to harvest the field.

The scope of this thesis is to use the aforementioned sampling to produce a thematic map with the predicted ripeness proxy for every vine in the field, combined with any insights our inference can offer. In this endeavor, spatial correlation between vines is highly crucial. Therefore, the following chapters will be invested in demonstrating the unique characteristics of spatial data and the optimal methods that are able to capture and integrate the spatial essence of our viticultural data.

*"Everything is related to everything else, but near things are more related than distant things."*

Waldo Tobler, First Law of Geography

# 2

# A few things about geospatial data

## 2.1 Types of spatial data and their characteristics

Whenever our data are associated with location, we think of them as spatial. When those locations are on the earth, we often are more specific and call them "geospatial". A location is most unambiguously described by a set of coordinates and a coordinate reference system (CRS). While we are already familiar with one CRS, the usual geographical latitude longitude reference system, it is important to note that there are plenty of ways to represent the location of something on the globe and each one of them has its own advantages and disadvantages.

Now, spatial phenomena can be generally thought of as either discrete locations (object with boundaries) or phenomena that can be observed everywhere (a spatial field). Discrete locations can be a research site, a town, a river, a road, even a country's borders. Examples of continuous phenomena include elevation, temperature or air quality.

The first discrete case is usually represented by **vector data**. Such data consists of a description of the "geometry" or "shape" of the locations, and normally also include variables with additional information about the locations. On the other hand, continuous data are usually represented by **raster**. In this case, we have an area combined with attribute values for every point inside the area. We discuss these two data types in turn.

### Vector Data

The main vector data types are **points, lines** and **polygons**. In all cases, the geometry of these data structures consists of sets of coordinate pairs (x,y). Points are the simplest case. Each point has one coordinate pair, and n associated variables. Let's take for example the European scenery and the countries that reside in the area. We can represent each country as a spatial point object and we could either represent each location through a dot or through an associated attribute like code name (Figure 2.1).

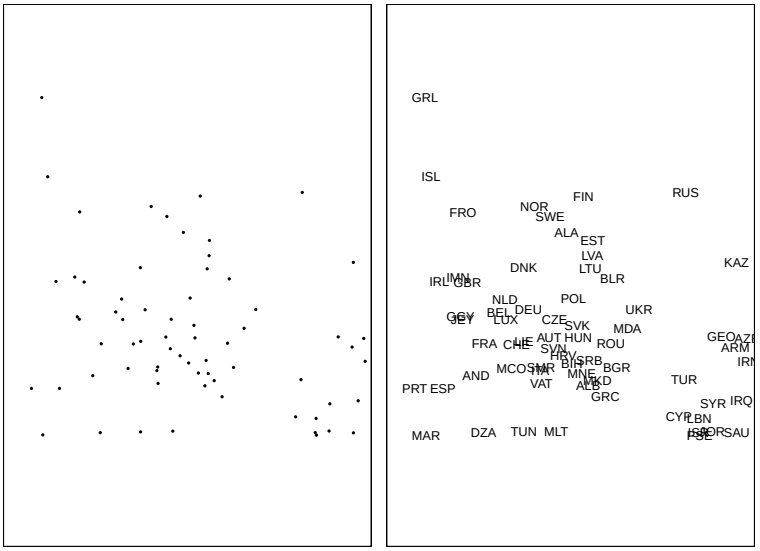



Figure 2.1: Spatial points for countries in the European area. Left: no attribute depicted, Right: Code of each country depicted.

The geometry of **lines** is a just a little bit more complex. First of all, the term "lines" refers to a set of one or more polylines (connected series of line segments). For example, in spatial analysis, a river and all its tributaries could be considered as a single "line" (but they could also be several lines, perhaps one for each tributary river) (Figure 2.2). Lines are represented as ordered sets of coordinates (nodes). The actual line segments can be computed (and drawn on a map) by connecting the points. Thus, the representation of a line is very similar to that of a multi-point structure. The main difference is that the ordering of the points is important, because we need to know which points should be connected.

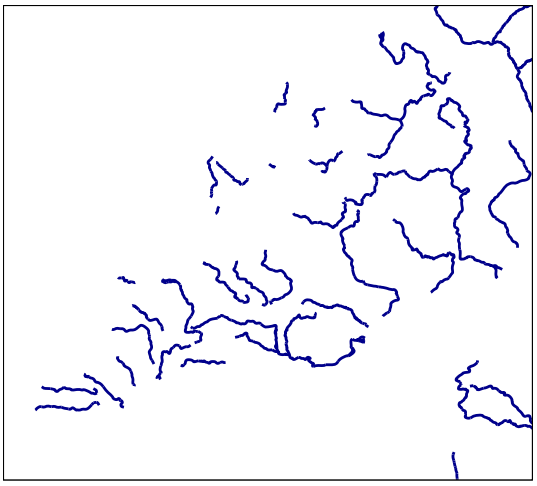


Figure 2.2: Spatial lines representing distinctive European rivers.

A **polygon** refers to a set of closed polylines. The geometry is very similar to that of lines, but to close a polygon the last coordinate pair must coincide with the first one. A complication with polygons is that they can have holes (that is, a polygon entirely enclosed by another polygon), that serve to remove parts of the enclosing polygon (for example to show an island inside a lake). Again, multiple polygons can be considered as a single geometry. For example the United States state of Hawaii consists of several islands. Each can be represented by a single polygon, but together they can be represented as a single (multi-) polygon of the Hawaiian islands. In Figure 2.3, we see different ways to depict the spatial polygons of each country in the European scenery.

### Raster Data

Raster data is commonly used to represent continuous variables. A raster divides the world into a grid of equally sized rectangles (referred to as cells or, in the context of remote sensing, pixels) that all have values (or a missing value) for the variables of interest. A raster cell value should normally represent the average (or majority) value for the area it covers, however we will present practical variations of this later on.

In contrast to vector data, in raster data the geometry is not explicitly stored as coordinates. It is implicitly set by knowing the spatial extent and the number or rows and columns in which the area is divided. From the extent and number of rows and columns, the size of the raster cells (spatial

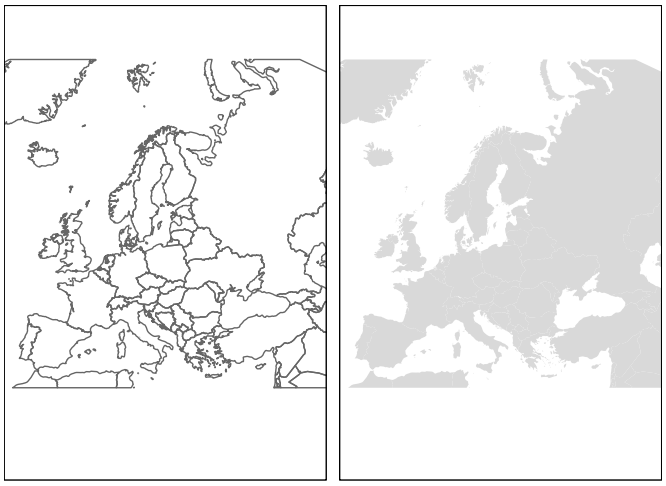


Figure 2.3: Spatial polygon for each country. Left: framing the borders of each polygon, Right: greying just the insides of polygons.

resolution) can be computed. While raster cells can be thought of as a set of regular polygons, it would be very inefficient to represent the data that way as coordinates for each cell would have to be stored explicitly. It would also dramatically decrease processing speed in most cases. In Figure 2.4 we depict tree density through a raster, while in Figure 2.5 we have created a map in which all kinds of spatial data coexist naturally.

## 2.2 Interpolation and Geostatistics

Geostatistical data are data that could in principle be measured anywhere, but that typically come as measurements at a limited number of observation locations: think gold grades in an ore body or particulate matter in air samples or, as in our case, sugar concentration in grape samples. The pattern of observation locations is usually not of primary interest, as it often results from considerations ranging from economical and physical constraints. The interest is usually in inference about variables that have not been measured such as maps of the predicted values, the uncertainties that go along with every prediction, or even inference of the process itself that generated the data. We could also say that geostatistics deals with the analysis of random fields $Z(s)$, with $Z$ random and $s$ the non-random spatial index. And in order to predict (interpolate) $Z$ at non-observed locations $s_0$, we have to estimate and model spatial correlation (covariance).

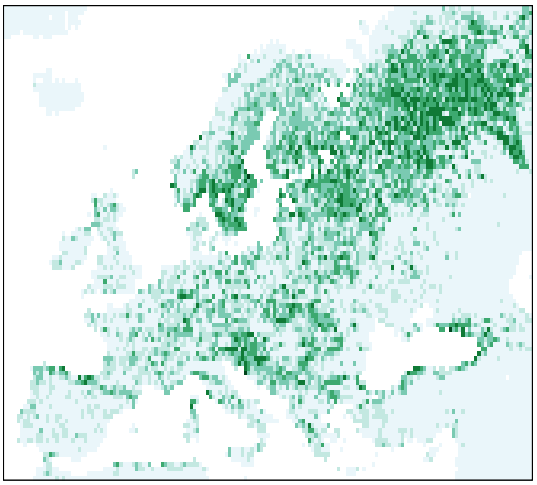


Figure 2.4: Raster data representing tree density inside European countries.

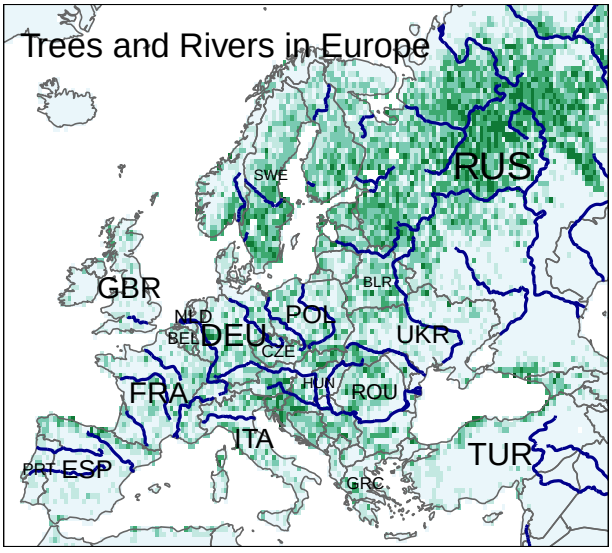



Figure 2.5: Map containing all types of spatial data - country surface through polygons, trees through raster, rivers through lines and names through points.

### 2.2.1 Why simple linear model is inefficient

From now on we will use the Meuse data set, created by Burrough and McDonell (1998), in order to demonstrate the concepts we intend to. Meuse is a classical geostatistical data set used frequently to demonstrate various spatial data analysis steps. The point data set consists of 155 samples of top soil heavy metal concentrations (ppm), along with a number of soil and landscape variables. The samples were collected in a flood plain of the river Meuse, near the village Stein. For now, let's assume that the variable of interest is zinc concentration.

| x | y | cadmium | copper | lead | zinc | elev | dist | om | ffreq | soil | lime | landuse | dist.m |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 181072 | 333611 | 11.7 | 85 | 299 | 1022 | 7.909 | 0.00135803 | 13.6 | 1 | 1 | 1 | Ah | 50 |
| 181025 | 333558 | 8.6 | 81 | 277 | 1141 | 6.983 | 0.01222430 | 14.0 | 1 | 1 | 1 | Ah | 30 |
| 181165 | 333537 | 6.5 | 68 | 199 | 640 | 7.800 | 0.10302900 | 13.0 | 1 | 1 | 1 | Ah | 150 |
| 181298 | 333484 | 2.6 | 81 | 116 | 257 | 7.655 | 0.19009400 | 8.0 | 1 | 2 | 0 | Ga | 270 |
| 181307 | 333330 | 2.8 | 48 | 117 | 269 | 7.480 | 0.27709000 | 8.7 | 1 | 2 | 0 | Ah | 380 |

Figure 2.6: Indicative rows of meuse dataset.

Now, the evident structure here is that zinc concentration is larger close to the river Meuse banks. In figure 2.7, we have also logged zinc concentration values in order to make the relationship more clear. If we let aside the geostatistical viewpoint for a second classical statistics approach, we could use this relationship to form a simple linear regression model

$$log(zinc) \sim sqrt(distance).$$

As we can see in figure 2.8 (left), after the square root transformation of distance variable, there is an acceptable linear relationship between the two variables. The graph on the right side depicts the residuals of the observations after fitting the linear model. Although the trend removes a large part of the variability, the residuals do not appear to behave as spatially unstructured or white noise: residuals with a similar value occur regularly close to another. This fact is the one responsible for the violation of linear model's assumption

$$e_i \overset{iid}{\sim} N(0, \sigma^2).$$

And this is why a geostatistical approach which includes estimation of spatial correlation is required in cases like this.

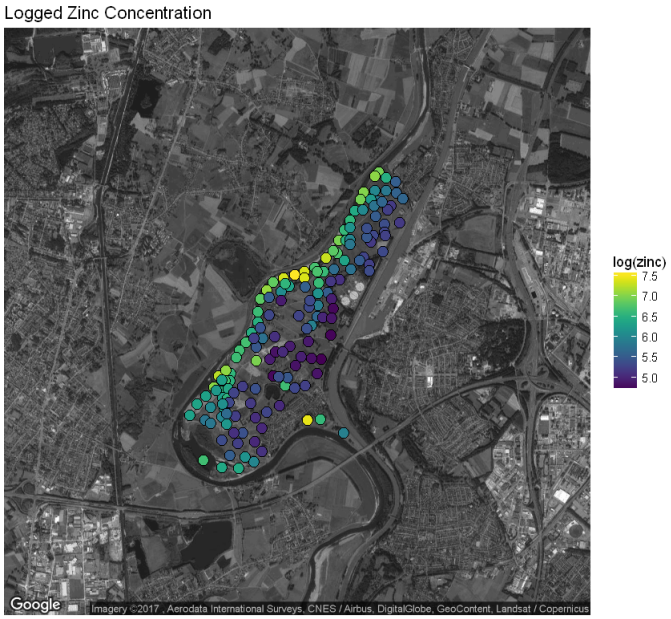



Figure 2.7: Soil samples and the respective logged zinc concentration by river meuse. Observe how zinc concentration is larger closer to the banks.

### 2.2.2 Spatial autocorrelation the classical way

We already discussed that the observations of a spatial phenomenon are not independent and identically distributed and therefore not compatible to classical statistics approach. And one way to grasp the notion of spatial correlation is to think of it as a simple extention of time correlation. In a classical time series perspective, an autoregressive process of order one (AR(1)) would be

$$y_t = \mu + \phi_1 y_{t-1} + \varepsilon_t,$$

suggesting a relationship between the values of the variable that are 1 period "far away". This notion of "distance" between values is intrinsic enough when we model time, especially in the discrete case. A model like this would also explicitly set variance of AR(1)

$$\gamma_0 = Cov(y_t, y_t) = \frac{\sigma^2}{1-\phi_1^2},$$

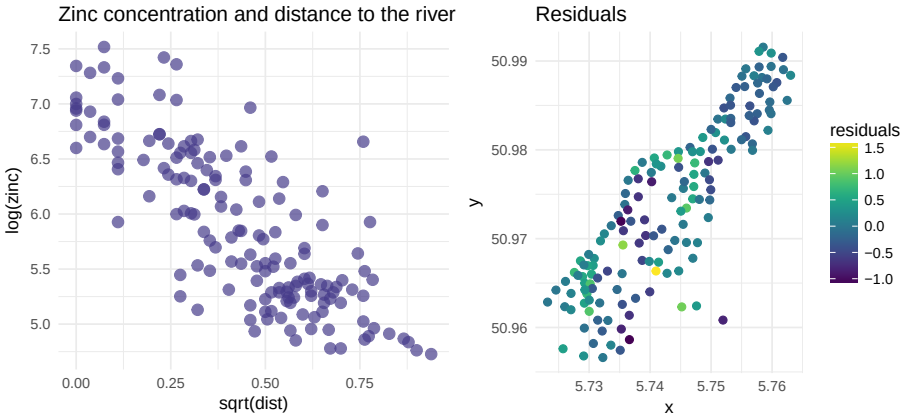



Figure 2.8: Left: Zinc concentration and distance to the river relationship. Right: Residuals of the fitted linear regression model.

autocovariance and autocorrelation at lag 1

$$\gamma_1 = Cov(y_t, y_{t-1}) = \phi_1 \frac{\sigma^2}{1-\phi_1^2}, \quad \rho_1 = \frac{\gamma_1}{\gamma_0} = \phi_1,$$

and autocovariance and autocorrelation at lag k in general

$$\gamma_k = Cov(y_t, y_{t-k}) = \phi_1^k \frac{\sigma^2}{1-\phi_1^2}, \quad \rho_k = \frac{\gamma_k}{\gamma_0} = \phi_1^k.$$

From modeling time to modeling space, we again search for relations between values of a certain distance, but this time we have a multitude of ways to perceive distance (difference, manhattan distance, euclidean distance or any kind of Minkowski distance, mahalanobis distance etc).

Now, in geostatistics the spatial correlation is traditionally modelled by the **variogram**. Here, the word variogram will be used synonymously with semivariogram. The variogram plots semivariance as a function of distance.

In standard statistical problems, correlation can be estimated from a scatterplot, when several data pairs $\{x, y\}$ are available. The spatial correlation between two observation of a variable $z(s)$ at locations $s_1$ and $s_2$ cannot be estimated, as only a single pair is available. To estimate spatial correlation from observational data, we therefore need to make stationarity assumptions before we can make any progress. One commonly used form of stationarity is *intrinsic stationarity*, which assumes that the process that

generated the samples is a random function $Z(s)$ composed of a mean and residual

$$Z(s) = \mu + e(s),$$

with a constant mean

$$E(Z(s)) = \mu,$$

and a variogram defined as

$$\gamma(h) = \frac{1}{2}E(Z(s) - Z(s+h))^2.$$

Under this assumption, we basically state that the variance of $Z$ is constant, and that spatial correlation of $Z$ does not depend on location $s$, but only on separation distance $h$. Then, we can form multiple pairs $\{z(s_i), z(s_j)\}$ that have (nearly) identical separation vectors $h = s_i - s_j$ and estimate correlation from them. If we further assume *isotropy*, which is direction independence of semivariance, we can replace the vector $h$ with its length, $||h||$.

Under this assumption, the variogram can be estimated from $N_h$ sample data pairs $z(s_i), z(s_i+h)$ for a number of distances (or distance intervals) $\tilde{h}_j$ by

$$\tilde{\gamma}(\tilde{h}_j) = \frac{1}{2N_h}\sum_{i=1}^{N_h}(Z(s_i) - Z(s_i+h))^2, \quad \forall h \in \tilde{h}_j$$

and this estimate is called the **sample variogram**.

A wide class of models (obviously parametric) is obtained when we insert predictors $X_j(s)$ in our modelling as in

$$Z(s) = \sum_{j=0}^{p} X_j(s)\beta_j + e(s) = X\beta + e(s),$$

with $X_j(s)$ the known spatial regressors and $\beta_j$ unknown regression coefficients. While we won't follow this path of modelling and will choose the bayesian non-parametric one, we will go on exploring the variogram for a bit more, as this process can offer us valuable insight we can use in the future.

### 2.2.3 Exploratory variogram analysis

A simple way to acknowledge that spatial correlation is present or not is to make scatter plots of pairs $Z(s_i)$ and $Z(s_j)$, grouped according to their separation distance $h_{ij} = ||s_i - s_j||$. This is done for the meuse dataset in

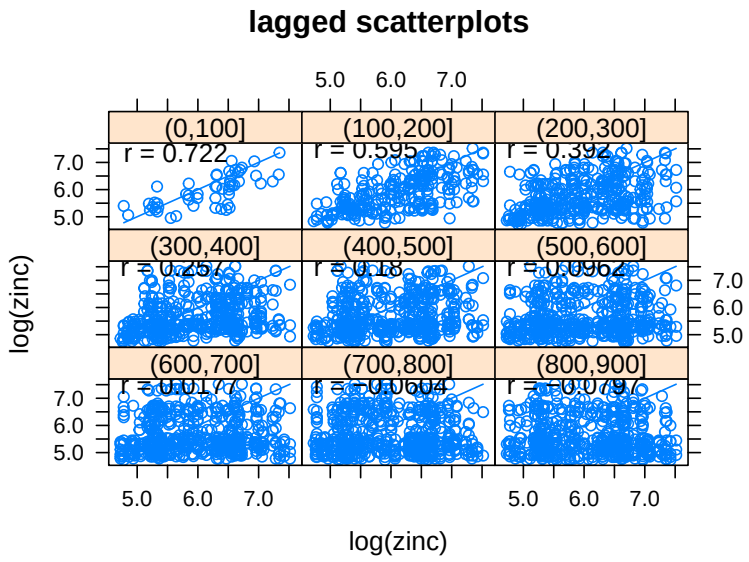



Figure 2.9: Lagged scatter plot for the log-zinc data in the meuse data set.

Figure 2.9, where the strip texts indicate the distance classes and sample correlation are shown in each panel.

A second way to explore spatial correlation is by plotting the variogram and the variogram cloud as in Figure 2.10. The variogram cloud is obtained by plotting all possible squared differences of observation pairs $(Z(s_i) - Z(s_j))^2$ against their separation distance $h_{ij}$.

The most important thing to consider in this classical approach is that the sample variogram $\tilde{\gamma}_h$ always contains a signal that results from the true variogram $\gamma_h$ and a sampling error, due to the fact that $N_h$ and $s$ are not infinite.

Thinking the classical way we could make hypothesis testing and confidence intervals to ensure that we can reject absence of spatial correlation and our estimation of variogram is valid. In general, however, concluding or even assuming that an underlying process is completely spatially uncorrelated is quite unrealistic for real, natural processes. A common case is that the spatial correlation is difficult to infer from sample data, because of their distribution, sample size, or spatial configuration. In the classical statistics way though, we are forced to estimate the covariance, erase the uncertainty our estimation introduced and infer from there on. The alternative bayesian approach here would blend a prior assumption with

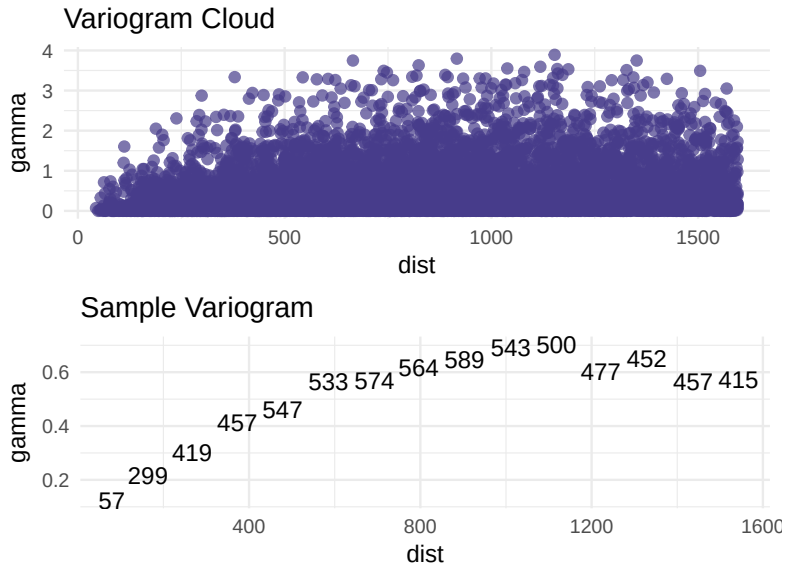



Figure 2.10: Variogram cloud (top) and sample variogram (bottom) for log-zinc data.

likelihood and draw conclusions based on a summary of knowledge. Gaussian process offers a bayesian machine learning framework under which we can make estimations, predictions, inference and decision making in an analytically tractable and computationally viable way in a geospatial setting.

*"Do not become an artist; be a Bayesian, much more scope for the imagination!"*

Unknown

# 3

# Gaussian Process to the rescue

## 3.1 Gaussian Process's advocate

Over the last decade there has been an explosion of work in the "kernel machines" area of machine learning. Probably the best known example of this is work on support vector machines, but during this period there has also been much activity concerning the application of Gaussian process models to machine learning tasks. Gaussian processes provide a principled, practical, probabilistic approach to learning in kernel machines. Theoretical and practical developments of over the last decade have made Gaussian processes a serious competitor for real supervised learning applications.

Roughly speaking a *stochastic process* is a generalization of a probability distribution (which describes a finite-dimensional random variable) to functions. By focusing on processes which are Gaussian, it turns out that the computations required for inference and learning become relatively easy. Thus, the supervised learning problems in machine learning which can be though of as learning a function from examples can be cast directly into the Gaussian process framework.

Historically, late nineties was a time when the field of neural networks was becoming mature and the many connections to statistical physics, probabilistic models and statistics became well known, and the first kernel-based learning algorithms were becoming popular. In retrospect, it is clear that

the time was ripe for the application of Gaussian processes to machine learning problems. Many researchers were realizing that neural networks were not so easy to apply in practice, due to the many decisions which needed to be made: what architecture, what activation functions, what learning rate, etc., and the lack of a principled framework to answer these questions. Of course approximations through the probabilistic framework were tried by MacKay [1992], as well as markov chain monte carlo methods by Neal[1996]. Furthermore, Neal sought in his thesis to demonstrate that using Bayesian formalism, one does not necessarily have problems with "overfitting" when the models get large and one should pursue the limit of large models. While his own work was focused on sophisticated Markov chain methods for inference in large finite networks, he did point out that **some of his networks became Gaussian processes** in the limit of infinite size and there may be simpler ways to do inference in this case.

It is also interesting to take a slightly wider historical perspective. The main reason why neural networks became popular was that they allowed the use of adaptive basis functions, as opposed to the well known linear models. The adaptive basis functions, or hidden units, could "learn" hidden features useful for the modelling problem at hand. However, this adaptivity came at the cost of a lot of practical problems. Later, with the advancement of the "kernel" era, it was realized that the limitation of fixed basis functions is not a big restriction if only one has enough of them (infinitely many) and one is careful to control problems of overfitting by using priors or regularization. The resulting models are much easier to handle than the adaptive basis function models, but have similar expressive power.

Supervised learning problems have been studied for more than a century in statistics and a large body of well-established theory has been developed. More recently, with the advance of affordable, fast computation, the machine learning community has addressed increasingly large and complex problems. Much of the basic theory and many algorithms are shared between the statistics and machine learning community. The primary differences are perhaps the types of the problems attacked and the goal of learning. At the risk of oversimplification, one could say that in statistics a prime focus is often in understanding the data and relationships in terms of models giving approximate summaries such as linear relations or independencies. In contrast, the goals in machine learning are primarily to make predictions as accurately as possible and to understand the behaviour of learning algorithms. These differing objectives have led to different developments in the two fields: for example, neural network al-

gorithms have been used extensively as black-box function approximators in machine learning, but to many statisticians they are less satisfactory because of the difficulties in interpreting such models.

Gaussian process models in some sense bridge the gap between the two communities. As we will see, Gaussian processes are mathematically equivalent to many well known models, including Bayesian linear models, spline models, large neural networks under suitable conditions and are closely related to others, such as support vector machines. Under the Gaussian process viewpoint, the models may be easier to handle and interpret than their conventional counterparts, such as neural networks. In the statistics community Gaussian processes have also been discussed many times, although it would probably be excessive to claim that their use is widespread except for certain specific applications such as spatial models and time series.

## 3.2 A swift introduction in Gaussian Process Regression

The simple linear regression model where the output is a linear combination of the inputs has been studied and used extensively. Its main virtues are simplicity of implementation and interpretability. Its drawbacks of course stem from the assumptions it makes. We have already discussed the assumption of independency and homoscedasticity of errors, that do not seem to hold in a geospatial problem. The most serious drawback though is that it only allows a limited flexibility; if the relationship between input and output cannot reasonably be approximated by a linear function, the model will give poor predictions.

Gaussian process is a framework that can surmount these types of problems in a fairly sophisticated way. Now, there are two ways to interpret Gaussian process (GP) regression models. One can think of a Gaussian process as defining a distribution over functions, and inference taking place directly in the space of functions, the **function-space view**. Although this view is appealing, it can be tricky to grasp in the beginning. Therefore we will start with the equivalent **weight-space view** which may be more familiar and accessible. And the perfect place to start for this is the bayesian standard linear regression model.

### 3.2.1 Bayesian linear regression and weight-space view

Let's assume the simple setting where

$$y = f(\boldsymbol{x}) + \boldsymbol{\varepsilon}, \quad f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w},$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ is a vector of weights (parameters) of the linear model, f is the function value and y is the observed target value (of course we include a bias term but we do not explicitly include it in our notation). Here we have assumed that the observed values $y$ differ from the function values $f(x)$ by additive noise, and we will further assume that this noise follows an independent, identically distributed Gaussian distribution with zero mean and variance $\sigma_n^2$

$$\boldsymbol{\varepsilon} \sim N(0, \sigma_n^2).$$

Naturally we can continue defining the likelihood, the probability density of the observations given the parameters, which exploits the independence assumption and multiplies the probabilities of observations to give

$$\begin{aligned} p(y|X,w) &= \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\Big(-\frac{(y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2}{2\sigma_n^2}\Big) \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\Big(-\frac{1}{2\sigma_n^2}|\boldsymbol{y} - X^T\boldsymbol{w}|^2\Big) = N(X^T\boldsymbol{w}, \sigma_n^2 I). \end{aligned}$$

In the bayesian perspective and in contrast with classical statistics, probability express our belief about something. In this train of thought, $\boldsymbol{w}$ are parameters under estimation, so we specify a *prior* distribution over them expressing our beliefs before we look at the observations. We put a zero mean Gaussian prior with covariance matrix $\Sigma_p$ on the weights

$$\boldsymbol{w} \sim N(\boldsymbol{0}, \Sigma_p).$$

Now, bayes rule is of course the main tool in bayesian statistics in order to combine and summarize prior information and likelihood using

$$posterior = \frac{likelihood\ x\ prior}{marginal\ likelihood}$$

or alternatively,

$$p(\boldsymbol{w}|\boldsymbol{y}, X) = \frac{p(\boldsymbol{y}|X, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|X)}.$$

The normalizing constant or else marginal likelihood is given by

$$p(\boldsymbol{y}|X) = \int p(\boldsymbol{y}|X,\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}.$$

If we omit the normalizing constant for a bit, we would observe that

$$\begin{aligned} p(\boldsymbol{w}|X,\boldsymbol{y}) &\propto \exp\Big(-\frac{1}{2\sigma_n^2}(\boldsymbol{y}-X^T\boldsymbol{w})^T(\boldsymbol{y}-X^T\boldsymbol{w})\Big))\exp\Big(-\frac{1}{2}\boldsymbol{w}^T\Sigma_p^{-1}\boldsymbol{w}\Big) \\ &\propto \exp\Big(-\frac{1}{2}(\boldsymbol{w}-\bar{\boldsymbol{w}})^T(\frac{1}{\sigma_n^2}XX^T+\Sigma_p^{-1})(\boldsymbol{w}-\bar{\boldsymbol{w}})\Big), \end{aligned}$$

where $\bar{\boldsymbol{w}} = \sigma_n^{-2}(\sigma_n^2XX^T+\Sigma_p^{-1})^{-1}X\boldsymbol{y}$, is of the Gaussian distribution form. So the posterior distribution is also a Gaussian with mean $\bar{\boldsymbol{w}}$ and covariance matrix $A^{-1}$

$$p(\boldsymbol{w}|X,\boldsymbol{y}) \sim N(\bar{\boldsymbol{w}} = \frac{1}{\sigma_n^2}A^{-1}X\boldsymbol{y}, A^{-1}),$$

where $A = \sigma_n^{-2}XX^T+\Sigma_p^{-1}$.

Note that bayesian statistics in theory have a serious computational problem, and that is calculating the normalizing constant we mentioned before. This constant, even in simple models, becomes analytically intractable when we move to high-dimensional spaces. Therefore, we cannot calculate posterior distribution in closed form. While there are methodologies (e.g. MCMC) to dribble this problem in its general form, there is a tiny family of models where we are able to compute posterior distribution in its closed form and this is called the conjugate priors family. In this family, the product of prior distribution and likelihood creates a posterior distribution form which we are able to pre-recognize. Beta-binomial, dirichlet-multinomial and Gaussian-Gaussian (like the one we are talking about in this chapter) are famous examples in this category where we truly are able to compute the quantities we need in closed form.

In order to make prediction for a single test case, we simply have to think of $f_*$ again as an extra parameter under estimation. Therefore, we can simply integrate out all the remaining parameters and arrive at the predictive distribution we need. In other terms, predictive distribution $f_* \triangleq f(\boldsymbol{x}_*)$ at $\boldsymbol{x}_*$ is given by averaging the output of all possible linear models w.r.t the Gaussian posterior

$$\begin{aligned} p(f_*|\boldsymbol{x}_*,X,\boldsymbol{y}) &= \int p(f_*|\boldsymbol{x}_*,\boldsymbol{w})p(\boldsymbol{w}|X,\boldsymbol{y})d\boldsymbol{w} \\ &= N\Big(\frac{1}{\sigma_n^2}\boldsymbol{x}_*^TA^{-1}X\boldsymbol{y}, \boldsymbol{x}_*^TA^{-1}\boldsymbol{x}_*\Big). \end{aligned}$$

The predictive distribution is again Gaussian, with a mean given by the posterior mean of the weights multiplied by the test input. The predictive variance is a quadratic form of the test input with the posterior covariance matrix. Again, we are still able to compute the closed form of the predictive distribution because of the Gaussian trick we mentioned before. However, the setup discussed so far still has the limitations of a simple linear model and its still unable to discover non-linear relationships between response variable and predictors.

### Projections of Inputs into Feature Space

An efficient way to overcome the aforementioned problem is to project the inputs into some high dimensional space using a set of basis functions. A scalar input $x$ could be projected for example into the space of powers of $x : \phi(x) = (1, x^2, x^3, ...)^T$ to implement polynomial regression. As long as the projections are fixed functions (i.e independent of the parameters $\boldsymbol{w}$) the model is still linear in the parameters and therefore analytically tractable. For now let's assume that the basis functions are given, though gaussian process formalism offers a way to choose them.

Using function $\phi(x)$ which maps a D-dimensional input vector $\boldsymbol{x}$ into an N dimensional feature space, we now model

$$f(\boldsymbol{x}) = \phi(\boldsymbol{x})^T \boldsymbol{w},$$

where the vector of parameters now has length N. The distributions stay the same with only change being the substitution of X with $\Phi(X)$. The predictive distribution becomes

$$f_*|\boldsymbol{x}_*, X, \boldsymbol{y} \sim N\Big(\frac{1}{\sigma_n^2}\phi(\boldsymbol{x}_*)^T A^{-1}\Phi\boldsymbol{y}, \phi(\boldsymbol{x}_*)^T A^{-1}\phi(\boldsymbol{x}_*)\Big),$$

where $\Phi = \Phi(X)$ and $A = \sigma_n^{-2}\Phi\Phi^T + \Sigma_p^{-1}$. Of course the difficult part in the above equation is to invert matrix $A$, where the dimension of the feature space is usually large. However, we can rewrite the equation in the following way

$$\begin{aligned} f_*|\boldsymbol{x}_*, X, \boldsymbol{y} \sim N(&\phi_*^T\Sigma_p\Phi(K + \sigma_n^2 I)^{-1}\boldsymbol{y}, \\ &\phi_*^T\Sigma_p\phi_* - \phi\Sigma_p\Phi(K + \sigma_n^2)^{-1}\Phi^T\Sigma_p\phi_*), \end{aligned}$$

where $\phi(\boldsymbol{x}_*) = \phi_*$ and $K = \Phi^T\Sigma_P\Phi$. In this form, we need to invert matrices of usually smaller dimension.

Notice in the above equation that features space always enters in the form of $\Phi^T\Sigma_p\Phi$, $\phi_*\Sigma_p\Phi$, or $\phi_*^T\Sigma_p\phi_*$, thus the entries of these matrices are invariably of the form $\phi(\boldsymbol{x})^T\Sigma_p\phi(\boldsymbol{x}')$ where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are in either the training or the test sets. Let us define $k(\boldsymbol{x},\boldsymbol{x}') = \phi(\boldsymbol{x})^T\Sigma_p\phi(\boldsymbol{x}')$ and call $k(\cdot,\cdot)$ a **covariance function** or **kernel**. Notice that $\phi(\boldsymbol{x})^T\Sigma_p\phi(\boldsymbol{x}')$ is also an inner product with respect to $\Sigma_p$. As $\Sigma_p$ is positive definite we can define $\Sigma_p^{1/2}$ so that $(\Sigma_p^{1/2})^2 = \Sigma^2$; for example if the SVD of $\Sigma_p = UDU^T$, where D is diagonal, then one form for $\Sigma_p^{1/2}$ is $UD^{1/2}U^T$. Then defining $\psi(\boldsymbol{x}) = \Sigma_p^{1/2}\phi(\boldsymbol{x})$ we obtain a simple dot product representation $k(\boldsymbol{x},\boldsymbol{x}') = \psi(\boldsymbol{x})\cdot\psi(\boldsymbol{x}')$.

In this setup, the algorithm is solely defined in terms of inner products in input space. Because of that, it can be lifted into feature space by replacing occurences of those inner products by $k(\boldsymbol{x},\boldsymbol{x}')$; this is sometimes called the **kernel trick**. This technique is particularly valuable in situations where it is more convenient to compute the kernel than the feature vectors themselves. Thus, we end up considering the kernel as the object of primary interest and the corresponding feature space as having secondary practical importance.

### 3.2.2 Function-space view

The alternative and equivalent way to approach a gaussian process is by considering inference directly in function space. Now we interpret the very same $f$ function as a random variable and we use a gaussian process to describe a distribution over functions.

**Definition 3.1.** A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Now, a Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(\boldsymbol{x})$ and the covariance function $k(\boldsymbol{x},\boldsymbol{x}')$ of a real process $f(\boldsymbol{x})$ as

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}[f(\boldsymbol{x})], \\ k(\boldsymbol{x},\boldsymbol{x}') &= \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))], \end{aligned}$$

and we will write the Gaussian process as

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

In our case the random variables represent the values of the function $f(\boldsymbol{x})$ at location $\boldsymbol{x}$. Often, Gaussian processes are defined over time or over

space, however this is not necessarily the case ; the index set $\mathcal{X}$ is the set of possible inputs and it can surely be $\mathbb{R}^d$.

The definition of the gaussian process as a collection of random variables induces a consistency requirement and gaussian distribution serves this purpose naturally. If we assume that a $\mathcal{GP}$ specifies $(y_1, y_2) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, then it must also specify $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ and $y_2 \sim \mathcal{N}(\mu_2, \Sigma_{22})$, where $\Sigma_{11}, \Sigma_{22}$ are the relevant submatrices of $\Sigma$. In other words, examination and analysis of larger sets of a larger set of variables does not change the distribution of the smaller set. The consistency requirement is automatically fulfilled if the covariance function specifies entries of the covariance matrix. The definition does not exclude Gaussian processes with finite index sets (which would simply be Gaussian distributions), though they are not so interesting in our purposes.

A simple example of Gaussian process can be attained by the Bayesian linear regression model $f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$ with prior $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$. The mean and the covariance would be

$$\begin{aligned} \mathbb{E}[f(\boldsymbol{x})] &= \boldsymbol{\phi}(\boldsymbol{x})^T \mathbb{E}[\boldsymbol{w}] = 0, \\ \mathbb{E}[f(\boldsymbol{x}) f(\boldsymbol{x}')] &= \boldsymbol{\phi}(\boldsymbol{x})^T \mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T] \boldsymbol{\phi}(\boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^T \Sigma_p \boldsymbol{\phi}(\boldsymbol{x}'). \end{aligned}$$

Thus $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ are jointly Gaussian with zero mean and covariance given by $\boldsymbol{\phi}(\boldsymbol{x})^T \Sigma_p \boldsymbol{\phi}(\boldsymbol{x}')$.

Covariance functions, being the most interesting part in a Gaussian process, will be further discussed later on. For now, we will use **squared exponential** or radial basis function or Gaussian kernel

$$cov(f(\boldsymbol{x}_p), f(\boldsymbol{x}_q)) = k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \exp(-\frac{1}{2}|\boldsymbol{x}_p - \boldsymbol{x}_q|^2).$$

Notice that the covariance function specifies the covariance between any pair of random variables and that the covariance between outputs is written as a function of the inputs. For this particular function, we can see that the covariance is almost unity between variables whose corresponding inputs are very close and decreases as their distance in the input space increases.

It can also be shown that the squared exponential covariance function corresponds to a Bayesian linear regression model with an infinite number of basis functions. Indeed for every positive definite covariance function $k(\cdot, \cdot)$, there exists a (possibly infinite) expansion in terms of basis functions, a property we will discuss more extensively later on. We can also

obtain the SE covariance function from the linear combination of an infinite number of Gaussian-shaped basis functions.

To make our demonstration more digestible, let's assume a non linear function of unidimensional input $f(x) = x\sin(x)$ over a continuous space (Figure 3.1) and say we want to "recover" this relationship using a Gaussian process. So we start by defining that

$$f(x) \sim \mathcal{GP}(m(x), K(x, x')),$$

where squared exponential is the covariance function.

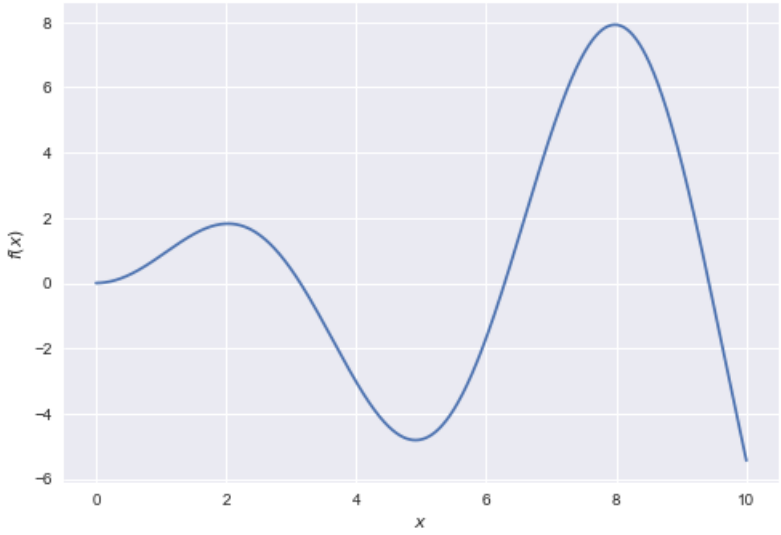



Figure 3.1: An example of non linear relationship between x and y: $x \cdot sin(x)$.

The specification of the covariance function implies a distribution over functions. To see this, we can draw samples from the distribution of functions evaluated at any number of points. Here we choose a number of input points, $X_*$ and we write out the covariance matrix for the squared exponential kernel elementwise. Technically, these input points play the role of test inputs and therefore carry a subscript asterisk; this part will clear itself out later on when both train and test points are involved. So we end up generating a Gaussian vector with the computed covariance matrix

$$\boldsymbol{f}_* \sim \mathcal{N}(\mathbf{0}, K(X_*, X_*))$$

and plot the generates values as a function of the inputs (Figure 3.2).

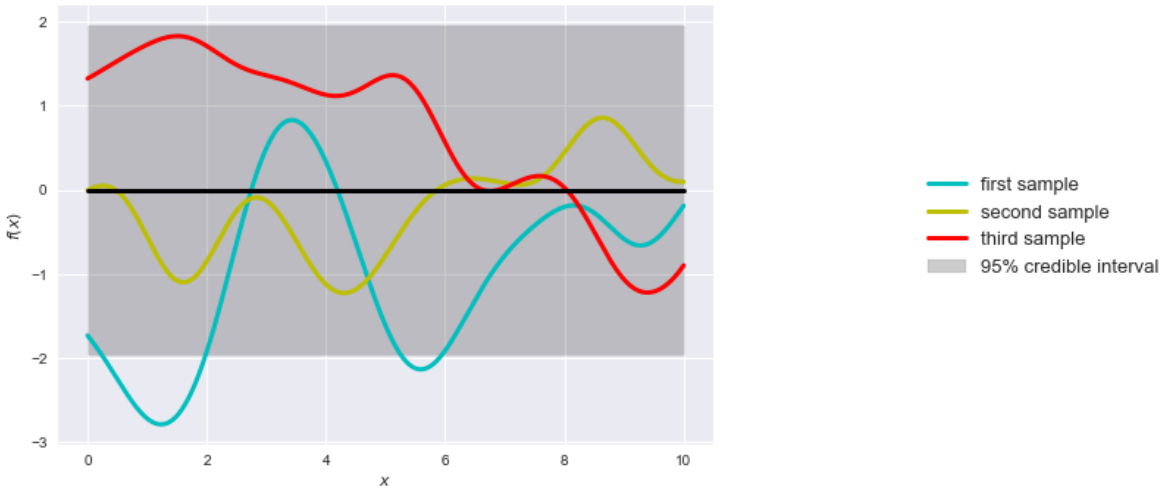



Figure 3.2: Three sample functions drawn from prior distribution along with their mean (0) and their point estimated standard deviation.

Notice that the functions look smooth. This is due to the fact that squared exponential kernel being infinitely mean-squared differentiable. We also see that the functions seem to have a characteristic length-scale, which informally can be thought of as roughly the distance you have to move in input space before the function value can change significantly. In figure 3.2 length-scale is equal to 1 unit. By replacing $|\boldsymbol{x}_p - \boldsymbol{x}_q|$ with $|\boldsymbol{x}_p - \boldsymbol{x}_q|/l$ in the squared exponential kernel we can change the characteristic length-scale of the process. Also, the overall variance of the random function can be controlled by a positive pre-factor before the exponential part of the squared exponential kernel.

### Prediction with Noise-free observations

Although we drew some random functions from the prior in order to witness the process from close enough, this is not usually interesting. What we want is to incorporate knowledge the training data provides about the function. Initially we will showcase the simple special case where the observations are noise free

$$\{(\boldsymbol{x}_i, f_i)|i = 1, ..., n\}.$$

The joint distribution of the training outputs, $\boldsymbol{f}$, and the test outputs $\boldsymbol{f}_*$ according to the prior is

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\Big(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\Big).$$

If we have $n$ training points and $n_*$ test points, then $K(X,X_*)$ denotes the $n \quad x \quad n_*$ matrix of the covariances evaluated at all pairs of training and test points. To get the posterior distribution over functions we need to restrict this joint prior distribution to contain only those functions which agree with the observed data points. In other words, we just have to condition the joint Gaussian prior distribution on the observation to give

$$\begin{aligned} \boldsymbol{f}_*|X_*,X,\boldsymbol{f} \sim \mathcal{N}(&K(X_*,X)K(X,X)^{-1}\boldsymbol{f}, \\ &K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)) \end{aligned}$$

In figure 3.3 we showcase three samples of the posterior predictive distribution, a results that stems from the six observations marked by blue dots. We have also painted with black the maximum aposteriori function and the corresponding 95% credible intervals along the input line. Notice how the interval of our predictions is larger in the areas where we have not observed points. It is also trivial to extend these computations to multidimensional input spaces (we only have to change the evaluation of the covariance function), although the resulting function will be harder to display graphically.

### Prediction with Noisy observations

In more realistic situations, we do not have access to function values themselves, but only noisy versions of the form $y = f(\boldsymbol{x}) + \varepsilon$. Assuming additive independent identically distribution Gaussian noise $\varepsilon$ with variance $\sigma_n^2$, the prior on the noisy observations becomes

$$cov(y_p,y_q) = k(\boldsymbol{x}_p,\boldsymbol{x}_q) + \sigma_n^2\delta_{pq} \; or \quad cov(\boldsymbol{y}) = K(X,X) + \sigma_n^2 I,$$

where $\delta_{pq}$ is a Kronecker delta. It follows from the independence assumption about the noise, that a diagonal matrix is added in comparison to the noise free case. Now the joint distribution of the observed with noise terms and the function values at the test locations under the prior is

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\Big(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\Big).$$

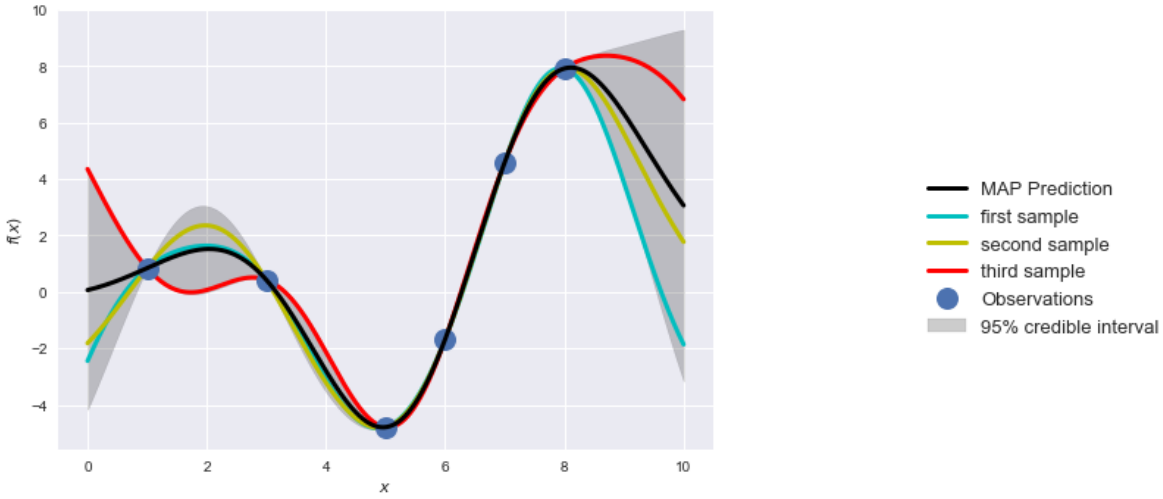



Figure 3.3: Predictions from Posterior predictive distribution, sample functions and their credible interval under noise free assumption.

In this case, the predictive equations for Gaussian process regression are

$$\begin{aligned} \boldsymbol{f}_*|X,\boldsymbol{y},X_* &\sim \mathcal{N}(\bar{\boldsymbol{f}}_*, cov(\boldsymbol{f}_*)), \quad \textit{where} \\ \bar{\boldsymbol{f}}_* &\triangleq \mathbb{E}[\boldsymbol{f}_*|X,\boldsymbol{y},X_*] = K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}\boldsymbol{y}, \\ cov(\boldsymbol{f}_*) &= K(X_*,K_*) - K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X*). \end{aligned}$$

Now we have exact correspondence with the weight space view if we identify $K(C,D) = \Phi(C)^T\Sigma_p\Phi(D)$, where C,D stand for either $X$ or $X_*$. For any set of basis function we can compute the corresponding covariance function as $k(\boldsymbol{x}_p,\boldsymbol{x}_q) = \phi(\boldsymbol{x}_P)^T\Sigma_p\phi(\boldsymbol{x}_q)$; conversely, for every positive definite covariance function k, there exists a (possibly) infinite expansion in terms of basis functions.

In order to make our notation more compact, from now on we will refer to $K(X,X)$ as $K$ and to $K(X,X_*)$ as $K_*$. In the case that there is only one test point $\boldsymbol{x}_*$, we write $\boldsymbol{k}(\boldsymbol{x}_*) = \boldsymbol{k}_*$ to denote the vector of covariances between the test point and the n training points. Using the compact notation and for a single test point $\boldsymbol{x}_*$, we reduce the predictive distribution equations to

$$\begin{aligned} \bar{f}_* &= \boldsymbol{k}_*^T(K+\sigma_n^2 I)^{-1}\boldsymbol{y} \\ \mathbb{V}[f_*] &= k(\boldsymbol{x}_*,\boldsymbol{x}_*) - \boldsymbol{k}_*^T(K+\sigma_n^2 I)^{-1}\boldsymbol{k}_*. \end{aligned}$$

If we delve further inside the above equations we can see at first that the mean prediction is a linear combination of observations $\boldsymbol{y}$; this is sometimes referred to as a linear predictor. Another way to look at this equation is to see it as a linear combination of n kernel functions, each one centered on a training point, by writing

$$\bar{f}(\boldsymbol{x}_*) = \sum_{i=1}^{n} a_i k(\boldsymbol{x}_i, \boldsymbol{x}_*)$$

where $\boldsymbol{a} = (K + \sigma_n^2 I)^{-1}\boldsymbol{y}$. The fact that the mean prediction for $f(\boldsymbol{x}_*)$ can be written as such, despite the fact that the GP can be represented in terms of a (possibly infinite) number of basis functions is one manifestation of the *representer theorem* we will again discuss later on.

We can understand this result intuitively because although the GP defines a joint Gaussian distribution over all of the y variables, one for each point in the index sex $X$, for making predictions at $\boldsymbol{x}_*$ we only care about the (n+1)-dimensional distribution defined by the n training points and the test point. As a Gaussian distribution is marginalized by just taking the relevant block of the joint covariance matrix, it is clear that conditioning this (n+1)-dimensional distribution on the observations gives us the desired result.

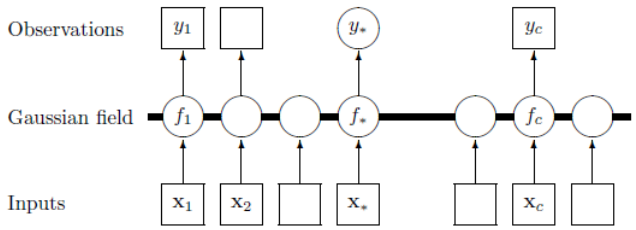



Figure 3.4: Chain graph for a GP regression. Squares represent observed variables and circles represent unknowns. The thick horizontal bar represents a set of fully connected nodes. Note that an observation $y_i$ is conditionally independent of all other nodes given the corresponding latent variable, $f_i$. Because of the marginalization property of GP, addition of further inputs, $\boldsymbol{x}$, latent variables, f, and unobserved targets, $y_*$, do not change the distribution of any other variables.

In the practical scenario of before, we can use the kernel for the noisy observations and compute the posterior distribution. We can again sample some function from this posterior distribution, we can pick the mean function and even create again the credible intervals (Figure 3.5). Notice also

in Figure 3.6, how we would have made inference if we assumed that there is no noise in our noisy observations. The noise-free case (blue line) will interpolate the points as it should do in the noise-absence case, while the noisy kernel will try to approximate the signal function.

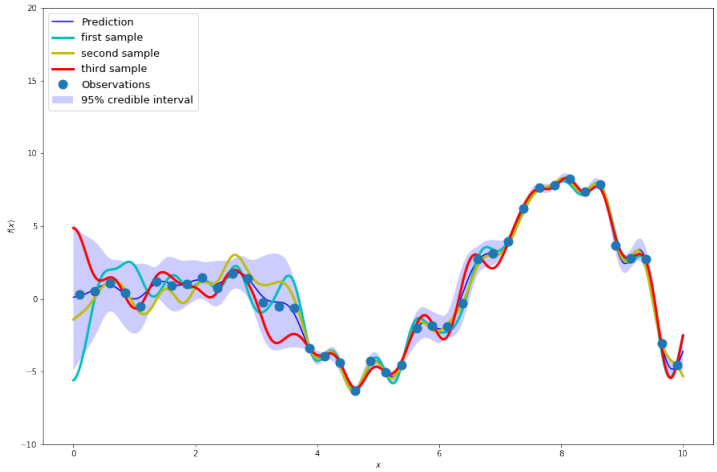



Figure 3.5: Predictions from Posterior predictive distribution, sample functions and their credible interval under noise assumption.

Note that the variance in the aforementioned predictive equation does not depend on the observed targets, but only on the inputs; a Gaussian distribution property. The variance is the difference between two terms: the first term $K(X_*, X_*)$ is simply the prior covariance; from that is subtracted a (positive) term, representing the information the observations gives us about the function. We can very simply compute the predictive distribution of test targets $\boldsymbol{y}_*$ by adding $\sigma_n^2 I$ to the variance in the expression for $cov(\boldsymbol{f}_*)$.

A practical implementation of Gaussian process regression in the noise-free case is shown in figure 3.7 (we just have to sum the noise variance for the noisy case). The algorithm uses Cholesky decomposition, instead of directly inverting the matrix, since it is faster, numerically more stable and the matrix is symmetric and positive semidefinite.

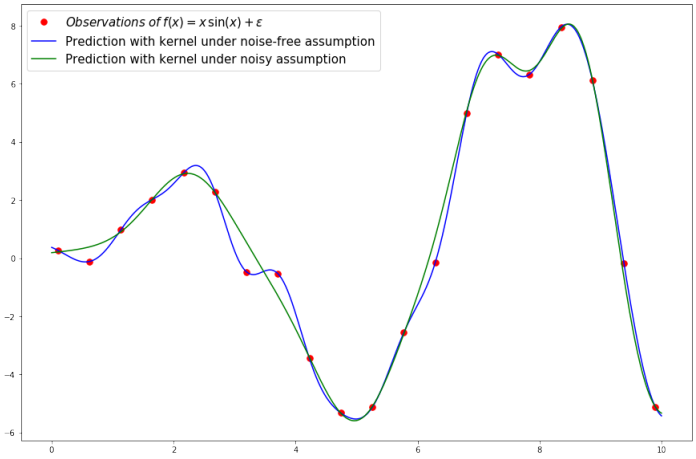



Figure 3.6: Visualization of the behaviour of GP under noise-free as well as noisy assumptions for the noisy data.

## 3.3 A little something about covariance functions

We have already seen that a covariance function is the crucial ingredient in a Gaussian process predictor, as it encodes our assumptions about the function which we wish to learn. From a slightly different point of view it is clear in supervised learning, that the notion of similarity between data points is crucial; it is a basic assumption that points with inputs $\boldsymbol{x}$ which are close are likely to have similar target values $\boldsymbol{y}$, and thus training points that are near to a test point should be informative about the prediction at that point. Under the Gaussian process view it is the covariance function that defines nearness or similarity.

A $\boldsymbol{stationary}$ covariance function is a function of $\boldsymbol{x} - \boldsymbol{x'}$. This term comes from the theory of stochastic processes, where a process which has constant mean and whose covariance function is invariant to translations is called weakly stationary. A process is strictly stationary if all of its finite dimensional distributions are invariant to translations. Therefore a stationary covariance is invariant to translations. For example the squared exponential covariance function is stationary. If further the covariance function is a function only of $|\boldsymbol{x} - \boldsymbol{x'}|$ it is called isotropic; it is thus invariant to all rigid motions. Squared exponential is isotropic too. As k is now only a function $\boldsymbol{r} = |\boldsymbol{x} - \boldsymbol{x'}|$ these are also called radial basis functions.

If a covariance function depends only on $\boldsymbol{x}$ and $\boldsymbol{x'}$ through $\boldsymbol{xx'}$ we call

**input**: $X$ (inputs), $\mathbf{y}$ (targets), $k$ (covariance function), $\sigma_n^2$ (noise level), $\mathbf{x}_*$ (test input)

2: $L := \text{cholesky}(K + \sigma_n^2 I)$

$\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$

4: $\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$ } predictive mean eq.

$\mathbf{v} := L \backslash \mathbf{k}_*$

6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$ } predictive variance eq.

$\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^\top \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2}\log 2\pi$ eq.

8: **return**: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)

Figure 3.7: Steps of the algorithm for GPR. For multiple test cases lines 4-6 are repeated. The computational complexity is $\frac{n^3}{6}$ for the Cholesky decomposition in line 2, and $\frac{n^2}{2}$ for solving triangular systems in line 3 and line 5.

it a dot product covariance function. A simple example is the covariance function $k(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_0^2 + \boldsymbol{x}\boldsymbol{x'}$ which can be obtained from linear regression by putting $\mathcal{N}(0,1)$ priors on the coefficients of $x_d (d = 1,...,D)$ and a prior of $\mathcal{N}(0,\sigma_0^2)$ on the bias. Another important example is the inhomogeneous polynomial kernel $k(\boldsymbol{x}, \boldsymbol{x'}) = (\sigma_0^2 + \boldsymbol{x}\boldsymbol{x'})^p$ where p is a positive integer. Dot product covariance functions are invariant to a rotation of the coordinates about the origin, but not translations.

A general name for a function k of two arguments mapping a pair of inputs $\boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{x'} \in \mathcal{X}$ into $\mathbb{R}$ is a kernel. This term arises in the theory of integral operators, where the operator $T_k$ is defined as

$$(T_k f)(\boldsymbol{x}) = \int_{\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{x'}) f(\boldsymbol{x'}) d\mu(\boldsymbol{x'}),$$

where $\mu$ denotes a measure. Clearly covariance functions must be symmetric from the definition of symmetry. Given a set of input points $\{\boldsymbol{x}_i | i = 1,...,n\}$ we can compute the Gram matrix K whose entries are $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. If k is a covariance function we call the matrix K the covariance matrix.

A real $n \times n$ matrix K which satisfies $Q(\boldsymbol{v}) = \boldsymbol{v}^T K \boldsymbol{v} \geq 0$ for all vectors $\boldsymbol{v} \in \mathbb{R}^n$ is called positive semidefinite. A Gram matrix is PSD if and only if all of its eigenvalues are non-negative. A Gram matrix corresponding to a general kernel function need not to be PSD, but the Gram matrix corresponding to a covariance function is PSD.

A kernel on the other hand is said to be positive semidefinite if

$$\int k(\boldsymbol{x}, \boldsymbol{x'}) f(\boldsymbol{x}) f(\boldsymbol{x'}) d\mu(\boldsymbol{x}) d\mu(\boldsymbol{x'}) \geq 0,$$

for all $f \in L_2(\mathcal{X}, \mu)$. Equivalently a kernel function which gives rise to PSD Gram matrices for any choice of $n \in N$ and $\mathcal{D}$ is positive definite. So a kernel function that satisfies the above property is a valid covariance function.

While there is a rich theory concerning the properties of valid covariance functions, the way we can prove that a kernel is a valid one, their eigenvalue-eigenvector viewpoint and the convenience of using a stationary covariance function, these parts live outside the scope of this thesis. We are only going to showcase some common ones and prove that we are able to produce new covariance functions by additive and multiplicative combinations of others, already valid ones.

### Squared exponential covariance function

The squared exponential (SE) covariance function has already been introduced and has the form

$$k_{SE}(r) = \exp\Big(-\frac{r^2}{2l^2}\Big),$$

with only parameter being characteristic length-scale l. This covariance function is infinitely differentiable, which means that the GP with this covariance function has mean squared derivatives of all orders, and is thus very smooth. Stein [1999] argues that such strong smoothness assumptions are unrealistic for modelling many physical processes and recommends the Matern class. However, the squared exponential is probably the most widely-used kernel within the kernel machines field. SE also has two extra properties that offer more space for experimentation on the best kernel. It is also infinite divisible in that $(k(r))^t$ is a valid kernel for all $t \geq 0$ and can be obtained by expanding the input $\boldsymbol{x}$ into a feature space defined by Gaussian-shaped basis functions centered densely in $\boldsymbol{x}$-space.

### Matern covariance function

The Matern class of covariance functions is given by

$$k_{Matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu}r}{l}\Big)^{\nu} K_{\nu}\Big(\frac{\sqrt{2\nu}r}{l}\Big),$$

with positive parameters $\nu$ and l, where $K_{\nu}$ is a modified Bessel function. The scaling is chosen so that for $\nu \to \infty$ we obtain the SE covariance function. The Matern covariance functions become especially simple when $\nu$ is $1/2$, indicating the once differentiable functions. Applications in the field of

machine learning also show that $\nu = 3/2$ and $\nu = 5/2$ are interesting cases, since once differentiable functions make the process very rough.

#### Rational Quadratic covariance function

The rational quadratic (RQ) covariance function

$$k_{RQ}(r) = \left(1 + \frac{r^2}{2al^2}\right)^{-a}$$

with $a, l > 0$ can be seen as a scale mixture (an infinite sum) of squared exponential (SE) covariance functions with different characteristic length-scales (sums of covariance functions are also a valid covariance, as we will prove later on).

#### Sum of covariance functions

The sum of two kernels is a kernel. Proof: consider the random process $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$, where $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are independent. Then $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}')$. This construction can be used e.g. to add together kernels with different characteristic length-scales (Rational Quadratic).

#### Product of covariance functions

The product of two kernels is a kernel. Proof: consider the random process $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) f_2(\boldsymbol{x})$, where $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are independent. Then $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') k_2(\boldsymbol{x}, \boldsymbol{x}')$. A simple extension of this argument means that $k^p(\boldsymbol{x}, \boldsymbol{x}')$ is a valid covariance function for $p \in \mathbb{N}$.

## 3.4 Model selection and adaptation of hyperparameters

Up until now, we have worked with given fixed covariance function and we have sketched the behaviour of some of the most common kernels. However, in many practical applications, it may not be easy to specify all aspects of the covariance function with confidence, let alone decide between them for the most suitable one. While some properties such as stationarity of the covariance function may be easy to determine from the context, we typically have rather vague information about other properties and values of

the hyper-parameters. In order to turn Gaussian processes into powerful practical tools it is essential to develop methods that address the model selection problem. We interpret the model selection problem rather broadly, including the discrete choice of the functional form for the covariance function, as well as the values of any hyperparameters. In fact the model selection can help both to refine the predictions of the model and give a valuable interpretation to the user about the properties of the data, e.g. that a non-stationary covariance function may be preferred over a stationary one.

### The model selection problem

A multitude of possible families of covariance functions exists, including squared exponential, polynomial, neural network, matern etc. We will refer to the selection of a covariance function and its parameters as *training* of a Gaussian process.

Let's start again with the well understood squared exponential kernel. This kernel is one of those that can be parameterized in terms of hyperparameters

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp(-\frac{1}{2}(\boldsymbol{x}_p - \boldsymbol{x}_q)^T M(\boldsymbol{x}_p - \boldsymbol{x}_q)) + \sigma_n^2 \delta_{pq},$$

where $\boldsymbol{\theta} = (\{M\}, \sigma_f^2, \sigma_n^2)^T$ is a vector containing all the hyperparameters and $\{M\}$ denotes the parameters in the symmetric matrix $M$. Possible choices for the matrix M include

$$M_1 = l^{-1}I, \quad M_2 = diag(l)^{-2}, \quad M_3 = \Lambda\Lambda^T + diag(l)^{-2},$$

where l is a vector of positive values and $\Lambda$ is a D x k matrix with $k < D$. The properties of functions with these covariance functions depend on values of the hyperparameters. For many covariance functions it is easy to interpret the meaning of the hyperparameters, which is of great importance when trying to understand your data. For the squared exponential covariance function with distance measure $M_2$, the $l_1, ..., l_D$ hyperparameters play the role of characteristic length-scales, indicating how far do you need to move in input space for the function values to become uncorrelated. Such a covariance function implements automatic relevance determination (ARD, Neal 1996), since the inverse of the length-scale determines how relevant an input is: if the length-scale has a very large value, covariance will become almost independent of that input, effectively removing it from the inference. We call the parameterization of $M_3$, the factor analysis distance due to the analogy with the unsupervised factor analysis model which seeks

to explain the data through a low rank plus diagonal decomposition. For high dimensional datasets the k columns of the $\Lambda$ matrix could identify a few directions in the input space with specially high relevance.

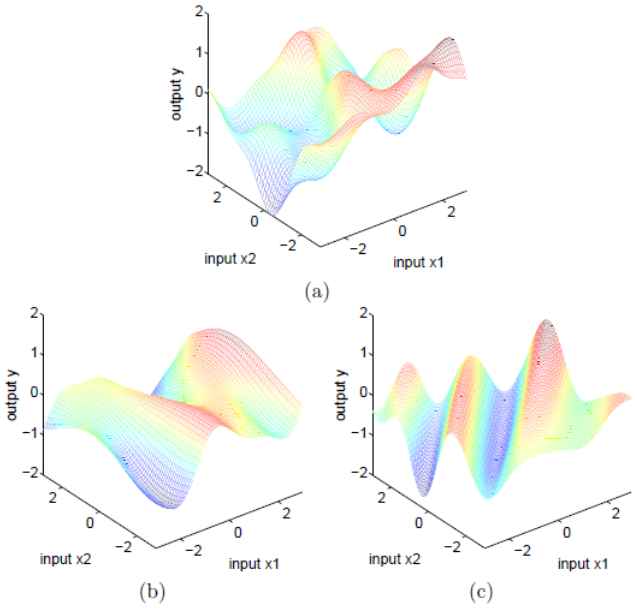



Figure 3.8: Functions with two dimensional input drawn at random from noise free squared exponential covariance function Gaussian processes, corresponding to the three different distance measures respectively. The parameters were: (a) $l = 1$, (b) $l = (1,3)^T$, and (c) $\Lambda = (1,-1)^T$, $l = (6,6)^T$. In panel (a) the two inputs are equally important, while in (b) the function varies less rapidly as a function $x_2$ than $x_1$. In (c) the $\Lambda$ column gives the direction of most rapid variation.

In figure 3.8, three different instances of the aforementioned factor analysis distance are showcased. In all of them, we can see function drawn from squared exponential covariance function GP for difference kind of M matrices. In panel (a) we get an isotropic behaviour- a general distance related length-scale. In panel (b), the characteristic length-scale is different along the two input axes; function varies rapidly as a function of $x_1$ but far slower as a function of $x_2$. In the third panel, the rapid variation is demonstrated along the (1,-1) direction. As we can see, we can end up with very different cases of variations even inside a single family of covariance functions. Our goal is, given a set of training data, to make inferences about the form and parameters of the covariance function.

The above realization that even for squared exponential kernel, there is a huge variety of possible distance measures should not be a cause of despair, rather seen as a possibility to learn. It requires, however, a systematic approach to model selection. Although there are endless variations in the suggestion for model selection in the literature, three principles cover most: (1) compute the probability of the model given the data, (2) estimate the generalization error and (3) bound the generalization error. In this thesis, we are going to make an approach for the first two principles. Bayesian view on model selection will involve the computation of the probability of the model given the data and cross-validation will estimate the generalization performance.

**Bayesian model selection**

It is very common to use a hierarchical specification of models when we take the bayesian scope of model selection. At the lowest level are the parameters $\boldsymbol{w}$, The parameters here could be the parameters of a linear model or even the weights in a neural network model. At the second level we have the hyperparameters $\boldsymbol{\theta}$ which control the distribution of the parameters at the bottom level. For example the "weight decay" term in a neural network or the "ridge" term in ridge regression are hyperparameters. At the top level we usually have a discrete set of possible model structures, $\mathcal{H}_i$, under consideration.

We are going to give a mechanistic description of the computations needed for bayesian inference at first until we move on applying these on a GP regression scheme. Inference takes place one level at a time, using the theory of pure bayesian statistics. At the bottom level, the posterior over the parameters is given by bayes' rule

$$p(\boldsymbol{w}|\boldsymbol{y},X,\boldsymbol{\theta},\mathcal{H}_i) = \frac{p(\boldsymbol{y}|X,\boldsymbol{w},\mathcal{H}_i)p(\boldsymbol{w}|\boldsymbol{\theta},\mathcal{H}_i)}{p(\boldsymbol{y}|X,\boldsymbol{\theta},\mathcal{H}_i)}.$$

The prior distribution here encodes as a probability distribution our knowledge about the parameters prior to seeing the data. We usually have vague prior information about the parameters, so we choose a rather vague prior distribution to reflect this. The normalizing constant is independent of the parameters and called **marginal likelihood** or evidence.

At the next level, we analogously express the posterior over the hyperparameters, where the marginal likelihood from the first level plays the role

of the likelihood

$$p(\boldsymbol{\theta}|\boldsymbol{y},X,\mathcal{H}_i) = \frac{p(\boldsymbol{y}|X,\boldsymbol{\theta},\mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\boldsymbol{y}|X,\mathcal{H}_i)}.$$

At the top level we compute

$$p(\mathcal{H}_i|\boldsymbol{y},X) = \frac{p(\boldsymbol{y}|X,\mathcal{H}_i)p(\mathcal{H}_i)}{p(\boldsymbol{y}|X)}$$

Now, the above implementation requires of the computation of multiple integrals. In many real-world settings, the integrals may or may not be analytically tractable. In practice, especially the evaluation of the middle level posterior may be difficult and as an approximation one may shy away from using the hyperparameter posterior and instead maximize the marginal likelihood with respect to $\boldsymbol{\theta}$. This approximation is also know as type 2 maximum likelihood (ML-2). It is primarily the marginal likelihood involving the integral over the parameter space which distinguishes the Bayesian scheme of inference from other schemes based on optimization.

**Cross Validation**

The basic idea of cross validation is to split the training set into two disjoint sets, one which is actually used for training and the other, the validation set, which is used to monitor performance. The performance on the validation set is used as a proxy for the generalization error and model selection is carried out using this measure.

The main drawback of the "vanilla" method is that we use only a fraction of the training set to train the model and the whole process is somewhat biased on the selection of validation set. To minimize these problems, CV is almost always used in the k-fold cross-validation setting: the data is split into k disjoint, equally sized subsets; validation is done on a single subset and training is done using the union of the remaining k-1 subsets, the entire procedure being repeated k times, each time with a different subset for validation. Thus, a large fraction of the data can be used for training and all cases appear as validation cases. The price is that k models must be trained instead of one.

Cross-validation can be used with any loss function. Although squared error loss is by far the most common for regression, there is no reason not to allow other loss functions. Under probabilistic models like Gaussian

processes, we can exploit the posterior covariance and incorporate uncertainties in our lost function. In our case, we will use both standardized mean squared error and negative log probability density.

In this thesis we are going to use the marginal likelihood optimization method to find the optimal hyperparameters to describe the data for a given kernel. One of the optimization problems here is that depending on the initial value for the hyperparameters, the gradient-based optimization might also converge to the high-noise solution. It is thus important to repeat the optimization several times for different initializations and keep the one with the best log marginal likelihood. This process is repeated for every cross-validation "split" and the average performance of each kernel on appropriate metrics will be tracked down. Now, since we have a regression problem and in the domain of machine learning we are mainly interested of our prediction performance, for a single data point $y_*$ we measure standardized squared error

$$\frac{(y_* - \bar{f}(\boldsymbol{x_*}))^2}{\mathbb{V}(f_*)}$$

and negative log probability density

$$-\log p(y_*|\mathcal{D}, \boldsymbol{x}_*) = \frac{1}{2} log(2\pi\sigma_*^2) + \frac{(y_* - \bar{f}(\boldsymbol{x_*}))^2}{2\sigma_*^2},$$

where $\sigma_* = \mathbb{V}(f_*) + \sigma_n^2$ due to assumption of noise. We will afterward average those quantities over test cases and over cross-validation loops and in the end argue between covariance functions based on the metrics.

**Meuse Again**

Again, we are going to use meuse dataset in order to demonstrate the gaussian process regression model selection method on a real problem. Let's also assume that zinc is again the variable of interest, using elevation, distance from the river and the coordinates as predictors once more. For the sake of our experiment we have nullified the zinc values of 40 points, in order to predict them later on. In figure 3.9 the samples are depicted with zinc levels colored with a continuous chromatic scale and null values represented with white. In this set, we have considered four classes of covariance functions (squared exponential, matern, rational quadratic and exponential sin squared), all of them augmented with a constant multiplying kernel to capture the magnitude and a white noise additive kernel to capture the

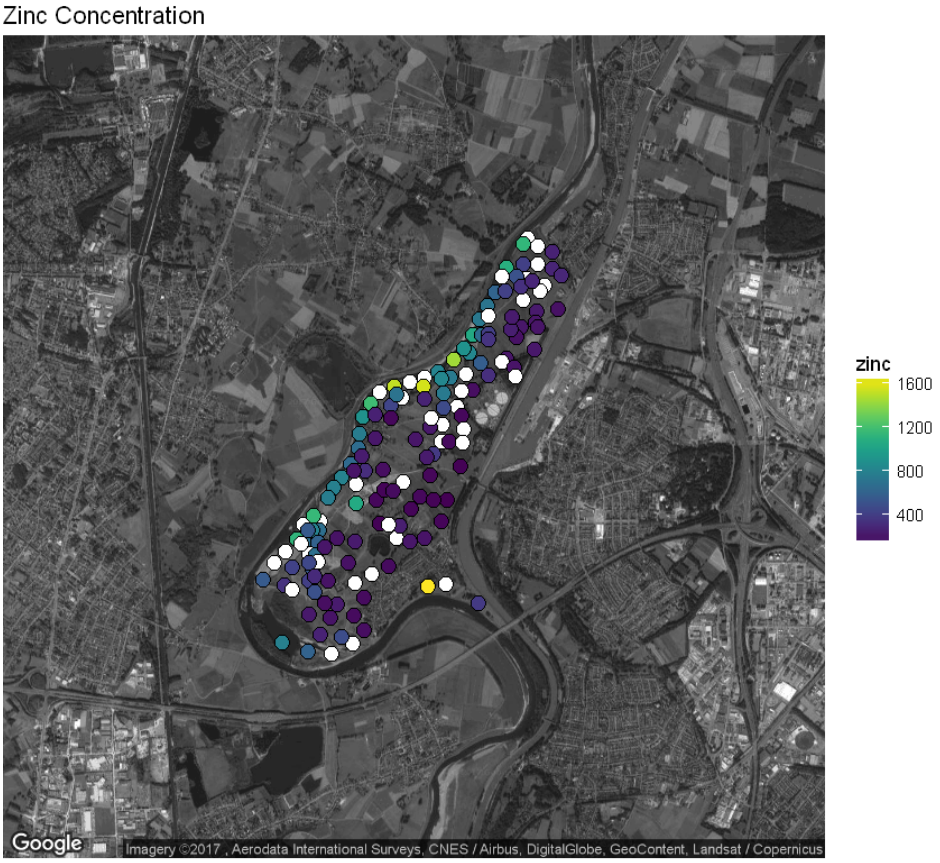



Figure 3.9: Plot of the samples with zinc level depicted through continuous color scale.

noise of the data. Our main goal with this dataset is not to be as sophisticated as we can be on our kernel choice, rather being confident that we can fit and compare kernels. With the process we described before, we cross-validated, fitted with log marginal likelihood and estimated them on their mean performance of standardized mean squared error and negative log probability density. In figure 3.10 we can see posterior covariance for each kernel and the respective performances in both metrics. With these at hand, we can pick the one that has the lowest standardized mean squared error and negative log probability, or pick the one that fits our needs if these two conditions do not coincide.

A first viewpoint to argue upon covariance functions has to do with error. Since we use the white kernel in order to estimate the noise level of the data, a posterior covariance with extended levels of noise and less weight put on the signal part of the covariance would surely be a not optimal way to describe the data. Since we have assumed from the beggining that our variable of interest is noisy, the thing we struggle to recover is the signal part. In the squared exponential case, we already know that large length scale values indicate independency between predictors and target variable. So one way to argue upon posterior covariances is through their estimated levels of noise. For example the RBF kernel uses an order of magnitude more noise than rational quadratic to describe the data. One reason that RBF kernel may be inappropriate to find relationships between data is that it is a case of a smooth, infinitely differentiable kernel and maybe this assumption is not a realistic one for our data. Another scope of comparison is of course the posterior length-scale parameter. We can say that it is a general rule that we want length-scale to be as small as possible for our input to be seen as highly related, although we must acknowledge there exists a threshold after which our kernel interpolates exactly the points and overfits on the data. On the other hand, Matern has a bit larger length scale than RBF and nu value equal to 1.5, a characteristic value for the "once differentiable" functions. An important comment here is that when nu parameter goes to infinity, matern and squared exponential kernels are equivalent. So nu's value ensures our thinking of squared exponential kernel being way too smooth for the description we need. As far as Exponential sin squared function, we can say again that this kernel explains nearly everything as noise, leaving aside the signal part of the kernel. Despite that, periodic kernel can be way more efficient when used in addition or multiplication with smoother kernels (although this lives outside the scope of this example). In the end, we can say that rational quadratic has the best scores among them, an acceptably low length scale, together with the lowest noise levels. We could make a lot more trials to find the suitable kernel, however between the tested one, we could say that rational quadratic is the more efficient to extract the underlying signal. In figure 3.11 we have also provided the predictions and the point estimates of the points of interest, using the rational quadratic kernel.

| | posterior covariance | SMSE | NLPD |
|---|---|---|---|
| 0 | 316**2 * RBF(length_scale=0.879) + WhiteKernel(noise_level=4.4e+04) | 2.829702 | 220.555307 |
| 1 | 316**2 * Matern(length_scale=1.07, nu=1.5) + WhiteKernel(noise_level=4.14e+04) | 2.392357 | 153.918649 |
| 2 | 316**2 * RationalQuadratic(alpha=0.158, length_scale=0.0544) + WhiteKernel(noise_level=4.02e+03) | 2.345033 | 138.135127 |
| 3 | 316**2 * ExpSineSquared(length_scale=4.1e+03, periodicity=1e+05) + WhiteKernel(noise_level=1e+05) | 15.149002 | 216.606436 |
| 4 | 316**2 * DotProduct(sigma_0=3.62e-05) + WhiteKernel(noise_level=6.04e+04) | 7.186723 | 147.640353 |

Figure 3.10: Posterior covariances for each kernel, along with the respective performance metrics.

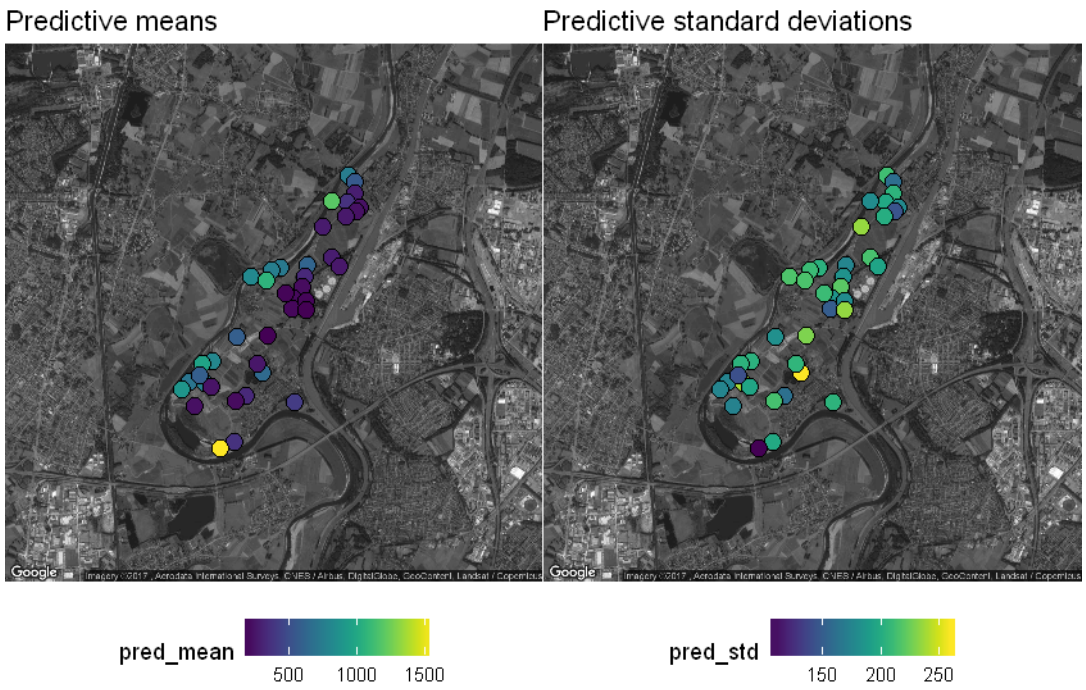



Figure 3.11: Predictions for the points of interest, along with the corresponding standard deviation of each.

*"Colours which appear through the Prism are to be derived from the Light of the white one."*

Isaak Newton

# 4
# Multispectral data as input

Remote sensing has been used in the agricultural sector for many years, aiming to offer the farmer a better understanding of their field and support his decisions. The underlying assumption that both remote sensing and this thesis has is that there is variability within the vineyard and the use of remote sensing data can offer information to understand it. It is in particular the use of multispectral imagery from airborne sensors that offer the remote determination of vineyard behaviour in terms of differing topography, soil characteristics, management practices, plant health and meso-climates (Bramley, 2003).

Now of course the underlying science lives in the sphere of optics and spectroscopy, where the interaction between matter and electromagnetic radiation is studied. As it turns out, materials and substances display a specific behaviour according to the type of wave that is cast upon them and using these unique spectral "fingerprints", we can locate and identify things that would be impossible to identify with just the human eye.

In agriculture, vegetation indices generated from the various spectral bands of aerial multispectral images are widely used. The most common indices provide a "means of capitalizing on the contrast that exists between vine biomass when measured in different wavebands" (Proffit et al., 2006). Normalized Difference Vegetation Index (NDVI) and the Plant Cell Density (PCD) or Ratio Vegetation Index (RVI) are two characteristic among

Figure 4.1: Visible light reflectance of sample case of a vineyard.

them

$$NDVI = \frac{NIR - Red}{NIR + red}$$
$$RVI = \frac{NIR}{Red}.$$

Both of these indices make use of the fact that healthy, vigorous vines will exhibit strong near-infrared reflectance and very low red reflectance, a behaviour that stems from the interaction between these radiations and chlorophyll. In Figure 4.2 we can see the pseudo-colour index image of a sample vineyard area, depicted in its normal visual reflectance in Figure 4.1.

Now, having already discovered the index that correlates with chlorophyl, one could say that we should just endeavor to discover the index that correlates with the quantity of interest e.g. ripeness. Since we are approaching the problem from its statistical machine learning viewpoint however, it makes sense to assume that there is a function of the inputs (reflectances or layers or bands) that approximates the ripeness inside the vineyard and apply a gaussian process regression in order to find the function and use it to predict-spatially interpolate.

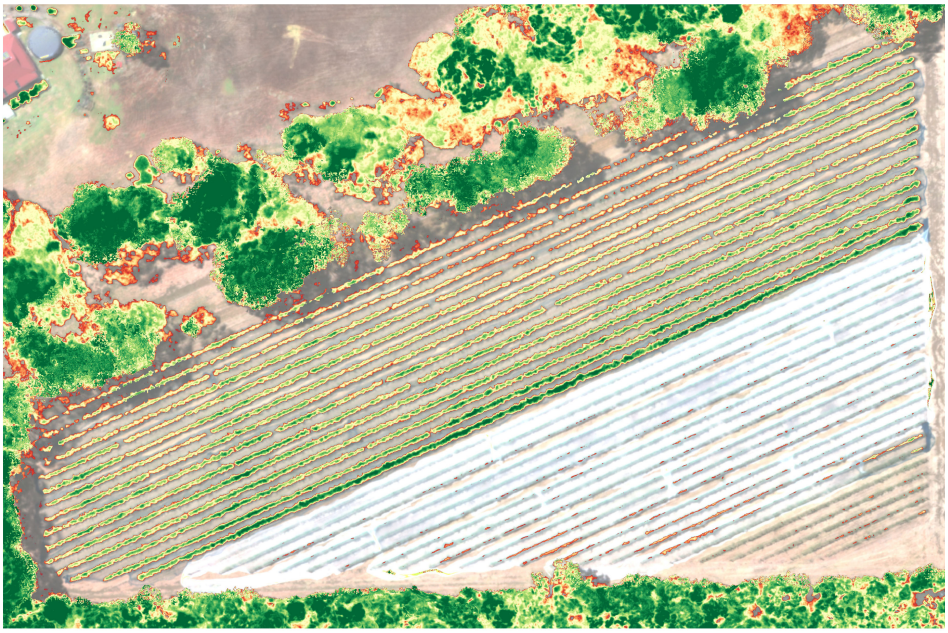


Figure 4.2: Normalized Difference Vegetation Index (NDVI) reflectance of sample case of a vineyard.

*"It is no coincidence that, on all four sides, in all four corners, the borders of the Roman Empire stopped where wine could no longer be made"*

Neel Burton

# 5 Our vineyard

Our case study for this thesis took place at Lytras Winery, Pikermi outside of Athens. The field we studied was of a single wine species, Malagouzia, and it extents over a 5,000 $m^2$ area. Now, as we already stated, our main goal is to predict/regress on a proxy related to grape ripeness. And as anyone could have guessed, sampling was the most intricate part of the process.

Figure 5.1: The three separate parts of lytras vineyard and winery. Our study took place in the middle one.

Since we studied the whole process, from strategizing the sampling procedure to predicting sugar/acidity in every vine, we found that the first step was one of the most difficult ones. In the classical statistics or machine learning approach, the main idea is that as the number of observations increases, the data becomes increasingly informative about $\boldsymbol{\theta}$. So following

this kind of thinking, we would want to sample from literally every location of the field, as many samples as possible. However, this viewpoint turns out to be problematic for various reasons. In the physical realm of agriculture, every observation represents a part of the crop, so each sample point we obtain means extra loss for the farming procedure. And since the whole analysis aims for the improvement of the farming process, it makes sense to want to gain insights while keeping the loss-sampling points number minimum. In our case study, we concluded that we have to constrain ourselves to 50 data points, each representing a set of 100 random grapes taken from a single vine. Now, the limited number of observations brings up the problem of how to obtain a random sample that is sufficiently informative to estimate the parameters at question. In order to handle this problem, we exploited a spatial statistics technique that creates a partition of the space-field into 50 strata (Figure 5.2, the interior of the white borders indicate strata) and we randomly chose one vine inside each strata. Figure 5.2 does not contain the quantification part of the vines, however presents accurately the raster creation of the field, the partition of the space into sub-regions and the random pick from each one of them. One could again argue that 50 sample cases are not enough to "learn" an area of 1197 vines, however the simple modification that one could make to add information (if he can afford it) is to pick n vines from each strata instead of one.

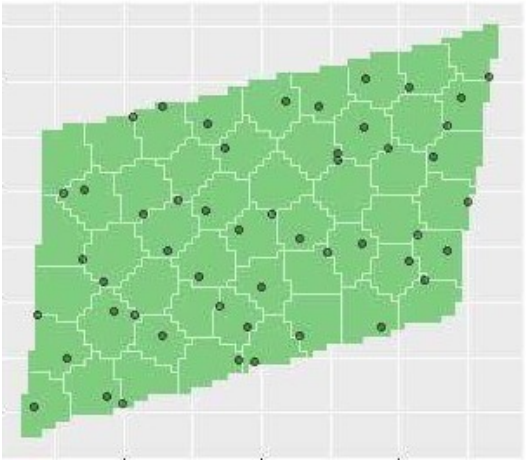


Figure 5.2: Abstract visualization of the spatial sampling method we used to select observations.

After sampling, we analyzed the data and measured four quantities of interest for each of them, weight (of 100 grapes sample), TAP (probable alcohol

content), ph and total acidity. The variable we are going to use for our spatial interpolation methodology is the fraction TAP/Total Acidity. Now, apart from the chemical and physical variables, we have also captured RGB and Infrared images of the field and we have augmented them with geolocation data. This way, after creating an extra ndvi variable $\frac{infrared-red}{infrared+red}$, we managed to obtain a reflectance in each band for every lat/long location of the field. Figure 5.3 showcases the pseudo-color/visible light and the infrared editions of the field and Figure 5.4 depicts a data cube, the standard data type for multispectral geospatial data.

Figure 5.3: RGB-visible light and infrared reflectances of the field.

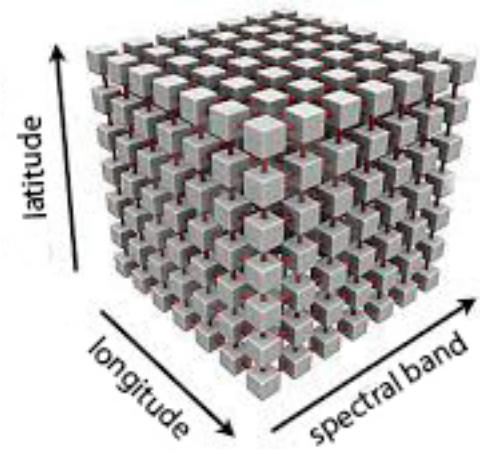



Figure 5.4: A data cube: data type we end up with when working with multispectral geospatial data.

What we had to do next was to merge these two data sources and since both of them are geolocated, a spatial join can do this job pretty efficiently. However, before joining them, a last pre-processing step is important to be taken into consideration. Since we want to extract a vine's reflectance and attach it to its location, it makes sense to not use the reflectance of the center of the vine but consider that a vine extends a bit further if you include arms, leaves, canes and canopy. In order to enclose all this information (the reflectance from every part of the vine), we aggregated the cells by a factor of 10 using mean function. This spatial resolution adjustment procedure is showcased in Figure 5.5, where we used a $2x2$ window (factor of 2) and max function in this case.

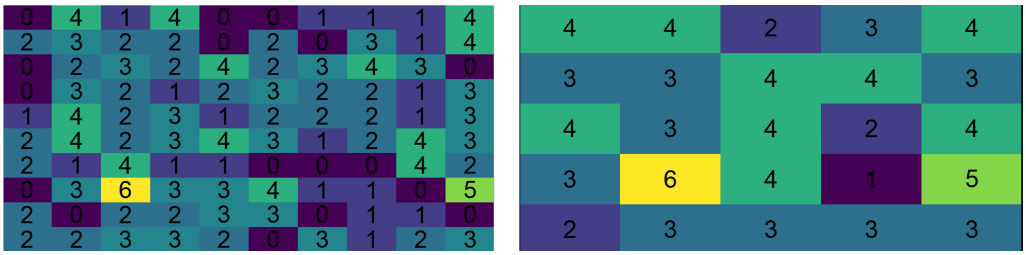



Figure 5.5: Aggregation on a raster by a factor of 2, using max function.

| red | green | blue | ir | ndvi | x | y | TA.OO |
|---|---|---|---|---|---|---|---|
| 47.454519 | 48.289348 | 36.637008 | 106.88 | 0.385081 | 23.951851 | 38.005501 | 1.227723 |
| 75.508617 | 68.965478 | 44.009110 | 111.43 | 0.196959 | 23.951954 | 38.005538 | 1.343750 |
| 99.689756 | 93.139695 | 63.837746 | 104.54 | 0.033965 | 23.951624 | 38.005480 | 0.927419 |
| 66.554706 | 66.616330 | 46.729278 | 109.90 | 0.262506 | 23.951857 | 38.005544 | 1.347368 |
| 91.807513 | 87.032076 | 59.069802 | 113.33 | 0.114995 | 23.951748 | 38.005523 | 1.096154 |

Figure 5.6: Sample rows of the data frame we are modelling.

After all these preprocessing steps, the data frame we end up with is of $1197x8$ dimension, some sample rows of which can be seen in Figure 5.6. Now, in Figure 5.7 we present our 50 observations, depicting TAP/Total Acidity through a continuous chromatic scale from purple to yellow. The first comment on our observation is that spatial sampling approach with strata manages not to leave large areas unsampled. However, since the variability we mentioned is apparent with no simple evident pattern (e.g.

while latitude increases, outcome variable increases too), the more points we obtain per strata the better. While there is no large part of the field unexplored, the magnitude of the variability indicates that the number of observation points is indeed small.

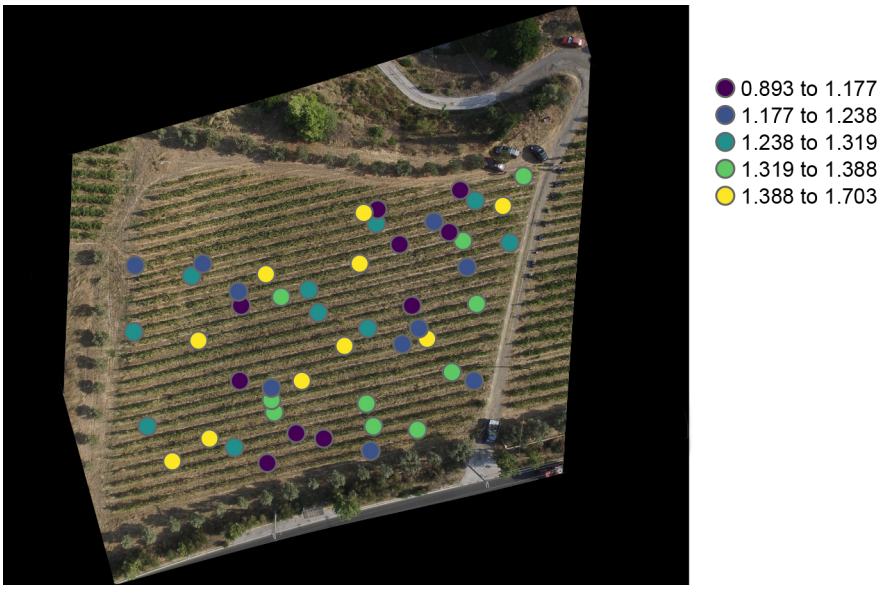



Figure 5.7: Observed TAP/Total Acidity and the corresponding locations.

While in a synthetic dataset one could reason upon the use of kernel and apply kernel engineering tricks e.g. observe a periodic motion and use a periodic kernel to capture it, instead we are aiming to establish a more systematic way of kernel choosing. The general idea is that we use the set of single kernels we can apply, use the powerset that the sums and products between them creates, stay indifferent between the elements of the powerset and let our performance benchmarking choose the most valid one for us. While we did not use every single element of the powerset in the end, we showcase in Figure 5.8 a large number of kernels and their scores in Standardized Mean Squared Error and Negative Log Probability Density.

As we can see in the benchmarking results, apart from scoring and predictive performances, every kernel understands the data in a different way. While a periodic kernel by itself(ExpSineSquared) displays low periodicity and relatively high length-scale, when combined through sum with RBF or RQ presents significantly lower length scale and significantly larger periodicity. This happens because the non-periodic part of the kernel takes over

the trend of the function and leaves the periodicities to ExpSineSquared, which makes length-scale really small to capture the sparse periodicities and periodicity parameter really big since these patterns do not repeat themselves often. We can also see that the products between RBF,Matern and RQ do not show any particular strengths or weaknesses, since one of the components gets small enough length-scale parameter to describe-interpolate the data and the other one describes a small part of what remains, so displays huge length-scale parameter. Furthermore, one of the interesting observations is that while Matern kernels learn nu parameter equal to 1.5, which translates to a one differentiable function, Matern does not display significant performance difference with RBF kernel. That can be translated to the underlying function not having any strong discontinuities or non-differentiable points ' infinite differentiability assumption does not seem that irrational. Although we don't have any specific reason to pick the best kernel, especially since this choice involves some bias about the form, the characteristics and the hyperparameters we think the posterior function is acceptable to have, we chose the one with lowest SMSE and NLPD in order to make the final predictions and uncertainties plot. In Figure 5.9, we can see that the spatial autocorrelation is again apparent. The spatial autocorrelation that has been absorbed from the data and from our kernel, which interprets the notion of distance, is depicted in our predictions. One could vaguely mark sub-regions inside which the values are close. Similar regions could someone denote in Figure 5.10 for uncertainties e.g. values seem to be higher near the edges.

| posterior covariance | SMSE | NLPD |
|---|---|---|
| 1.29**2 * RBF(length_scale=31) + WhiteKernel(noise_level=0.0278) | 1.083689 | 0.117819 |
| 1.29**2 * Matern(length_scale=52.8, nu=1.5) + WhiteKernel(noise_level=0.0278) | 1.083825 | 0.117908 |
| 1.28**2 * RationalQuadratic(alpha=0.00146, length_scale=25.7) + WhiteKernel(noise_level=0.0277) | 1.111897 | 0.120626 |
| 1.29**2 * ExpSineSquared(length_scale=7.79, periodicity=24.9) + WhiteKernel(noise_level=0.0278) | 1.072662 | 0.167999 |
| 1.29**2 * RBF(length_scale=31) * RBF(length_scale=1.07e+03) + WhiteKernel(noise_level=0.0278) | 1.083686 | 0.117819 |
| 1.29**2 * RBF(length_scale=1.03e+04) * Matern(length_scale=52.8, nu=1.5) + WhiteKernel(noise_level=0.0278) | 1.084541 | 0.117923 |
| 1.29**2 * Matern(length_scale=52.8, nu=1.5) * Matern(length_scale=7.78e+04, nu=1.5) + WhiteKernel(noise_level=0.0278) | 1.084027 | 0.117912 |
| 1.29**2 * RBF(length_scale=31) + 0.00473**2 * RBF(length_scale=118) + WhiteKernel(noise_level=0.0278) | 1.125954 | 0.122336 |
| 1.15**2 * ExpSineSquared(length_scale=0.228, periodicity=7.18e+03) + 0.574**2 * RBF(length_scale=13.9) + WhiteKernel(noise_level=0.0278) | 1.060862 | 0.117571 |
| 0.684**2 * ExpSineSquared(length_scale=21.1, periodicity=0.000491) + 3.1**2 * RationalQuadratic(alpha=1.43e-05, length_scale=0.0424) + WhiteKernel(noise_level=0.0248) | 1.088955 | 0.117613 |
| 0.0416**2 * ExpSineSquared(length_scale=0.0729, periodicity=17.6) + 1.29**2 * Matern(length_scale=61.9, nu=1.5) + WhiteKernel(noise_level=0.0263) | 1.125467 | 0.120851 |

Figure 5.8: Benchmark results of the medifferent posterior covariance functions.

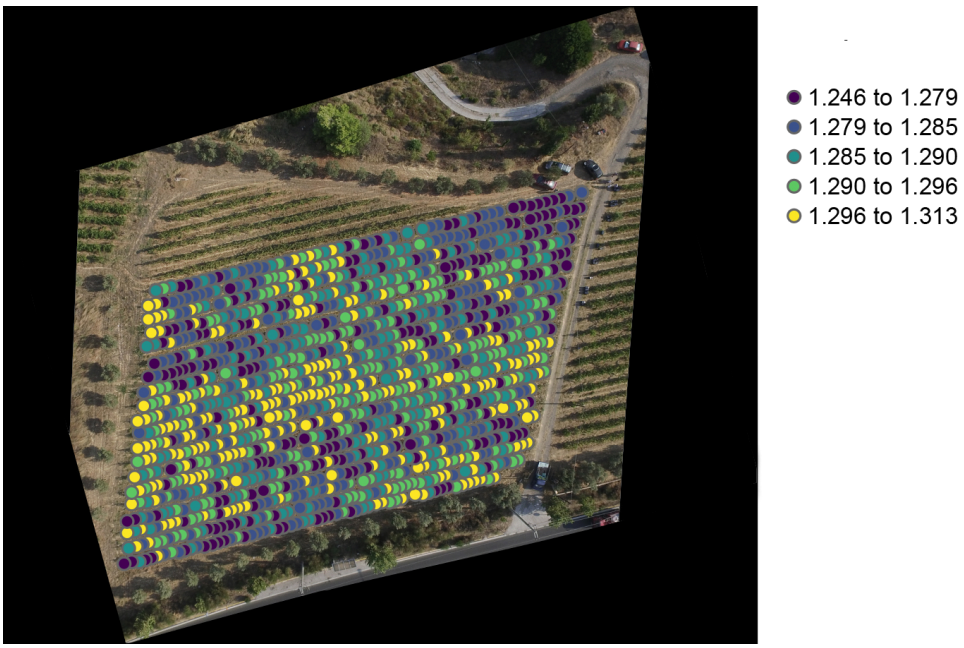



Figure 5.9: Predicted values for test set.

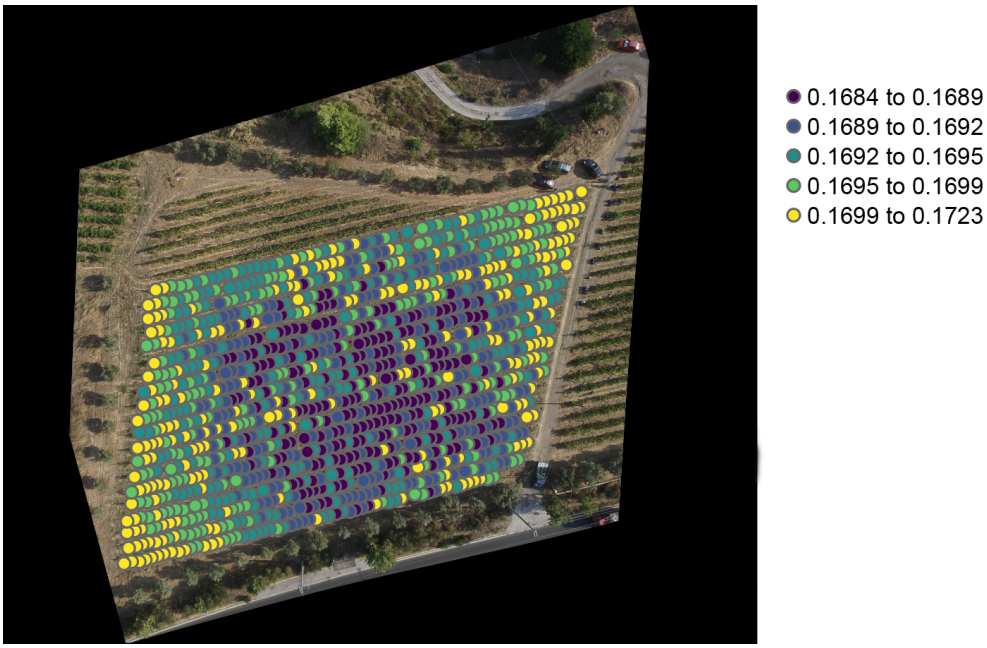



Figure 5.10: Predictive uncertainties for test set.

# Epilogue

One of the basic ingredients of the process we followed to spatially predict the ripeness proxy variable is the combination of multispectral input together with the machine learning framework. For years, multispectral data have been exploited to uncover certain characteristics that physics can interpret. Experts were then supposed to infer upon the informative sensor measurements, aiming to find ways to interpret the process. And of course her whole analysis would be depended upon our capability to experimentally (the physics way) find the spectral proxies that can explain the variability of ripeness, flavor, vigor etc. Now, machine learning appears to be a promising tool to capture the spatial manifestations of grapes and any agricultural good in general, able to fill the, up until lately, biased space between remote sensing and farming. Furthermore, Gaussian process in specific seems powerful enough to locate the required transformation of inputs (since we are exploring the projection space), that maximizes our predictive capabilities toward any process characteristic we deem useful.

Through our study and trials, an interesting note came upon us. We combined the use of marginal likelihood optimization to pick the best hyperparameters for a given model family-kernel with predictive metrics, supported also by cross-validation to avoid overfitting and perform best to the unknown dataset. Even better, since we work with gaussian processes, we gain inferential capabilities that classical machine learning lacks. The extra evident mathematical steps on the path to "automated machine learning" would be to enrich our base kernels list (RBF, RQ, Matern,...) and better compute the set that sums and products of them create. And there is indeed plenty of work on this area from research groups like the Automatic Statistician Project (Figure 5.11), which aims to build artificial intelligence for data science itself. Acknowledging our distance from an entity like this and returning to our main interest being vineyard applications, there are some obvious ways to improve the process as it is.

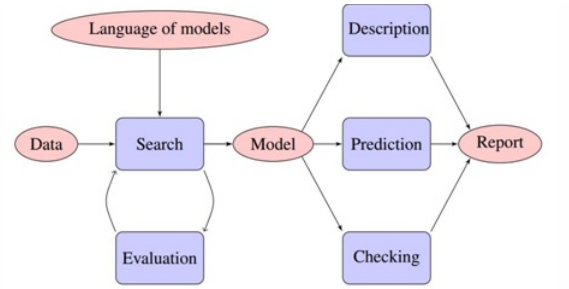



Figure 5.11: A draft of what Automatic Statistician Project is trying to accomplish.

The ripening process is clearly also related to variables about soil concentration in certain substances, water, morphological attributes of the field, average sun exposure through the year, root depth, canopy weight, temperature, rain, climate etc. So additional measurements or proxy predictors should be included and indeed remote sensing is a promising source for some of them, due to the easiness of measuring multiple locations in large areas. Again, extended research has come up with ways of using thermal input, spectrums to estimate soil composition etc. Time can also prove very useful and the framework we used can easily be extended to spatio-temporal data and offer us insight not only about the ripeness level but its progress too. And that would be one more reason to make use of additional basic kernels like heteroscedastic ones that treat variability in a more clustered way. Progress should lastly be made in the image processing frontier to adjust for sun, clouds and noise in general during imaging, as well as transformations on imagery like convolution or clustering to reason upon the most informative version of the input data.

# Bibliography

## Mathematical Books


[BPGRP08] Roger S Bivand, Edzer J Pebesma, Virgilio Gomez-Rubio, and Edzer Jan Pebesma. *Applied spatial data analysis with R*, volume 747248717. Springer, 2008.

[RW06] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.


## Viticulture Books


[RML+13] C Rey, MP Martín, A Lobo, I Luna, MP Diago, B Millan, and J Tardaguila. Multispectral imagery acquired from a uav to assess the spatial variability of a tempranillo vineyard. In *Precision agriculture13*, pages 617–624. Springer, 2013.

[SSS16] JL Smit, G Sithole, and AE Strever. Vine signal extraction–an application of remote sensing in precision viticulture. *South African Journal of Enology and Viticulture*, 31(2):65–74, 2016.


## Miscellaneous


[DS15] Jinshu Wang Dennis Sun. Stats 253: Analysis of spatial and temporal data, stanford university, 2015.

[Wic] Charlotte Wickham. Working with geospatial data in r.