Forecasting Project Report

Title:

Forecasting: Gaussian Regression

BY

Azza Kamoun

Mouna Hmani

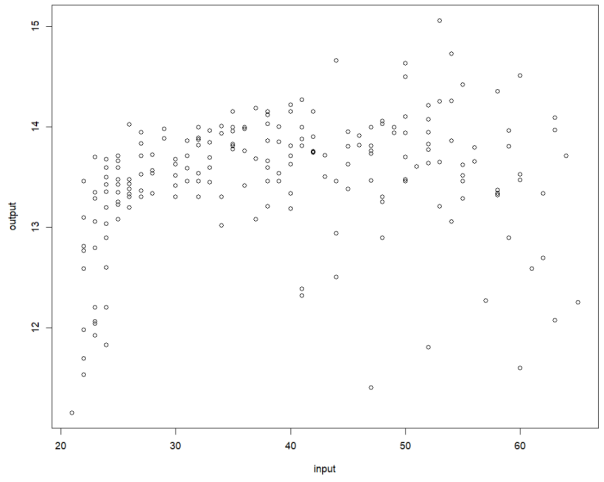
1) Exploring the Dataset

a) Package

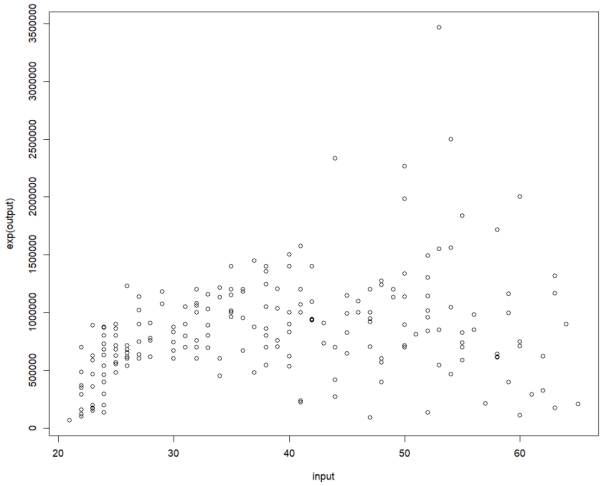
The dataset we worked on is imported from the R Package 'SemiPar' which was published on October 12, 2022 by Matt Wand. The package mainly has functions for semiparametric regression analysis.

b) Dataset

"age.income" is a data frame. It has 205 pairs of observations that give insight on the age and income of Canadian workers in 1971.
age represents age in years & **log.income** represents the logarithm of income



Age VS Log(income)



Age VS income ($)r

c) Variables

- Age:
The range of ages is between 21 and 65 with an average of 39 years old.
- Log(income):
Log income is between 11 and 15, in other words, the incomes are between 60K/$ and 3M/$ with an average of 700K/$

```
summary(input)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
21.00   27.00   38.00   38.85   49.00   65.00

summary(output)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
11.16   13.30   13.61   13.49   13.87   15.06
```

2) Theory: Expected Methods & Results

a) Gaussian process (GP)

GP is a **generic supervised learning** method that is generally used for regression and classification problems.
One of the main features it may have is that its observations have a multivariate normal distribution which makes the algorithm itself being characterized by:

- The mean (mean vector and mean function)
- The covariance (covariance matrix and covariance function)
-

Y → The Gaussian process that will be generated in every simulation
This one can be written as follows: Y (x) ~ GP(m(x), k(x, x0)).

such that **k** is the covariance function which is crucial to estimate the predictions.

b) Hyperparameter:

Theta θ: The length parameter is a parameter used to determine the correlation function

Sigma σ: Noise is also an important parameter for the correlation function.

c) Model Evaluation

MSPE: To calculate the error we used the mean squared prediction error (MSPE) and it is calculated by the mean of the squared difference between our output and predictions.

3) Practical: Implementation of the Gaussian Regression:

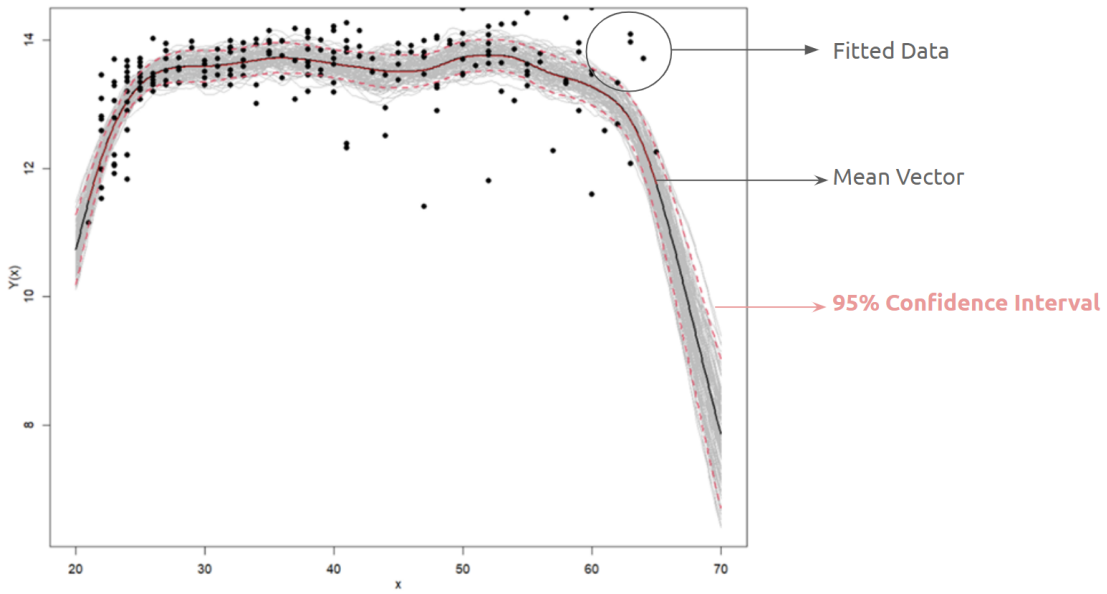
a) Basic Algorithm

As the main goal of the project is to work on the optimization of the gaussian regression, we started by developing a first **basic code**, that will later get enriched.
→ The "basic algorithm" took **random parameters**, and the covariance function was also randomly chosen (Matern 3/2).

Steps to develop the "Basic Algorithm":

1- Loading the data form 'SemiPar'
2- Defining the parameters: number of simulations, number of discretizations, sigma, theta..
3- Defining the covariance function (3/2 mater)
4- Intermediary and Final calculation of the **conditional Covariance Matrix** and **Conditional Mean Vector** using the cholesky method.
[Main results]
5- Generate the output of each of the simulation (Y)
6- Preparation of the **Conditional Covariance** and **Conditional Mean Functions.**
7- Creation of the "Kriging /mean" function (will be used for predictions)
8- Regression: Estimating the data based on the mean function

Steps to develop the "Basic Algorithm":



After running the algorithm, this is the regression obtained after applying the gaussian simulation. One can notice that most of the simulations are within the 95% confidence interval.
FYI: There is no numerical assessment at this stage since this is just a starting point.

b) Optimization 1: Test & Training the Dataset

Given the first results obtained from the basic code, our aim is to work on its optimization. Since we are dealing with a supervised learning technique, one of the first optimizations we may do is dividing the dataset into test and training sets.

To determine the best split ratio, we experimented with 2 ratios and compared the MSPEs:

	70/30	80/20
MSPE	**0.34**	0.37

→ **Result:** For the following optimizations we will be using the split ratio of 70% for the training set and 30% for the test set.

c) Optimization 2: Testing New Covariance Functions

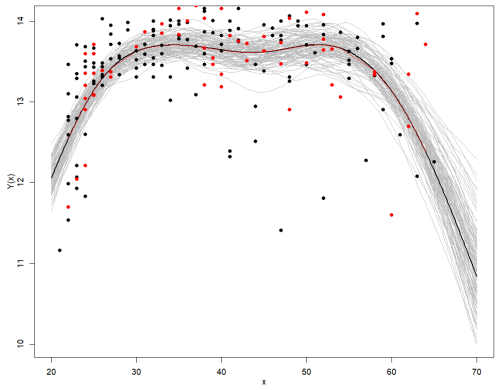
Now that we have divided our dataset, we are going to focus on the optimization of the covariance functions since they are the main determinants of the regression output. We opted for 4 main functions: Matern 3/2 - Matern 5/2 - Exponential and Squared Exponential

	Numeric	MSPE
Matern 3/2	$\left(1+\frac{\sqrt{3}\lvert x-x'\rvert}{\theta}\right)\exp\left(-\frac{\sqrt{3}\lvert x-x'\rvert}{\theta}\right)$	0.34
Matern 5/2	$\left(1+\frac{\sqrt{5}\lvert x-x'\rvert}{\theta}+\frac{5(x-x')^2}{3\theta^2}\right)\exp\left(-\frac{\sqrt{5}\lvert x-x'\rvert}{\theta}\right)$	0.34
Exponential	$\exp\left(-\frac{\lvert x-x'\rvert}{\theta}\right)$	0.41
Squared Exponential	$\exp\left(-\frac{(x-x')^2}{2\theta^2}\right)$	0.28

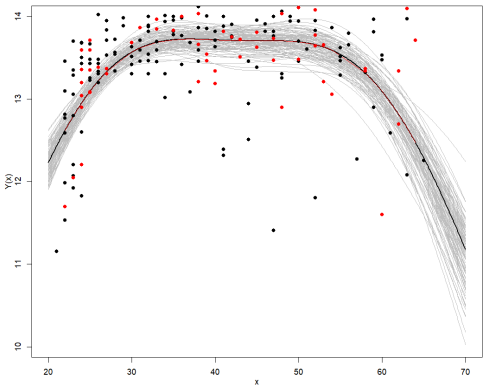
→ **Result:**

- **Numerically:** The most optimal covariance function amongst the latter is the **Squared Exponential** which MSPE is the lowest compared to the rest (0.28). Meanwhile, the **Exponential** is the least performant one with an MSPE of 0.41.

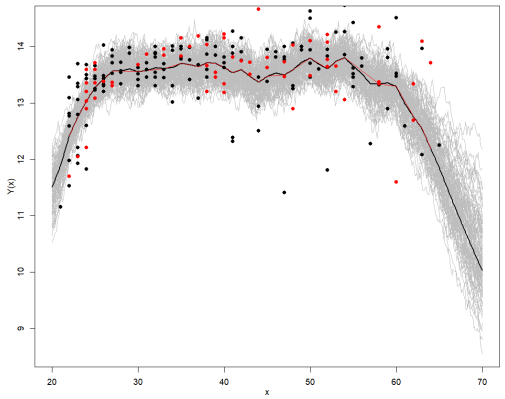
- **<u>Graphically:</u>** The results above can be easily interpreted in the graphs as well since the Squared Exponential graph has the **best smoothness,** contrarily to the exponential which is very unsmooth.



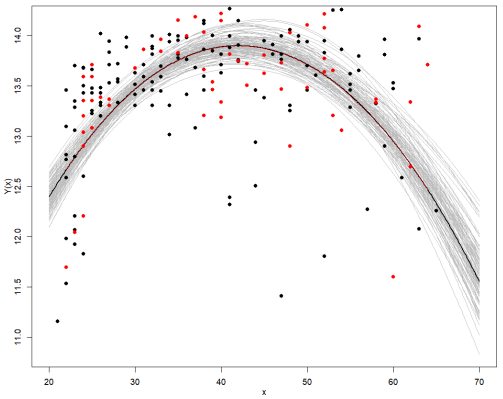
Matern 3/2 (Relatively Smooth)



Matern 5/2 (Relatively Smooth)



Exponential (Very NOT Smooth)



Squared Exponential (Very Smooth)

d) Optimization 3: Estimating Better Parameters

As any machine learning model, Gaussian Process has several hyperparameters and as we know changing the hyperparameters gives different predictive performance to our model.
Optimization of these parameters could be unachievable if we try to do them manually. Thus, it is needed to work on a function to minimize the MSPE to and give the optimal parameters for the optimal results.
In our case we have two hyperparameters to optimize:

- **The theta:** which is the difference between our instances
- **The sigma:** which is the standard deviation of the noise.

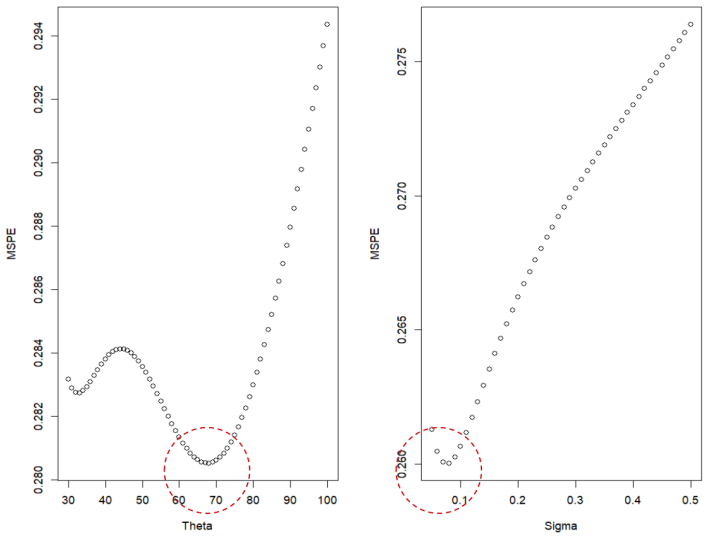
→ To calculate the error we used the mean squared prediction error **(MSPE)** and it is calculated by the mean of the squared difference between our output and predictions.

→ **The function "optim"** was used to give us new values for the parameters in order to minimize the MSPE. The function "optim" provides algorithms for general-purpose optimisations .
The function method works with different methods, we are gonna be working with three different methods which are the **SANN, CG** and **BFGS.** (d.2- Numerical Optimization with "optim")

Methods	Def
SANN	is by default a variant of simulated annealing given in Belisle (1992). Simulated-annealing belongs to the class of stochastic global optimization methods
CG	is a conjugate gradients method based on that by Fletcher and Reeves (1964).Conjugate gradient methods will generally be more fragile than the BFGS method, but as they do not store a matrix they may be successful in much larger optimization problems.
BFGS	Method "BFGS" is a quasi-Newton method (also known as a variable metric algorithm), specifically that published simultaneously in 1970 by Broyden, Fletcher, Goldfarb and Shanno. This uses function values and gradients to build up a picture of the surface to be optimized

d.1- Graphic Optimization for initialization:

→ It should be known that **our optimum function** depends enormously on the **initialized value.** This was deducted from running the function with different initialized values manually when all the optimized parameters were close to the initial values . Thus to choose the optimal initialization we tried to visualize how the function MSPE will behave with the different values, which gave us the following results and then we took approximate values and worked on our optimal function.



→ **Result:** We will chose the following values: **Theta = 70 , Sig = 0.1**

d.2- Numerical Optimization with "optim":

Comparing MSPEs:

	MSPE Before Optimization	MSPE after Initialization	MSPE after the Optimization		
			BFGS	SANN	CG
Matern 3/2	0.34	0.261	**0.2605**	0.2608	**0.2605**
Matern 5/2	0.34	0.262	**0.2597**	0.2599	**0.2597**
Exponential	0.41	0.369	**0.283**	0.284	0.285
Squared Exponential	0.28	0.274	0.2596	0.2693	**0.2595**

Optimized Parameters:

	Parameters Before Optimization	MSPE after Initialization	Parameters after the Optimization		
			BFGS	SANN	CG
Matern 3/2	σ = 40 θ= sd(train.output)	σ = 70 θ= 0.1	σ =70 θ= 0.153	σ = 65.91 θ= 0.17	σ = 70 θ= 0.26
Matern 5/2			σ = 70 θ= 0.05	σ = 62.51 **θ= - 0.07**	σ = 70 θ= 0.05
Exponential			σ = 75.8 θ= 0.59	σ = 72.03 θ= 0.60	σ =70.05 θ=0.59
Squared Exponential			σ = 69.99 θ= 0.002	σ = 61.53 **θ= - 0.04**	**σ =69.99** **θ=0.002**

→ **Result:** With a given dataset and an aim to estimate the income of a Canadian worker with respect to age, 4 attempts at optimizing the Gaussian Process have been undergone.

1. **Matern 3/2:** In the first attempt we worked with the matern 3/2 funcion, The best result of the MSPE here was 0.2605.
2. **Matern 5/2:** In the second attempt to optimize GP, the covariance function is replaced with another, resulting in an MSPE of 0.2597, which is higher than that of Matern 3/2.
3. **Exponential:** In the third attempt to optimize GP, the covariance function is replaced yet again, however this time the MSPE was 0.283, which happens to be ranking it the weakest amongst.
4. **Squared Exponential:** The last thing was to change the covariance into the squared exponential function and we got the 0.2595 which was the best results among all the covariance functions.

→ The method used in the "optim" function resulted in different parameters but approximately close. The most optimal one in our case in terms of results and computational time is **CG,** the conjugate gradients method. It is worth mentioning that the fastest method was BFGS and slowest was SANN.

4) Conclusions

a) Summary

This report dealt with the implementation of a gaussian process to the data set of canadian log income in regards to age. In order to get the best results, after inspecting the data we split our data into two sets; the training and the testing to test our model. And then we started with the optimization of the model by changing the covariance function. The mean of squared predicted error was used to compare how the model is performing each time. At a first instance we got the optimal MSPE from the covariance function of squared exponential. After that we needed to do further optimization on the hyperparameters using the optimization function in R with several methods such as SANN , CG and BFGS. This function was used on all cases of the covariance to compare if after the optimization will get different results. Even Though we got closer values of MSPE, the squared exponential gave us the best results and optimal model with the use of CG as an optimization method.

b) Perspective

There is no debate around how solid of an algorithm Gaussian Processes is, however it is undeniable that there are more algorithms that have proven competent at giving strong predictions, and the process of getting these predictions has been made easier with the help of this code snippet:

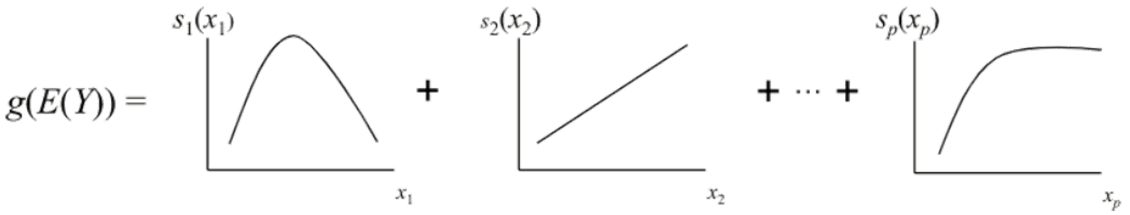
```
> pf <- ggplot(age.income, aes(age, log.income)) + geom_point() + xlab("Age") + ylab("log.income")

>pf+geom_smooth(method="gam", formula = y ~ s(x, bs = "cs"))
```

Code	Explanation
ggplot	is to plot a scatterplot that takes 3 parameters to reveal the correlation between x and y.
age.income	providing the dataset source
aes()	the mapping, we specify what goes on the x axis and what goes on the y axis
age, log.income	age goes on the x axis while log.income goes on the y axis
X lab and y lab	adding a label to the x and y axis
Pf +	to plot the scatter plot and add to it
geom_point	to specify that the data is visualized as points/dots
Geom_smooth	function to add a regression line to the scatterplot

Method	specify the method to be used, in this case GAM
formula = y ~ s(x	the formula to be used, in this case, computing the regression between y and x
bs = "cs"	specifies a shrinkage version of the attribute “cr”, which is cubic regression; it takes less points to form the regression line.

Generalized Additive Model (GAM) is an additive technique used to create models where the impact of predictive variables is smooth, even if they are not linear functio/ns.



Why should we consider using GAM method:

1. Easy to analyze.
2. Flexible predictor functions can find hidden patterns present in the data.
3. Regularization of predictor functions helps avoid overfitting.

References

- "Optim: General-purpose Optimization" , RDocumentation https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/optim
- H. Maatouk and X. Bay. Gaussian process emulators for computer experiments with inequality constraints Mathematical Geosciences, 49(5):557–582, 2017.
- "The principle, visualization and code implementation of Gaussian Processes" Earth System Science and Remote Sensing- Nov 15, 2020 link
- A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions EricSchulza Maarten Speeken brinkb Andreas Krausec link
- Package 'SemiPar' October 12, 2022 Documentation link
- C. E. Rasmussen and C. K.I. Williams.
- Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning) 2006

