A Case Study of Gaussian Process Regression

Zhang Genghan

zhanggh19@mails.tsinghua.edu.cn

January 9, 2022

Abstract

Gaussian Process Regression(GPR), a class of supervised machine learning algorithms, can make predictions with few parameters. In this paper, we will revisit the theory foundations and the basic assumptions of GPR, and conduct a case study of stock trends prediction using GPR.

1 Introduction

Regression is to predict continuous quantities based on the known data. Meer et al. gave a review about regression methods. We focus on Gaussian Process Regression(GPR) in this paper. GPR is also known as Kriging in geostatistics, and surrogates or emulators in computer experiments Liu et al..This way of using Gaussian processes in modeling deterministic functions is becoming popular in machine learning Kanagawa et al.. We follow Rasmussen to interpret GPR from two equivalent views, weight-space view and function-space view. Both assume some gaussian properties of input data and the model itself. However, they are different on the sample space. We choose stock trends prediction as a case study because historical stock price data is stochastic and time-series Qiu et al. which can be solved by GPR theoretically. We use the Gpytorch developed by Gardner et al. to implement the algorithm and choose the stock prices of HPE, INTC and AAPL from 2016-12-05 to 2021-12-03[1] as the dataset. We choose those three datasets based on the *capital asset pricing model* Jensen et al.. Because the stock price of larger enterprises are less influenced by accidents, they can fit the GPR better. The following paper is organized as: Section 2 introduces some basic concepts and theorems, Section 3 revisits weight-space view and function-space view, and Section 4 describes the basic GPR algorithm and the algorithm we use in our case study. Our experiments[2] are detailed in Section 5. Section 6 concludes the paper.

2 Preliminary

In this section, we will introduce *Gaussian Distribution*, *Joint Gaussian Distribution*, *Gaussian Process*, and *Linear Regression* which are the basis of GPR. We will also introduce three theorems necessary for deriving GPR.

[1]The data were downloaded from `https://finance.yahoo.com/`

[2]The code can be found at `https://github.com/zhang677/A-case-study-of-GPR`

2.1 Definitions

2.1.1 Gaussian Distribution

Definition 1. *A random variable X is Gaussian distributed with mean μ and variance σ^2 aka. $x \sim \mathcal{N}(\mu, \sigma^2)$if its probability density function(PDF) is:*Murphy

$$P_X(x) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$

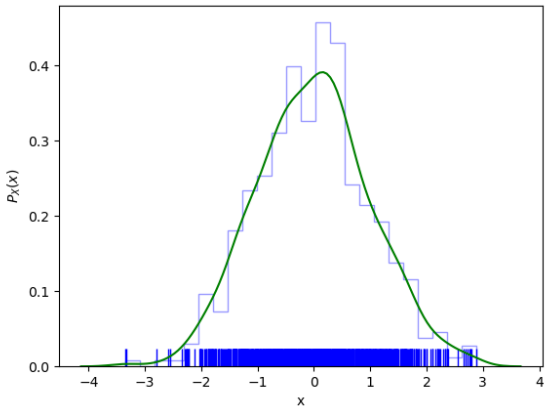


Figure 1: One thousand normal distributed data points are plotted as blue vertical bars on the x axis. The PDF of these data points was plotted as a green bell curve.

2.1.2 Joint Gaussian Distribution

Definition 2. *n random variables $\mathbf{X} = (X_1, X_2, ..., X_n)$ are joint Gaussian distributed aka. $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ if their PDF is:* Murphy

$$P_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} exp\left(\frac{-1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2}$$

where $\boldsymbol{\mu} = \mathbb{E}[\mathbf{X}] \in \mathbb{R}^n$ is the *mean vector*, $\boldsymbol{\Sigma} = \mathbb{E}[(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^T] \in \mathbb{R}^{n\times n}$ is the *covariance matrix*.

2.1.3 Gaussian Process(GP)

Definition 3. *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.* If real process $f(\mathbf{x})$ is GP, it is completely defined by its *mean function* $m(\mathbf{x})$ and *covariance function* $k(\mathbf{x}, \mathbf{x}')$, *aka.* $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ Rasmussen

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \end{aligned} \tag{3}$$

A more rigorous definition can be found in Kanagawa et al..

2.1.4 Linear Regression

Since we only consider the time-series data in this paper, we only focus on one-dimension input and continuous one-dimension output. However, we still need to review the multidimensional input case, because the *kernel method* will transform the one-dimension input to multidimensional first and then conduct the linear regression. We will review the *kernel method* after that.

A *training set* $D = \{(\mathbf{x}_i, y_i)|i = 1, 2, ...n, \mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}\} = X, \mathbf{y}$, where $\mathbf{x}$ is called *input vector* and y is called target. We also have a *test set* $T = \{(\mathbf{x}_{*i}, y_{*i})|i = 1, 2, ...k, \mathbf{x}_{*i} \in \mathbb{R}^m, y_{*i} \in \mathbb{R}\}$. Ng pointed out that the goal of regression is, given a training set, to learn a function $f : \mathcal{X} \to \mathcal{Y}$ so that $f(\mathbf{x}_*)$ is a "good" predictor of y_* in the test set . In linear regression, $f = \mathbf{x}^T\mathbf{w}$ where input vectors are multidimensional. GPR makes some assumptions on the input vector $\mathbf{x}$ and weights $\mathbf{w}$ so that it can predict the distribution of y_* and define a metric of the prediction.

Obviously, the expressiveness will be limited if input vectors are one-dimension. One direct solution is to project inputs into some high dimensional space and do linear regression in that space, which is called *kernel method*. More formally, $f = \phi(\mathbf{x}^T)\mathbf{w}, \phi : \mathbb{R} \to \mathbb{R}^m$. We will further explain this idea in Section 3. More details about kernel methods can be found in Rasmussen.

2.2 Theorems

Theorem 1. *Linear Operation reserves Guassian: Given* $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \mathbf{Y} = \mathbf{AX} + \mathbf{B}$,*then* $\mathbf{Y} \in \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{B}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T)$

Theorem 2. *Conditional Gaussian: Given* $\begin{pmatrix}\mathbf{X_1}\\ \mathbf{X_2}\end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu_1}\\ \boldsymbol{\mu_2}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right), \mathbf{X_1} \in \mathbb{R}^n$ *and* $\mathbf{X_2} \in \mathbb{R}^m$*, then*

$$\mathbf{X_1}|\mathbf{X_2} \sim \mathcal{N}(\boldsymbol{\mu_1} + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x_2} - \boldsymbol{\mu_2}), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21})$$

Theorem 3. *Swap the condition with the random variable: Given random variables* W, X, Y*,* X *and* W *are independent, then* $P(W|X, Y) = \frac{P(Y|X,W)P(W)}{P(Y|X)}$.
proof: Bayes's Rule: $P(W, Y|X) = P(W|X, Y)P(Y|X) = P(Y|X, W)P(W|X)$, X and W are independent: $P(W|X) = P(W) \Rightarrow P(W|X, Y) = \frac{P(Y|X,W)P(W)}{P(Y|X)}$ □

3 Two views

In contrast with the normal linear regression whose output is a real value, the output of GPR is a random variable. In this section, we will revisit two ways of interpreting GPR and both are equivalent. In section 3.1, we focus on the multidimensional input first, and then turn to the kernel method.

3.1 Weight-space View

Assumption 1. The *residue* follow an i.i.d Gaussian distribution.

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{x}^T\mathbf{w} \\ y_i - f(\mathbf{x_i}) &= \epsilon_i \\ \epsilon_i &\sim \mathcal{N}(0, \sigma_r^2) \end{aligned} \tag{4}$$

Assumption 2. In order to express our beliefs about the parameters before looking at the training set we need to specify a *prior* over $\mathbf{w}$ and assume $\mathbf{w}$ is independent of $\mathbf{x}$

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p) \tag{5}$$

Based on the Assumption 1 and 2, $\mathbf{y}$ becomes a sample of a random variable $\mathbf{y}$ whose PDF is:

$$p(\mathbf{y}|X,\mathbf{w}) = \prod_{i=1}^{n} p(y_i|\mathbf{x_i},\mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_r} exp(-\frac{{\epsilon_i}^2}{2\sigma_r^2}) \sim \mathcal{N}(X^T\mathbf{w}, \sigma_r^2\mathbf{I}) \tag{6}$$

Based on Theorem 3, Equation 6, and Assumption 2, we obtain the *posterior* $p(\mathbf{w}|X,\mathbf{y})$

$$p(\mathbf{w}|X,\mathbf{y}) = \frac{p(\mathbf{y}|X,\mathbf{w})p(\mathbf{w})}{\int p(\mathbf{y}|X,\mathbf{w})p(\mathbf{w})d\mathbf{w}} \sim \mathcal{N}(\frac{A^{-1}X\mathbf{y}}{\sigma_r^2}, A^{-1}) \tag{7}$$

where $A = \frac{XX^T}{\sigma_r^2} + \mathbf{\Sigma}_p^{-1}$. Using Equation 7 we update the estimation of $\mathbf{w}$ based on the knowledge of the training set. Then we use such posterior to estimate the distribution of $f(\mathbf{x}_*)$

$$p(f(\mathbf{x}_*)|\mathbf{x}_*,X,\mathbf{y}) = \int p(f(\mathbf{x}_*)|\mathbf{x}_*,\mathbf{w})p(\mathbf{w}|X,\mathbf{y})d\mathbf{w} \sim \mathcal{N}(\frac{{\mathbf{x}_*}^T A^{-1}X\mathbf{y}}{\sigma_r^2}, {\mathbf{x}_*}^T A^{-1}\mathbf{x}_*) \tag{8}$$

After applying the kernel function, Equation 8 becomes:

$$p(f(\mathbf{x}_*)|\mathbf{x}_*,X,\mathbf{y}) \sim \mathcal{N}(\frac{\phi(\mathbf{x}_*)^T A^{-1}\phi(X)\mathbf{y}}{\sigma_r^2}, \phi(\mathbf{x}_*)^T A^{-1}\phi(\mathbf{x}_*)) \tag{9}$$

where $A = \frac{\phi(X)\phi(X)^T}{\sigma_r^2} + \mathbf{\Sigma}_p^{-1}$ is of size $k \times k$, the invert of which may bring about huge computational load. To reduce the computational load, we can define $\phi_* = \phi(\mathbf{x}_*)$, $\phi = \phi(X)$, $K = k(\mathbf{x},\mathbf{x}) = \phi^T\mathbf{\Sigma_p}\phi, k(\mathbf{x}_*,\mathbf{x}) = \phi_*^T\mathbf{\Sigma_p}\phi, k(\mathbf{x}_*,\mathbf{x}_*) = \phi_*^T\mathbf{\Sigma_p}\phi_*, k(\mathbf{x},\mathbf{x}_*) = \phi^T\mathbf{\Sigma_p}\phi_*$ and rewrite Equation 9 in the following way:

$$\sigma_r^{-2}\phi(K + \sigma_r^2\mathbf{I}) = \sigma_r^{-2}\phi(\phi^T\mathbf{\Sigma_p}\phi + \sigma_r^2\mathbf{I}) = A\mathbf{\Sigma_p}\phi \Rightarrow A^{-1}\sigma_r^{-2}\phi = \mathbf{\Sigma_p}\phi(K + \sigma_r^2\mathbf{I})^{-1} \tag{10}$$

$$f(\mathbf{x}_*)|\mathbf{x}_*,X,\mathbf{y} \sim \mathcal{N}(k(\mathbf{x}_*,\mathbf{x})(K + \sigma_r^2\mathbf{I})^{-1}\mathbf{y}, k(\mathbf{x}_*,\mathbf{x}_*) - k(\mathbf{x}_*,\mathbf{x})(K + \sigma_r^2\mathbf{I})^{-1}k(\mathbf{x},\mathbf{x}_*)) \tag{11}$$

where $(K + \sigma_r^2\mathbf{I})^{-1}$ is of size $m \times m$ which is much smaller than $k \times k$. More details about the proof can be found in Rasmussen.

3.2 Function-space View

The index set of GP is the set of possible inputs $\mathbf{x}$. Based on 6, the output values of given $\{\mathbf{x_i}\}$ follows Gaussian distribution. Therefore, the linear regression result can be interpreted as a sample path of a GP and that's why we call this "function-space" view. In other words, both the training set and the test set are samples of the same GP, and GPR is to find such GP with the highest probability. Based on Definition 2.1.3 and Assumption 2, we denote $f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w}$, with prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_p)$:

$$m(\mathbf{x}) = \phi(\mathbf{x})^T\mathbb{E}[\mathbf{w}] = 0$$

$$k(\mathbf{x},\mathbf{x}') = \phi(\mathbf{x})^T\mathbb{E}[\mathbf{w}\mathbf{w}^T]\phi(\mathbf{x}') = \phi(\mathbf{x})^T\mathbf{\Sigma}_p\phi(\mathbf{x}') \tag{12}$$

However, the function-space view can express more than the white noise. We can assume the observations are noise free, and the *covariance function* can model the relationship between possible inputs. Therefore, we formalize the function-space view as:

$$\begin{bmatrix}\mathbf{f}(\mathbf{x}) \\ \mathbf{f}(\mathbf{x}_*)\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*)\end{bmatrix}\right) \tag{13}$$

where $K(X, X')$ is the covariance matrix between two multidimensional random variables. Using Theorem 2 , denoting $\mathbf{f}_* = \mathbf{f}(\mathbf{x}_*)$,$\mathbf{f} = \mathbf{f}(\mathbf{x})$, we can derive the predictive distribution:

$$\mathbf{f}_*|X_*, X, \mathbf{f} \sim \mathcal{N}\left(K(X_*, X)K(X, X)^{-1}\mathbf{f}, K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\right) \tag{14}$$

We can also add the noise term, the only change is that $K(X, X) \rightarrow K(X, X) + \sigma_r^2\mathbf{I}$ and the predictive distribution becomes:

$$\begin{aligned}\mathbf{f}_*|X_*, X, \mathbf{f} \sim \mathcal{N}(&K(X_*, X)(K(X, X) + \sigma_r^2\mathbf{I})^{-1}\mathbf{f} \\ &, K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_r^2\mathbf{I})^{-1}K(X, X_*))\end{aligned} \tag{15}$$

Equation 11 expresses the same idea with Equation 15. Equation 11 describes the predictive distribution on a single test point, while Equation 15 on the whole test set.

4 Algorithm Design

In this section, we will detail our training and testing algorithm. The details of using Cholesky decomposition to calculate the $(K(X, X) + \sigma_r^2\mathbf{I})^{-1}$ term and of forwarding and back-propagating the positive definite covariance functions are handled by the Gpytorch package.

4.1 Hyperparameters of kernel functions

Equation 15 didn't show the hyperparameters in GPR. The hyperparameters contain the kernel functions and the parameters of the kernel functions. Kernel functions include square-exponential kernels(SE), Matérn kernels, and Polynomial kernels etc. SE kernels have parameters σ^2, Matern kernels have parameters α and h, and Polynomial kernels have parameters m and c. Kanagawa et al. summarized those and other common kernels. There various optimization goals of the hyperparameters, and we choose to maximize the *marginal likelihood*.

Definition 4. Marginal likelihood refers to the marginalization over the function value $\mathbf{f}$:Rasmussen

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}) \tag{16}$$

The assumption declared in Section 3 can yield the *log marginal likelihood*:

$$log(p(\mathbf{y}|X)) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_r^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}log|K + \sigma_r^2\mathbf{I}| - \frac{n}{2}log2\pi \tag{17}$$

4.2 Train and test

The pseudo code of our algorithm is as follow. Algorithm 1 is the training process which will determine the hyperparameters of the kernel functions. Algorithm 2 uses the kernel functions to generate the predictive distribution. Since GPR only use linear operations, we use Gpytorch developed by Gardner et al. to implement our algorithm. The *Backpropagate* function can be implemented using Pytorch automatic differentiation mechanism Paszke et al.. The $Kernel_1$ and $Kernel_2$ are arbitrary kernel functions and we use fixed rules (unweighted summation) here. Gönen and Alpaydın also reviewed more complex rules. Table 1 shows all the kernels we use in this paper.

Table 1: Kernel functions and their parameters

Kernel	Kernel function	Parameters	Hyperparameters	Learnable
Linear	$v(\mathbf{x}^T\mathbf{x}')$	$v \in \mathbb{R}^+$		v
Matern	$\frac{1}{2^{\alpha-1}\Gamma(\alpha)} d^\alpha K_\alpha d$	$d = \frac{\sqrt{2\alpha}\|\|\mathbf{x}-\mathbf{x}'\|\|}{h}$, Γ is the gamma function and K_α is the modified Bessel Function of the second kind of order α	$\alpha = 2.5$	h
Polynomial	$(\mathbf{x}^T\mathbf{x} + \mathbf{c})^m$	$m \in \mathbb{N}$	$m = 3$	$\mathbf{c}$
RBF	$exp\left(-\frac{\|\|\mathbf{x}-\mathbf{x}'\|\|^2}{\gamma^2}\right)$	$\gamma > 0$		γ

Algorithm 1: An implementation of noise free GPR training using gradient descent

Data: $D = [(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)] = [X, Y], t$
Result: h_1, h_2
$k_1 \leftarrow Kernel_1(h_1)$; /* h_1, h_2 are parameters of kernels */
$k_2 \leftarrow Kernel_2(h_2)$; /* $Kernel_1, Kernel_2 \in \{RBF, Linear, Matern, Polynomial\}$ */
$iter \leftarrow 0$;
while $iter < t$ **do**
  $\mathbf{m} = calcMean(X)$; /* apply Equation 11 at each data point */
  $\mathbf{\Sigma} = calcVar(X, k_1 + k_2)$; /* apply Equation 11 at each data point */
  $\mathbf{y}' = MultivariateNormal(\mathbf{m}, \mathbf{\Sigma})$; /* return the mean and variance */
  $loss = -MLL(\mathbf{y}', Y)$; /* calculate marginal likelihood */
  $\sigma, l, v \leftarrow Backpropagate(loss)$;
end

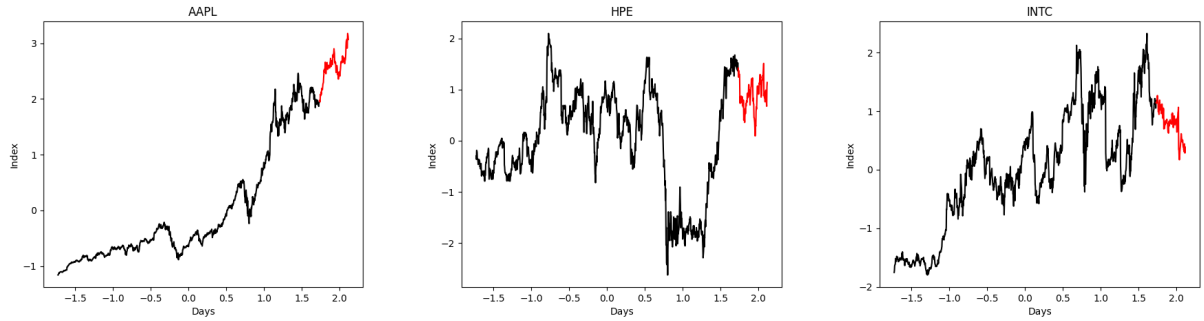
Algorithm 2: GPR testing

Data: $h_1, h_2, [x_{*1}, x_{*2}, ..., x_{*n}]$
Result: $\mathbf{m}_* = [m_{*1}, m_{*2}, ..., m_{*n}], \boldsymbol{\sigma}_*^2 = [\sigma_{*1}^2, \sigma_{*2}^2, ..., \sigma_{*n}^2]$
$k_1 \leftarrow Kernel_1(h_1)$;
$k_2 \leftarrow Kernel_2(h_2)$;
$\mathbf{m} = calcMean(X)$;
$\mathbf{\Sigma} = calcVar(X, k_1 + k_2)$;
$\mathbf{y}' = MultivariateNormal(\mathbf{m}, \mathbf{\Sigma})$;
$\mathbf{m}_* = GetMean(\mathbf{y}')$;
$\boldsymbol{\sigma}_*^2 = GetVar(\mathbf{y}')$;

5 Experimental Results

5.1 Raw data

We choose the daily adjusted close stock prices on of HPE, INTC and APPL from 2016-12-05 to 2021-12-03. Each enterprise has 1259 pieces of records. We choose the first 1134 records as the training set and the rest as the test set. We regularize the dataset by $x' = \frac{x-\mu_x}{\sigma_x}$, $y' = \frac{y-\mu_y}{\sigma_y}$. $x'_* = \frac{x_*-\mu_x}{\sigma_x}$, $y'_* = \frac{y_*-\mu_y}{\sigma_y}$. In fig 2, three training sets and test sets are plotted.



(a) AAPL (b) HPE (c) INTC

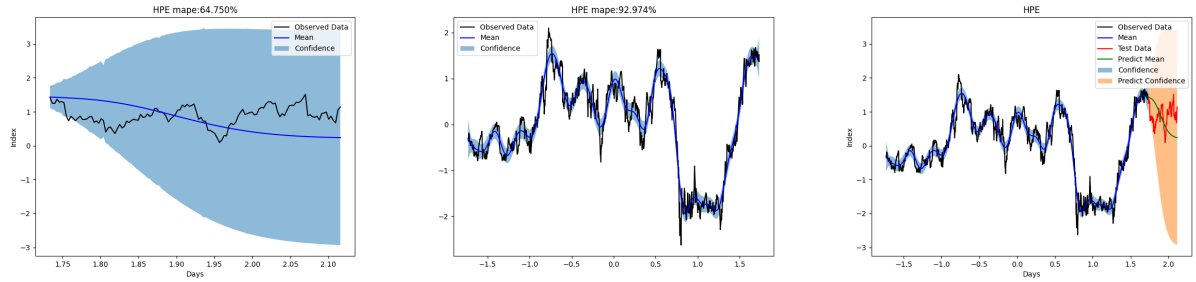
Figure 2: Training set (black line) and test set (red line) of three datasets

5.2 Results

We have 6 kernel function choices on each dataset and after trial and error we plot the best results. We use the *mean absolute percentage error*(MAPE) proposed by De Myttenaere et al. to measure the prediction error. In Fig 3, 4 and 5 We first plot the interpolate result on the training set, and then we plot the prediction result on the test set. Finally we combine both figures to show the trend. Model 2 and Model 3 predict the trend well, while the prediction of Model 1 is not correct. MAPE results are summarized in Table 2

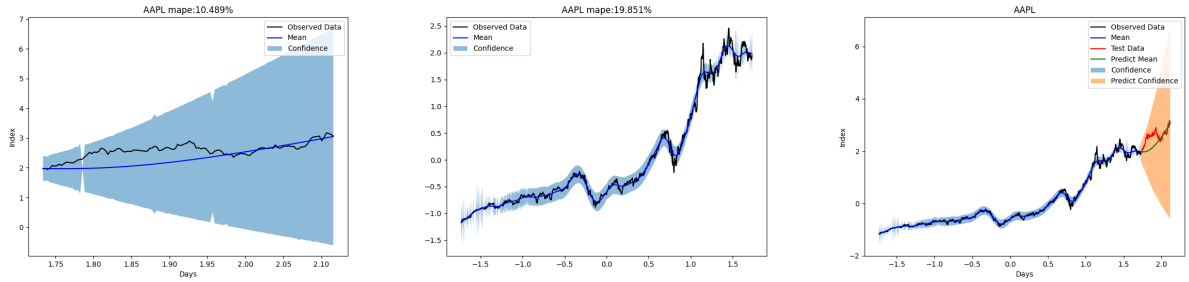
Table 2: MAPE of prediction and interpolation on three datasets

Dataset	Interpolation	Prediction	Kernels
HPE	92.974%	64.750%	Model 1
AAPL	19.851%	10.489%	Model 2
INTC	66.945%	41.279%	Model 3



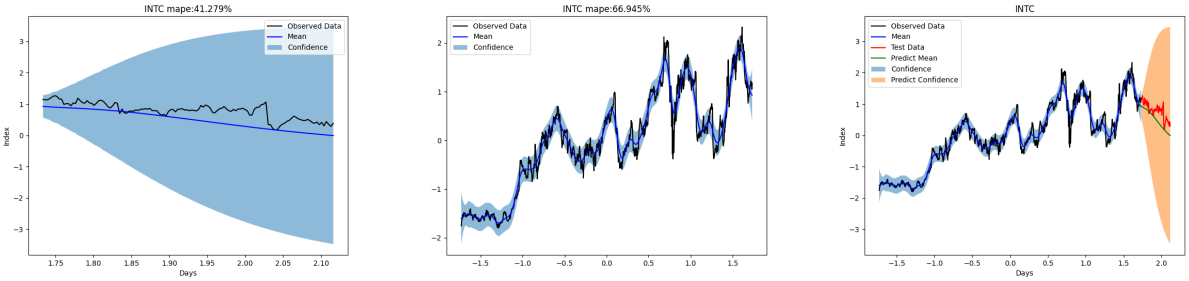
(a) Test results lr=0.3 iter=7 (b) Interpolate results lr=0.3 iter=7 (c) All results lr=0.3 iter=7

Figure 3: Model 1: RBF+Linear on HPE



(a) Test results lr=0.3 iter=7 (b) Interpolate results lr=0.3 iter=7 (c) All results lr=0.3 iter=7

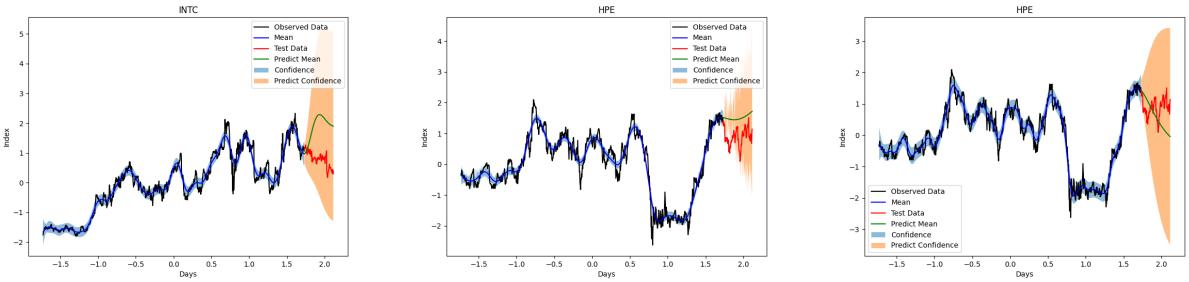
Figure 4: Model 2: Polynomial+Matern on AAPL



(a) Test results lr=0.3 iter=5 (b) Interpolate results lr=0.3 iter=5 (c) All results lr=0.3 iter=5

Figure 5: Model 3: RBF+Matern on INTC

We also apply a model trained on the training set of one dataset to the another dataset. As shown in Fig 6, we find that the model can make good prediction, which, however, is not always the case. Interpolate results are always good. Such phenomenon is very interesting, deserving further study.



(a) Model 1 on INTC (b) Model 2 on HPE (c) Model 3 on HPE

Figure 6: Apply one model to other datasets

6 Conclusion

In this paper, we review the theory basis of GPR and conduct a case study using three enterprises' stock price. To be specific, we revisit weight-space view and function-space view which are two

equivalent methods of deriving GPR. We use fixed rules to assign kernels and back-propagation to optimize the parameters of the kernels.

References

Peter Meer, Doron Mintz, Azriel Rosenfeld, and Dong Yoon Kim. Robust regression methods for computer vision: A review. *International journal of computer vision*, 6(1):59–70, 1991.

Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11): 4405–4423, 2020.

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Jiayu Qiu, Bin Wang, and Changjun Zhou. Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one*, 15(1):e0227222, 2020.

Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *arXiv preprint arXiv:1809.11165*, 2018.

Michael C Jensen, Fischer Black, and Myron S Scholes. The capital asset pricing model: Some empirical tests. *Studies in the Theory of Capital Markets*, 1972.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Andrew Ng. Cs229 lecture notes. *CS229 Lecture notes*, 1(1):1–3, 2000.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS 2017 workshop*, 2017.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

Arnaud De Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. Mean absolute percentage error for regression models. *Neurocomputing*, 192:38–48, 2016.