Multivariate Gaussian Process Regressions of the S&P 500 Equity Index

Rodney Greene, Ph.D.

Gaussian Process regression is a function approximation method that obtains the posterior distribution of a *clean process* (*i.e.*, with additive noise removed) conditional on observations that are distorted by additive noise. It is well suited for approximating financial market observables, like equity indices, as functions of other macroeconomic variables (*e.g.* other indices). In this quick empirical study I compare modeling the S&P 500 index daily close timeseries as a function of time (this form is the standard representation of a timeseries) and as a function of the VIX volatility index and the Federal Funds Effective Rate. I seek to illustrate the power of *multivariate Gaussian Process Regression* (**GPR**) by leveraging quantitative metrics (*i.e.* the MSE) as well as qualitative comparisons of power spectrum density spectra of the S&P 500 time series and the multivariate regressions. This document and the code developed for this exercise is available at *multivariate_GaussianProcessRegression*

The note begins with an overview of the GPR methodology with specific mappings to the *GPyTorch* Python package. Following is a description of the data, which is sourced from the U.S. Federal Reserve Bank of St. Louis. The next section demonstrates the power of the *GPyTorch* Python package for implementing multivariate GPR of the S&P 500 index. Hyperparameter optimization is not addressed in this note and is reserved for a later submission.

1 Gaussian Process Regression Primer

The classic reference for Gaussian Processes is Ref [3]. Consider a scalar-valued process driven by a time-dependent vector, $f(\boldsymbol{x})$, corrupted by additive i.i.d. noise, ϵ. This noise is usually modeled as $\mathcal{N}(0, \sigma_n^2)$. The n observations of this corrupted signal are $\{y_i \in \boldsymbol{y}(\boldsymbol{x}), \boldsymbol{x}_i \in \boldsymbol{X}, i \in [1, \ldots, n]\}$. The GP *ansatz* states the prior distribution for these observations is a zero-mean multivariate Gaussian, $p(\boldsymbol{y}(x)) = \mathcal{N}\big(\boldsymbol{0}, \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})\big)$. The covariance of the observations $\boldsymbol{y}(\boldsymbol{X})$ is written as a kernel matrix-valued function $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$. An essential component of GPR engineering is selection of a suitable kernel. In what follows I will compare the performance of the kernels: the *squared exponential* (also called the *radial basis function*) and the *spectral mixture* kernels.

The regression problem is to estimate the predictive posterior distribution of the uncorrupted process (evaluated at test points, $\boldsymbol{f}_*(\boldsymbol{X}_*)$) conditional on the observations of the corrupted signal, $\boldsymbol{y}(\boldsymbol{X})$. The first step is to apply the GP *ansatz* to write the GP prior for the observations and test points. We obtain

$$\begin{bmatrix} \boldsymbol{y}(\boldsymbol{X}) \\ \boldsymbol{f}_*(\boldsymbol{X}_*) \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix}\right) \tag{1}$$

Upon applying the classical statistical conditioning and marginalization [1] (specifically, conditioning on the observations) one obtains the *predictive posterior* as

$$p\big(\boldsymbol{f}_*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{X}_*\big) = \mathcal{N}\big(\langle\boldsymbol{f}_*\rangle, \boldsymbol{cov}(\boldsymbol{f}_*)\big), \text{where} \tag{2}$$

$$\langle\boldsymbol{f}_*\rangle = \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X})\big[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}+\sigma_n^2\boldsymbol{I}\big]^{-1}\boldsymbol{y}(\boldsymbol{X}) \tag{3}$$

$$\boldsymbol{cov}(\boldsymbol{f}_*) = \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X})\big[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma_n^2\boldsymbol{I}\big]^{-1}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}_*) \tag{4}$$

The *GPyTorch* Python package is a GPR implementation that supports GPU acceleration, which is useful for calculating the computationally intensive matrix inversions in the calculation of the predictive posterior. The examples below do not explicitly use the GPU functionality but instead focus on the quality of the S&P 500 index regression as the number of regressors is increased and the covariance kernel is changed. The *gPyTorch.models* class is a representation of the GP prior, $p(\boldsymbol{f}(\boldsymbol{x}))$. It supports a generalization to non-zero mean, although that flexibility is rarely employed in practice. The GP prior is constructed with a user-specified covariance kernel function.

The class *gpytorch.likelihoods* is a representation of the predictive posterior, *i.e.*,

$$likelihood(model(\boldsymbol{X})) = p(\boldsymbol{f}_*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{X}_*).$$

2 The Financial Data

The data is sourced from the U.S. Federal Reserve Bank of St. Louis' *FRED* website, the publicly accessible economic data warehouse of the U.S. central bank. The *fredpy* Python package has a simple API for downloading macroeconomic timeseries from the FRED site into pandas dataframes. The user needs a license key string to download from FRED. The key may be obtained at no-cost from *here*.

Table 1 summarizes the time series employed for this study.

Table 1: FRED Data

Symbol	Description	Training Domain
SP500	Dependent variable. Daily closing levels of the Standard and Poor's 500 equity index.	1 March 2021 - 30 June 2023
VIXCLS	Driver variable. Daily closing levels of the Chicago Board Options Exchange S&P 500 index option volatility index.	
DFF	Driver variable. Daily closing level of the interest rate banks trade federal funds overnight.	

3 Multivariate Kernels

The two covariance kernels employed in this exercise are the squared-exponential (also called the radial basis function) and the spectral mixture kernel function, which take the forms

[1]See Appendix A.2 of Ref. [3].

below. The squared-exponential kernel decrease as the distance between the $\boldsymbol{x}'s$ increase. The diagonal $\boldsymbol{\Theta}^{-2}$ matrix contains characteristic length scale factors for each driver variable. So as points become less similar (in other words, are further apart) their covariance decreases over the corresponding length scale.

$$K_{RBF}(\boldsymbol{x}_1, \boldsymbol{x}_2) = scale \times \exp\left(-\frac{1}{2}\big(\boldsymbol{x}_1 - \boldsymbol{x}_2\big)^T \boldsymbol{\Theta}^{-2}\big(\boldsymbol{x}_1 - \boldsymbol{x}_2\big)\right) \tag{5}$$

The spectral mixture kernel (see Ref. [2]) was motivated by the desire to first model the desired frequency response of a kernel, then transform it to the spatial domain via the inverse Fourier transform. The diagonal Σ matrix contains length scale factors for decay of the Q frequency components. Each component frequency is characterized by the $\boldsymbol{\mu}$ vector. Explicitly,

$$K_{SM}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{q=1}^{Q} w_q \exp\left(-\frac{1}{2}\big(\boldsymbol{x}_1 - \boldsymbol{x}_2\big)^T \boldsymbol{\Sigma}_q\big(\boldsymbol{x}_1 - \boldsymbol{x}_2\big)\right)\cos\Big(\boldsymbol{\mu}_q^T\big(\boldsymbol{x}_1 - \boldsymbol{x}_2\big)\Big) \tag{6}$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$, $\boldsymbol{\Sigma}_q = diag\big(\sigma_1^{(q)}, \ldots, \sigma_n^{(q)}\big)$, and $w_q, \mu_q \in \mathbb{R}_+$.

As will be demonstrated in the next section, the squared-exponential kernel captures low-frequency components of the S&P 500 time series, whereas a five-component spectral mixture kernel captures both low and high frequency components.

Finally I note that in order to limit the focus of this little note, no *training* is performed herein. In other words, the kernel hyperparameters are not calculated by solving an optimization problem. Normally, the hyperparameters are obtained by maximizing the marginal likelihood.[2] The gPyTorch package has a robust optimizer. Nevertheless, as the focus of this note is multivariate regression and not the training procedure, we will save the model selection / training discussion for a later submission.

4 In-Sample Performance Comparison of Multivariate Regressions

The GP regressions of the S&P 500 levels observed over the 1 March 2021 - 30 June 2023 training period are performed as shown in Table 2

Table 2: Regressions Under Study

ResponseVariable	**Driver Variables**
S&P 500	Tenor
	Tenor, VIXCLS
	Tenor, VIXCLS, DFF

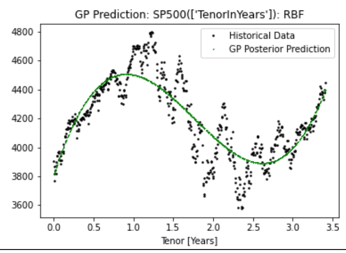
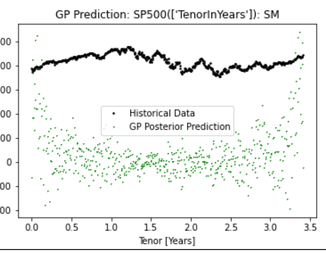
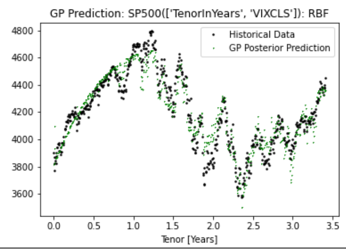
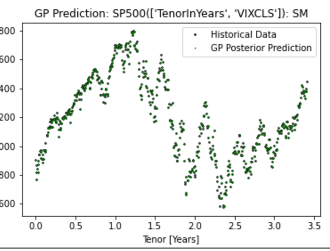
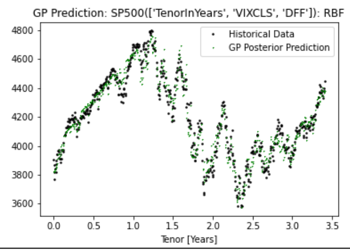
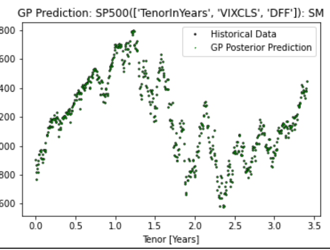
[2]For this exercise, the squared-exponential hyperparameters are obtained manually via trial and error. The spectral mixture hyperparameters are calculated from the data by the method gpytorch.kernels.SpectralMixtureKernel.nitialize_from_data .

Each row of the table is executed using both the squared-exponential and spectral mixture kernels, respectively. The *Tenor* driver is simply the years to the respective observation date with respect to 1 Mar 2021, the start of the training domain. Hence the first row represents the simplest univariate GP time series regression with respect to time.

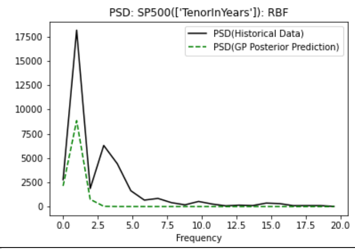
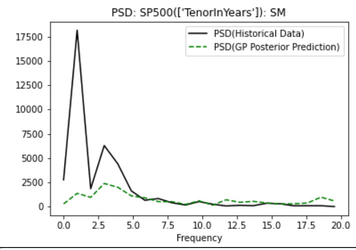
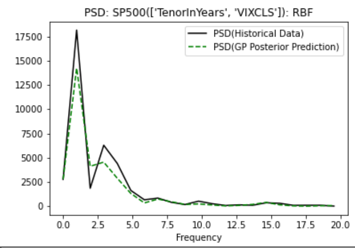
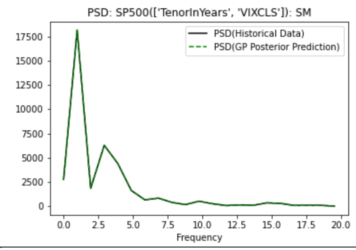
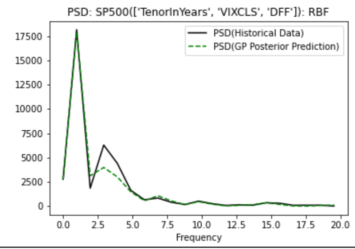
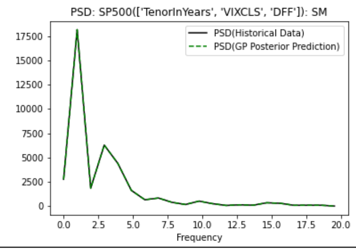
Table 3 summarizes the quantitative comparisons of the regressions. The *mean-squared error* is calculated using the standard *sklearn.metrics.mean_squared_error method* . In all cases, as the number of driver variables is increased the MSE decreases. This result is expected because as the regressions contain more economic information we expect they will better model the S&P 500 dynamics. The table also shows that very little frequency information is captured in the univariate regressions. As a result, the univariate SM regression is quite bad. The univariate squared-exponential regression only captures the very smooth, slowly varying sinusoidal pattern in the historical S&P 500 time series. These observations are clearly confirmed in the plots of Table 4. Here the predicted prior mean, $\langle \boldsymbol{f}_* \rangle$, are plotted with the historical time series. Both the squared-exponential and spectral mixture multivariate regressions replicate the historical data quite well by eye. The spectral mixture regressions have excellent agreement.

Table 3: Training-Domain MSE Results

	Mean-Squared Error	
Driver Variables	**Squared-Exponential Kernel**	**Spectral Mixture Kernel**
Tenor	19747	14451295
Tenor, VIXCLS	8200	~ 0
Tenor, VIXCLS, DFF	3331	~ 0

Table 4: Regression Comparisons to Historical S&P 500 Time Series		
Driver Variables	**Squared-Exponential Kernel**	**Spectral Mixture Kernel**
Tenor		
Tenor, VIXCLS		
Tenor, VIXCLS, DFF		

Finally Table 5 shows the power spectrum density spectra (as calculated by scipy.signal.welch) of the historical time series and the regressions. Clearly the multivariate GP regressions obtain excellent agreement with the S&P 500 spectra. Recall the spectral mixture kernel was designed with the kernel frequency response as a starting point. Hence its superior frequency agreement with the S&P 500 is expected.

Table 5: Power Spectrum Density Comparisons		
Driver Variables	**Squared-Exponential Kernel**	**Spectral Mixture Kernel**
Tenor		
Tenor, VIXCLS		
Tenor, VIXCLS, DFF		

For completeness, I note the covariance of the predictive posteriors, $\boldsymbol{cov}(\boldsymbol{f}_*)$, for all these regressions is quite small. This result suggests more work needs to be done to properly capture the noise structure of the response and driver variables.

5 Conclusion

The *GPyTorch* package has been employed to demonstrate the power of multivariate Gaussian Process regression to model volatile financial time series. In addition, the superior frequency response of the spectral mixture covariance kernel with respect to the squared - exponential kernel has been confirmed. I hope the code repository will facilitate more investigations of multivariate GPR in the finance domain.

References

[1] *GPyTorch*

[2] Parra, Gabriel and Felipe Tobar (2017) *Spectral Mixture Kernels for Multi-Output Gaussian Processes*, 31st Conference on Neural Information Processing Systems (NIPS2017), Long Beach, CA, USA.

[3] Rasmussen, Carl E. and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning*, MIT Press.