Gaussian Processes for Financial Forecasting

John Goodacre
Supervisors: John Shawe-Taylor, Ahrash Daneshvar, Dariush Hosseini

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Science

in

Machine Learning.

Department of Computer Science

University College London

September 1, 2017

I, John Goodacre, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Code for this project, barring trade implementation code, can be found in the following public repository:
https://github.com/jg8610/multi-task-project/

Abstract

This dissertation investigates the use of Gaussian Processes (GPs) for forecasting mean-reverting time series with underlying structure, using fairly unexplored functional and augmented data structures.

Whereas many traditional forecasting techniques focus on the short-term dynamics of a time series. GPs offer the promise of forecasting not only the mean prediction of a time series but the whole probability distribution over an entire future trajectory. This offers advantages within the financial arena, where one may have a correct prediction but still lose out due to incorrect volatility assessments wiping out capital, and further advantages in trade selection whereby multiple Sharpe ratios can be forecast adjusted for round trip transaction costs when considering trade decisions.

The functional data representation used in this thesis enables longer term predictions by 'borrowing' information from prior years even as one moves further from the training data in the current year. The augmented representation enhances the training set by adding multiple training targets for multiple points in the future, again encouraging long term predictions.

The implementation closely follows [7], who tested efficacy on commodity futures. We vary when it comes to testing. We instead use simulated data with similar underlying characteristics. We build a test bed to stress both the data representations and models in the presence of increasing noise, fat tails and inappropriate

kernels - all likely in reality. By simulating we have the advantage of comparing our forecast distribution through time, against a full simulation of the actual distribution of our test set though time, removing the typical uncertainty inherent in testing the efficacy of time series models on real data. We enable feature prediction through augmentation, and use sub-sampling to make this viable for GPs.

The experiments showed the efficacy of the functional and augmented data representations, quantified the impact of noise and fat tails on these models and showed where simpler models as used by practitioners suffice. We explored the impact of the wrong initial kernel choice and showed how functional-augmentation can in some circumstances mitigate that impact. We showed how augmentation can improve predictive capability where only a short training series is available and finally we showed innovative uses of augmented Gaussian Processes for trading real life exchange traded futures.

Acknowledgements

Thanks to John Shawe-Taylor, Ahrash, and Dariush for their help and direction. John in particular thanks for your time and caring supervision. Ahrash, it was great to discuss all this stuff with you and felt like it ended just when it was about to get interesting. Dariush, thanks so much for your time and ideas to improve the project. Also thanks to Zais Group, Glen and Andreas in particular and for the use of their cloud computing resources.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we give a brief introduction to the problem being discussed, and motivation behind it. Then we lay out the questions being asked and finally, we outline the structure of the thesis.

1.1 Motivation

This thesis will investigate the efficacy of Gaussian Processes (GPs) for regression of both simulated financial processes and real life financial data. We closely follow [7, 1], who apply less familiar data representations as inputs to GPs in order to predict mean reverting commodity futures and equities respectively.

Chapados and Bengio [7] reported positive test statistics and trading performance applied to real commodity futures, however in real life time series we essentially have one realisation of a noisy process. One cannot go back in time and press replay. With enough data and assuming the underlying process is unchanging then we can gain confidence. But in the finance domain there is always a nagging doubt that one's backtests may be less successful in real life.

Motivated by the ability to perform a longer term forecast of an entire trajectory from [7]. We explore the efficacy of these data representations and models on simulated data with similar characteristics to real financial data, but where we can control the test set parameters. Finally, we apply a simulation to create the full

distribution of the test set.This gives us the ability to test whether the GP really did learn the full distribution of the noisy time-series.

Given this, we move on to real life financial time series and show how GPs can be used in a real trading strategy and the extra benefits enabled by having a mean function forecast as well as the uncertainty about ones forecast, and how this can be implemented directly into trade decision making.

1.2 Trajectory Prediction

For many time series models we are constrained to forecast over a fixed horizon, where short term dynamics may be captured but where the prediction degrades to the unconditional expectation of the process longer term.

Other forms of regression commonly used within finance apply a form of conditional expectation, asking the question ”What will the value of my target ’y’, be under the hypothesis that my features have the values ’X’?”.

In this thesis we are interested in attempting to predict an entire trajectory going forwards using Gaussian Process regression. Furthermore we look at the circumstances where this prediction can be improved beyond a standard auto-regressive model or indeed a standard (one dimensional Gaussian Process) through the functional and augmented data representations as used by [7, 1].

1.3 Research

Given an implementation of GPs, the various data representations, and a simulated time series. We train our different models on this time series and attempt to predict the distribution going forwards.

Because we have simulated data, at the point of prediction we can rerun our time series from this point multiple times, in order to create the full distribution of the test set. This gives us both a predictive and test distribution for this single

training example. We do this for multiple training examples.

This enables us to examine the mean squared error of the mean prediction and the standard deviation prediction for several points in the future in order to assess the predictive ability of the model.

We begin with a time series that has obvious structure, then add an Ornstein Uhlenbeck (OU) process to this structure. In [7] the claim was that under the functional representation information could be borrowed from previous years as the GP moves further away from the training data. By giving the data structure we can assess this. However life gets more difficult as the noise in the OU process is increased.

The first part of the research looks at how these approaches compare to one another and to the standard mean reverting auto-regressive models commonly used in finance, in the presence of increasing noise.

The second part of the research makes life harder still for the GPs and closer to real life with fatter tailed non gaussian noise and the prospect of being unlucky enough to have chosen an inappropriate kernel prior. Again we examine the performance of these more complex models to each other and an auto-regressive model.

The next part of the research briefly examines the impact of having little training data on the models and shows how augmentation can help.

The final part of the research moves on to a real life problem of using GP's to trade exchange traded funds and shows how having not just a prediction of the mean function, but also the uncertainty about that prediction can be directly implemented into a trading strategy.

1.4 Structure of the Thesis

In Chapter 2, we present the regression problem, starting from linear regression, moving onto Bayesian regression and finally to Gaussian Processes. A key aspect of a Gaussian Process is its Kernel, which we also discuss. Finally, we discuss some practical issues as well as automatic relevance determination, which we shall use later on.

In Chapter 3, we present the functional, augmented and functional-augmented data representations, discussing 'observation time' and 'target time', as well as our scaling method for implementing augmentation. We then go on to discuss the Ornstein Uhlenbeck process and the traditional methods of calibrating this, a rather hard to beat 'straw man' model, as we shall see. In Chapter 4, we lay out our experiments and the results for each. In Chapter 5, we finish by discussing our conclusions, and potential areas for further work.

Chapter 2

Background

2.1 Gaussian Processes

In this chapter we lay out the foundations, describing how to construct Gaussian Processes (GPs), their kernels, and how to train and optimise GPs. In addition we describe automatic relevance determination, a method that can aid in assessing feature relevance in an analogous way to L1 regularisation or the Lasso for standard regression problems. Varying presentations of these sections can be found in standard textbooks such as [2, 18, 23, 27], this exposition closely follows both lectures and notation at Cambridge and UCL. There are no claims of originality.

2.1.1 From Linear Regression to Bayesian Regression

When we wish to predict a continuous value given a set of observations then we are in a regression setting. Possibly the simplest form of regression one can perform is Linear Regression.

A regression is often assumed to comprise two parts. A systematic variation which could be accurately predicted if the underlying process is known, $f(\mathbf{x})$ and a random variation reflecting the inherent lack of predictability of our system.

$$y = f(\mathbf{x}) + \varepsilon \quad \text{,where } \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{2.1}$$

2.1.2 Linear Regression

Below I give a very brief exposition of Linear regression, a necessary starter if one is unfamiliar with GPs.

If we consider a set of training data, with n samples, where for each observation $\mathbf{x}_i$ we have an output y_i. Then a regression function $f(x)$ is linear if it can be written as

$$f(\mathbf{x}) = \mathbf{w}'\mathbf{x} \tag{2.2}$$

where the target values have gaussian noise such that:

$$y = f(\mathbf{x}) + \varepsilon \quad \text{,with } \varepsilon \sim \mathcal{N}(0, \sigma_n^2)$$

Given $\mathbf{w}$ and assuming independence the likelihood can be written as

$$\Pr(y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}) = \prod_{i=1}^{n} \Pr(y_i | \mathbf{x}_i, \mathbf{w}) = \mathcal{N}(\mathbf{y}; \mathbf{X}'\mathbf{w}, \sigma_n^2 \mathbf{I}) \tag{2.3}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix and $\mathbf{X} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_n\right]$, $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$

The values for $\mathbf{w}$ can be found by maximising the likelihood or equivalently minimising the least squares cost.

$$\min_{\mathbf{w}} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{w}'\mathbf{x}_i)^2 \right\} = \min_{\mathbf{w}} \left\{ \|\mathbf{y} - \mathbf{X}'\mathbf{w}\|^2 \right\}$$

We take the partial derivative with respect to $\mathbf{w}$ and set equal to zero, giving the solution

$$\hat{w} = \left(\mathbf{X}\mathbf{X}'\right)^{-1}\mathbf{X}\mathbf{y} \tag{2.4}$$

Note that the above solution for $\hat{w}$ is a single point estimate for the model pa-

rameters. However, in reality the training data is only one possible set of samples of training data each of which could give slightly different answers for the maximum likelihood. Note also that if we have prior beliefs about $\hat{w}$, this is not reflected in our solution.

The Bayesian methodology generalises the above in that rather than a point estimate, we consider a distribution over the parameters. In which case the distribution of our prediction is given as

$$\Pr(y|\mathbf{x},\mathbf{y},\mathbf{X}) = \int \Pr(y|\mathbf{x},\mathbf{w}) \Pr(\mathbf{w}|\mathbf{y},\mathbf{X}) \mathrm{d}\mathbf{w} \tag{2.5}$$

where $\Pr(\mathbf{w}|\mathbf{y},\mathbf{X})$ is the distribution over the parameters given our training data and where we have treated $\mathbf{w}$ as a latent variable and integrated out.
Given the training data $\mathbf{y}$ and $\mathbf{X}$ we can write the distribution of the weights using bayes rule

$$\Pr(\mathbf{w}|\mathbf{y},\mathbf{X}) = \frac{\Pr(\mathbf{y}|\mathbf{X},\mathbf{w})}{\Pr(\mathbf{y}|\mathbf{X})} \tag{2.6}$$

Therefore $\Pr(\mathbf{w}|\mathbf{y},\mathbf{X}) \propto \Pr(\mathbf{y}|\mathbf{X},\mathbf{w}) \Pr(\mathbf{w})$. By giving a Gaussian prior to $\mathbf{w}$ we have that $\Pr(\mathbf{y}|\mathbf{X},\mathbf{w}) = \mathcal{N}(\mathbf{y};\mathbf{X}'\mathbf{w},\sigma_n^2\mathbf{I})$, with $\mathbf{w} \sim \mathcal{N}(0,\sigma_w^2\mathbf{I})$.
The weights distribution can now be written in the following form

$$\log(\Pr(\mathbf{w}|\mathbf{y},\mathbf{X})) \propto -\frac{1}{2\sigma_n^2}(\mathbf{y}-\mathbf{X}'\mathbf{w})'(\mathbf{y}-\mathbf{X}'\mathbf{w}) - \frac{1}{2\sigma_w^2}\mathbf{w}'\mathbf{w}$$

Given that both are gaussians then a standard derivation gives

$$\Pr(\mathbf{w}|\mathbf{y},\mathbf{X}) = \mathcal{N}(\mathbf{w};\mu_w,\Sigma_w) \tag{2.7}$$

$$\text{with, } \mu_w = \frac{1}{\sigma_n^2}\Big(\frac{1}{\sigma_n^2}\mathbf{X}\mathbf{X}' + \frac{1}{\sigma_w^2}\mathbf{I}\Big)^{-1}\mathbf{X}\mathbf{y}, \quad \text{and} \quad \Sigma_w = \Big(\frac{1}{\sigma_n^2}\mathbf{X}\mathbf{X}' + \frac{1}{\sigma_w^2}\mathbf{I}\Big)^{-1}$$

where intuitively we see that as $\sigma_w^2 \to \infty$ then we recover our max-likelihood estimate.

For noiseless prediction,

$$\Pr(f(\mathbf{x})|\mathbf{x},\mathbf{X},\mathbf{y}) = \int \Pr(f(\mathbf{x})|\mathbf{x},\mathbf{w})\Pr(\mathbf{w}|\mathbf{X},\mathbf{y})\mathrm{d}\mathbf{w} = \mathcal{N}(f(\mathbf{x});\mathbf{x}'\mu_w,\mathbf{x}'\Sigma_w\mathbf{x})$$

and finally for noisy prediction we have that,

$$\Pr(y|\mathbf{x},\mathbf{X},\mathbf{y}) = \int \Pr(y|f(\mathbf{x}))\Pr(f(\mathbf{x})|\mathbf{X},\mathbf{y},\mathbf{x})\mathrm{d}\mathbf{w} = \mathcal{N}(y;\mathbf{x}'\mu_w,\mathbf{x}'\Sigma_w\mathbf{x}+\sigma_n^2) \tag{2.8}$$

2.1.3 Basis Functions

One obvious limitation of the approach thus far comes from the word linear. That is we require a linear relationship to exist between the observations and the target values. However we can easily expand the hypothesis space of our regression by using some non-linear basis function $\phi()$. In particular we may rewrite our mapping as

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i\phi(\|\mathbf{x}-\mathbf{x}_i\|) = \phi(\mathbf{x})'\mathbf{w} \tag{2.9}$$

where $\phi(\mathbf{x}) = [\phi(\|\mathbf{x}-\mathbf{x}_1\|)\ldots\phi(\|\mathbf{x}-\mathbf{x}_n\|)]'$ and again $y = f(\mathbf{x})+\varepsilon$

This is now a non-linear regression but of course linear in the parameters. For our training data we have that $\mathbf{y} = \Phi\mathbf{w}+\varepsilon$, where

$$\Phi = \begin{bmatrix} \phi(\|\mathbf{x}_1-\mathbf{x}_1\|) & \cdots & \phi(\|\mathbf{x}_1-\mathbf{x}_n\|) \\ \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_n-\mathbf{x}_1\|) & \cdots & \phi(\|\mathbf{x}_n-\mathbf{x}_n\|) \end{bmatrix} = \begin{bmatrix} \phi(\mathbf{x}_1)' \\ \vdots \\ \phi(\mathbf{x}_n)' \end{bmatrix} \tag{2.10}$$

In the case of maximum likelihood then assuming an inverse exists then $\hat{\mathbf{w}} = \Phi^{-1}\mathbf{y}$.

Going bayesian, we follow exactly the same analysis as we have just done. i.e. let the prior distribution of our weight vector be gaussian $\Pr(\mathbf{w}) = \mathcal{N}(\mathbf{w};0,\sigma_w^2\mathbf{I})$ and to find our posterior weight distribution we have that $\Pr(\mathbf{y}|\mathbf{X},\mathrm{w}) = \mathcal{N}(\mathbf{y};\Phi\mathbf{w},\sigma_n^2\mathbf{I})$.

Therefore

$$\Pr(\mathbf{w}|\mathbf{X},\mathbf{y}) = \mathcal{N}(\mathbf{w};\mu_w,\Sigma_w), \text{ where}$$

$$\mu_w = \tfrac{1}{\sigma_n^2}\left(\tfrac{1}{\sigma_n^2}\Phi'\Phi + \tfrac{1}{\sigma_w^2}\mathbf{I}\right)^{-1}\Phi'\mathbf{y}, \quad \text{and} \quad \Sigma_w = \left(\tfrac{1}{\sigma_n^2}\Phi'\Phi + \tfrac{1}{\sigma_w^2}\mathbf{I}\right)^{-1} \tag{2.11}$$

2.1.4 From Bayesian Regression to Gaussian Processes

Many explanations of GPs appear to lead back to Rasmussen and Williams [27] including the Chapados paper [6] that this thesis tracks fairly closely. This explanation follows that trend, and is not original but based upon [6, 27].

A Gaussian Process is a probability distribution over functions. It requires any finite set of function values $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2),\ldots,f(\mathbf{x}_N)]$ to have a joint Gaussian distribution. For finance types with an aversion to the word Gaussian, it is important to stress the flexibility of GPs, where due to their non-parametric nature one effectively fits a basis function to every single point, but where over-fitting can be avoided in many circumstances (with common sense) through their bayesian treatment. However there are extensions such as t-processes [31].

Before the GP is conditioned on training data it is completely characterised by its mean and covariance function, without loss of generality we can often assume a zero mean function. The covariance function or kernel determines the type of structure the GP can learn and also how well it generalises to new data.

If we let $\mathbf{x}$ index into the real process $f(\mathbf{x})$. We write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot)) \tag{2.12}$$

where functions $m(\cdot)$ and $k(\cdot,\cdot)$ are, respectively, the mean and covariance functions

$$m(\mathbf{x}) = E[f(\mathbf{x})], \;\; k(\mathbf{x}_1,\mathbf{x}_2) = E[(f(\mathbf{x}_1) - m(\mathbf{x}_1))(f(\mathbf{x}_2) - m(\mathbf{x}_2))]. \tag{2.13}$$

Again we assume that we are given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1,\ldots,N\}$ with

$\mathbf{x}_i \in R^D$ and $y_i \in R$. If $\mathbf{X}$ is our matrix of training inputs and $\mathbf{y}$ the vector of targets and as before assume that each y_i is a noisy measurement from $f(\mathbf{x})$

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \text{where } \varepsilon_i \sim \mathcal{N}(0, \sigma_n^2) \tag{2.14}$$

If we assume the GP prior has a mean of zero, then

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\cdot,\cdot)) \tag{2.15}$$

and using our definition of $\mathbf{f}$ from before then the prior distribution is given by

$$\mathbf{f} \sim \mathcal{GP}(0, K(\mathbf{X},\mathbf{X})) \tag{2.16}$$

$$\text{where } K(\mathbf{X},\mathbf{X}) = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & \cdots & k(\mathbf{x}_n,\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_1,\mathbf{x}_n) & \cdots & k(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}$$

This Gram matrix needs to be positive semi-definite to be a valid kernel. If we now move from the training set to an additional set of test point, given by $\mathbf{X}_*$ with unknown function values $\mathbf{f}_*$, then the joint prior is given by

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) & K(\mathbf{X},\mathbf{X}_*) \\ K(\mathbf{X}_*,\mathbf{X}) & K(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix}\right) \tag{2.17}$$

We can now show the predictive distribution for the test points. Using bayes theorem the joint posterior given our training data is

$$\Pr(\mathbf{f},\mathbf{f}_*|\mathbf{y}) = \frac{\Pr(\mathbf{y}|\mathbf{f},\mathbf{f}_*)\Pr(\mathbf{f},\mathbf{f}_*)}{\Pr(\mathbf{y})} = \frac{\Pr(\mathbf{y}|\mathbf{f})\Pr(\mathbf{f},\mathbf{f}_*)}{\Pr(\mathbf{y})} \tag{2.18}$$

Where $\Pr(\mathbf{y}|\mathbf{f},\mathbf{f}_*) = \Pr(\mathbf{y},\mathbf{f})$ since the likelihood is conditionally independent of $\mathbf{f}_*$ given $\mathbf{f}$, and

$$\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 I_N) \tag{2.19}$$

I_N being the $N \times N$ identity matrix. We obtain our predictive distribution by

marginalising out the training set latent variables,

$$\Pr(\mathbf{f}_*|\mathbf{y}) = \int \Pr(\mathbf{f},\mathbf{f}_*|\mathbf{y})\mathrm{d}\mathbf{f} = \frac{1}{Pr(\mathbf{y})}\int \Pr(\mathbf{y}|\mathbf{f})\Pr(\mathbf{f},\mathbf{f}_*)\mathrm{d}\mathbf{f} \tag{2.20}$$

These are all Gaussian, thus the result of the normalised marginal is also Gaussian, and can be shown to have mean and covariance given by

$$\begin{aligned} E[\mathbf{f}_*|\mathbf{y}] &= K(\mathbf{X}_*,\mathbf{X})\Lambda^{-1}\mathbf{y}, \\ \mathrm{Cov}[\mathbf{f}_*|\mathbf{y}] &= K(\mathbf{X}_*,\mathbf{X}_*) - K(\mathbf{X}_*,\mathbf{X})\Lambda^{-1}K(\mathbf{X},\mathbf{X}_*) \end{aligned} \tag{2.21}$$

$$\text{where } \Lambda = K(\mathbf{X},\mathbf{X}) + \sigma_n^2 I_N$$

Notice the similarities with the basis function expansion, except this time we are using kernels and our space is infinite dimensional. Notice also that we have a matrix inversion with Λ^{-1}. At present this is perhaps the achilles heel of GPs, where compared to say Deep Learning we have scaling issues in the number of training examples. Indeed the matrix inversion requires $O(N^3)$ time and $O(N^2)$ space. Note in practice rather than an actual inversion, one might prefer to solve a linear system or use a Cholesky decomposition due to issues of numeric stability.

An attractive natural outcome of GP regression is an expression for not only the expected value at the test points but also the full covariance matrix between these points.

2.1.5 Covariance functions

Kernel methods are both broadly applicable and a large subject in their own right. They have applications in regression problems such as Kernel ridge regression and of course Gaussian process regression. Classification through Support Vector machines and a myriad of other areas, together with a considerable body of theoretical results. From the point of view of this thesis suffice to say that the covariance function is absolutely key to GPs and I will only mention kernels in the context of Gaussian Process regression as a covariance function prior, with hyper-parameters to be optimised. Further references on kernel methods themselves can be found in

Shawe-Taylor and Scholkopf [32, 30].

2.1.5.1 Similarity

A central idea in machine learning is the concept of 'similarity', in short, points **x** and **x**' that are 'close', should also have similar target values y and y'. It is precisely the covariance function that defines this concept of similarity in the case of Gaussian Process regression. The covariance function acts as our prior and sets out the space of functions that can be learned. An incorrect kernel prior can thus lead to poor generalisation.

Because we can construct kernels through combinations of other kernels, we can lend considerable structure to our covariance function, which in turn can express considerable structure in its predictions post optimisation as we show later.

In terms of GP workflow, we choose a covariance function prior, and fit the model to our training data. Which we can then use to predict. The covariance function itself has hyper-parameters which may be optimised for better generalisation.

2.1.5.2 Examples of Covariance functions applied to Mauna Loa

The Mauna Loa atmospheric CO2 data comprises the CO2 concentration recorded at the Mauna Loa Observatory from 1958, to 2001. This was used by Rasmussen [27] to illustrate the capability of a structured kernel. Fig 2.1 illustrates the time series, to the human eye it exhibits a clear structure.

In Fig 2.2 we see the GP workflow in action and the importance of kernel choice and hyper-parameter optimisation. For each figure the training data is shown in green. The unseen test data is shown in blue. The aim is to fit the training data and to be a good predictor of the unseen test data. The point of prediction begins from the vertical black line. The blue error bars and yellow error bars show the GP's confidence in its prediction for training and test data respectively.

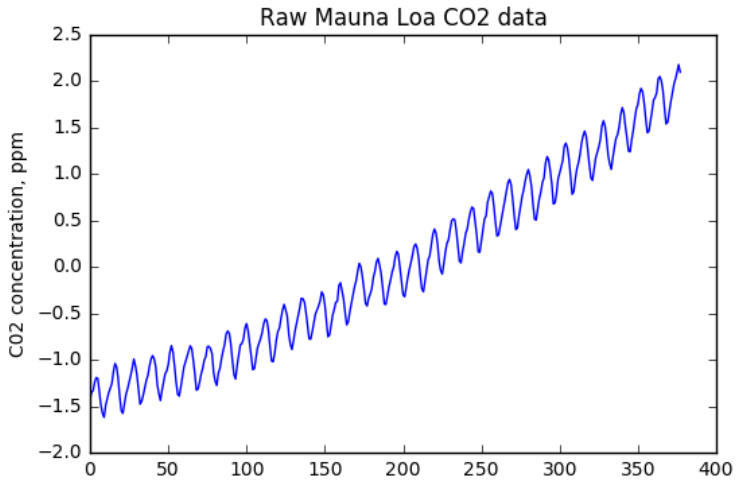

Figure 2.1: Mauna Loa CO2 concentration

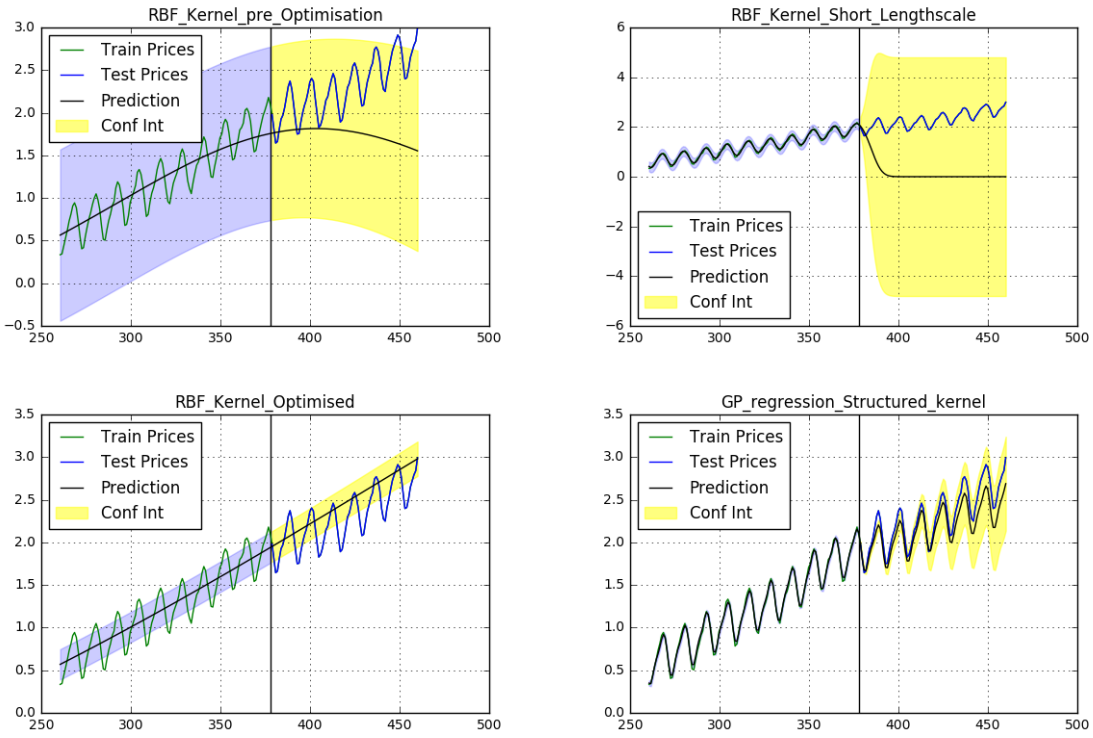

Figure 2.2: The Impact of kernel choice and length-scale

In the first three charts a radial basis function kernel is used. In the last a structured kernel, made up of a linear, quadratic, periodic and radial basis (smoothing kernel), is used.

In the first chart a GP with an RBF kernel is fitted to the data. Although it interpolates the training data, the confidence intervals are wide. The second chart shows the impact of shortening the length-scale, the GP now interpolates the training data excellently but only needs to move a short distance from the training data before it becomes uninformed. In the third chart the GP has been optimised, here it interpolates both the training data and the unseen test data. Notice that because of the bayesian treatment, it does not overfit the training data and indeed does not feel the need to interpolate all the data points. In the third chart the issue is that the kernel lacks structure. In the final chart the GP fits the training data excellently, and does a good job at predicting the unseen test set, the key difference being the richer structure of the kernel.

There are a great many choices of kernel, and it is this that means that GPs are implictly a generalisation of many other regression methods. For example, the Kalman filter, splines and of course basic linear regression are examples of GPs with specific kernels. The main kernels used in experiments in this thesis are the Ornstein Uhlenbeck, Rational quadratic, Matern, and Squared Exponential, whose equations are given below.

2.1.5.3 Kernel Equations

We have included the equations for the main kernels used in this thesis. To aid intuition we have also given a graphical view of the OU and Rational quadratic kernels see Figs 2.3, 2.4, showing the impact of the kernel with increasing distance and some sample paths. Both kernels have a length-scale of 0.5. One can clearly see the roughness of the OU kernel versus the RQ.

OU Kernel

$$k_{ou}(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{|\mathbf{x}-\mathbf{x}'|}{l}\right)$$

Rational quadratic kernel

$$k_{rq}(\mathbf{x},\mathbf{x}') = \left(1+\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\alpha l^2}\right)^{-\alpha}$$

Matern kernel

$$k_{mat}(\mathbf{x},\mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu|\mathbf{x}-\mathbf{x}'|}}{l}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu|\mathbf{x}-\mathbf{x}'|}}{l}\right)$$

Squared Exponential kernel

$$k_{se}(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2l^2}\right)$$

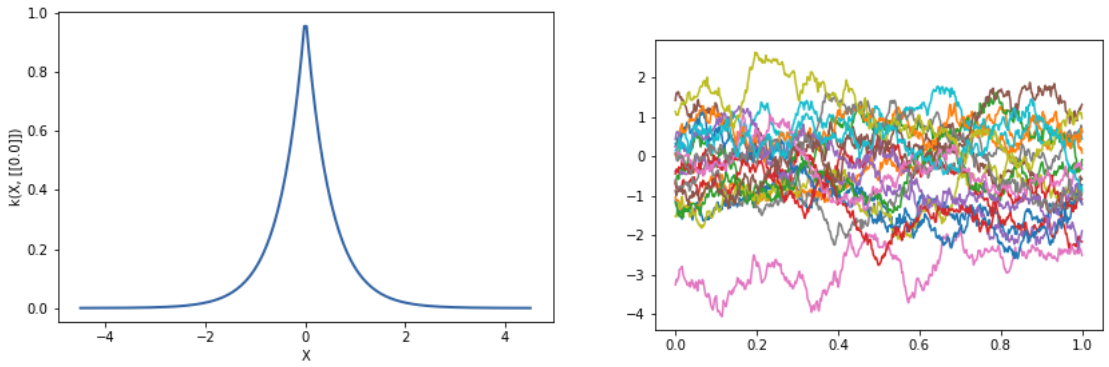

Figure 2.3: The OU Kernel and sample paths

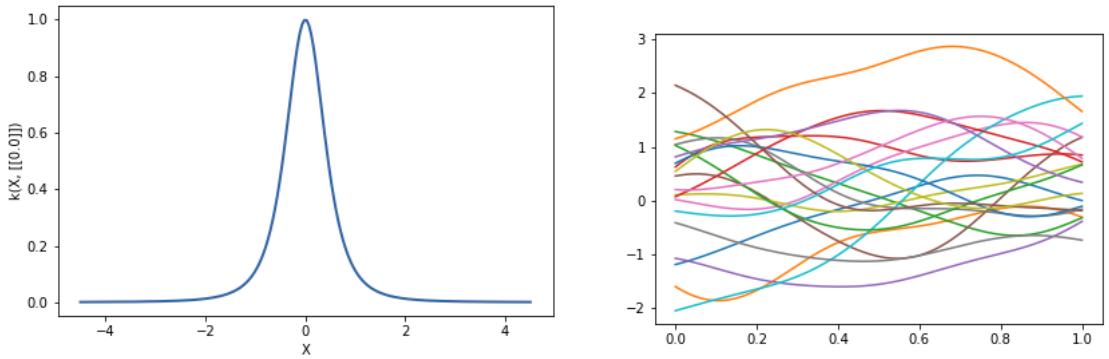

Figure 2.4: The RQ Kernel and sample paths

2.1.6 Optimisation of Hyperparameters

In the majority of machine learning models the accepted technique for optimising hyper-parameters would be through some form of cross-validation using some error

function, measured against a hold-out validation set, often repeated several times. This is a well known and generally accepted method. This is of course possible for GPs also.

An additional option using bayesian techniques is to write the marginal likelihood conditioned upon the hyperparameters of the covariance function. The details may be found in Rasmussen [27] and I give only the briefest of sketches here following Chapados [6].

If θ is the set of hyper-parameters to be optimised and $K_{\mathbf{X}}\theta$ is the covariance matrix that has been calculated using hyperparameters of value θ.

$$K_{\mathbf{X}}(\theta_{i,j}) = k(\mathbf{x}_i, \mathbf{x}_j; \theta) \tag{2.22}$$

Then the marginal likelihood is

$$\Pr(\mathbf{y}|\mathbf{X}, \theta) = \int \Pr(\mathbf{y}|\mathbf{f}, \mathbf{X}) \Pr(\mathbf{f}|\mathbf{X}, \theta) \mathrm{d}\mathbf{f}, \tag{2.23}$$

Under the GP prior $\mathbf{f}|\mathbf{X}, \theta \sim \mathcal{N}(0, K_{\mathbf{X}}(\theta))$ the log-likelihood is

$$\log \Pr(\mathbf{f}|\mathbf{X}, \theta) = -\frac{1}{2}\mathbf{f}' K_{\mathbf{X}}^{-1}(\theta)\mathbf{f} - \frac{1}{2}\log|K_{\mathbf{X}}(\theta)| - \frac{N}{2}\log 2\pi \tag{2.24}$$

Again we are dealing with gaussians and so Rasmussen [27] gives an analytic result for the marginalisation (here I borrow from [6]).

$$\log \Pr(\mathbf{y}|\mathbf{X}, \theta) = \frac{1}{2}\mathbf{y}'(K_{\mathbf{X}}(\theta) + \sigma_n^2 I_N)^{-1}\mathbf{y} - \frac{1}{2}\log|K_{\mathbf{X}}(\theta) + \sigma_n^2 I_N| - \frac{N}{2}\log 2\pi \tag{2.25}$$

Where the first term encourages the model to fit the data and the second controls the capacity of the model thus helping to prevent over-fitting. This can now be

maximised numerically, i.e. we seek

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log \Pr(\mathbf{y}|\mathbf{X}, \theta). \tag{2.26}$$

Which can be found by taking the partial derivative of the marginal log-likelihood with respect to the hyper-parameters given by

$$\frac{\partial \log \Pr(\mathbf{y}|\mathbf{X}, \theta)}{\partial \theta_i} = \frac{1}{2}\mathbf{y}' K_{\mathbf{X}}^{-1}(\theta) \frac{\partial K_{\mathbf{X}}(\theta)}{\partial \theta_i} K_{\mathbf{X}}^{-1}(\theta)\mathbf{y} - \frac{1}{2} Tr\left(K_{\mathbf{X}}^{-1}(\theta) \frac{\partial K_{\mathbf{X}}(\theta)}{\partial \theta_i} \right) \tag{2.27}$$

In this thesis we mostly used the GPy package from Sheffield University, as well as GPFlow [22]. GPy uses the SciPy implementation of L-BFGS, a quasi-newton optimisation method that approximates the Broyden-Fletcher-Goldfarb-Shanno algorithm.

2.1.7 Practical Issues

We have glossed over two issues which matter in practice. The first is that this likelihood is often non-convex and thus may be subject to local maxima. It is thus good practice to apply restarts to the optimisation algorithm. Depending upon the time taken for optimisations, in this thesis we ranged from 3-8 restarts in an attempt to find better maxima. The second is that we have quietly gone non-bayesian. We are now taking point estimates for our hyper-parameters, a full bayesian treatment would instead define a prior distribution on the hyper-parameters and marginalise. The cost of course being computational.

This was not an area of investigation for this thesis, but perhaps worth further investigation is the robustness of these forecasts when we move away from a full bayesian treatment of the hyper-parameters. Anecdotally we found that when using the functional representation explained in Chapter 3, we could alter length-scale hyper-parameters by a fairly small amount sometimes causing quite different forecasts, yet with only very small changes to the log-likelihood of the training data.

A final point which again was not the subject of this thesis, was the subject of cross-validation versus maximising marginal likelihood in practice. As practitioners or ex-practitioners we intuitively felt safer practice might be to examine both. Although the bayesian treatment is correct, there are also embedded assumptions such as having a sensible kernel prior. Again this was not the subject of our experiments. The issue with cross-validation is the curse of dimensionality as the number of hyper-parameters increases. However, if one can cross-validate then other options are opened up such as the ability to create user defined error functions, which may be more directly applicable to the financial domain.

2.1.8 Automatic Relevance Determination

Thus far, regardless of the dimensionality of our inputs $\mathbf{x}$ we use an isotropic norm with a global length-scale hyper-parameter. For example in the case of the squared-exponential kernel.

$$k_{se}(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2l^2}\right)$$

There is a single characteristic length-scale l, operating across all input dimensions. If we believe that the different dimensions may have varying noise levels and ability to predict, then we can rewrite the kernel as follows

$$k_{se}(\mathbf{x},\mathbf{x}') = \exp\left(-\sum_{i=1}^{D}\frac{(\mathbf{x}_i-\mathbf{x}_i')^2}{2\sigma_i^2}\right)$$

Now we have D σ's, one for each dimension. Post optimisation those with a high value have relatively little importance. Thus enabling an implicit form of feature selection, similar in spirit (although not detail) to methods such as L1 regularisation. In this thesis ARD is used for each of the kernels again similar in spirit to previous work in the area by Chapados [6].

In our final real life experiment, we made direct use of ARD, when we found that the GP could predict the emerging market NAV premium very well without

needing the VIX as a feature. This we could see by examining the length-scales of our Matern5/2 kernel.

Chapter 3

Implementation

3.1 Data representations

In the section we describe the data representations used in the thesis. The philosophical idea behind these representations is to enable a forecast over a complete future trajectory. The covariance function within GPs has a length-scale hyper-parameter. If the test data is 'far' away from any of the training data, 'far' being much further than the length-scale of the kernel, then the test data is deemed to be uncorrelated with any existing data and the GP's prediction will be uninformed. The functional and augmented data representations used by Chapados [6, 7] and motivated by Ramsay, Functional Data Analysis [26] is an attempt to mitigate these effects through a different representation of the data and enable longer term forecasts. We test these representations in Chapter 4.

3.1.1 Functional

The idea of the functional representation is to consider a whole curve as an example. This sounds far more complex than it actually is in this context. Following the notation and methodology of [6, 1], we consider a set of N time series each of length M_i, $\{y_t^i\}, i = 1,\ldots,N$ and $t = 1,\ldots,M_i$. Each i represents a different year, and instead of representing the series as one continuous series, we chop it up and simply add an extra dimension. The target price is now indexed by a separate year number and day number within that year. Both are independent variables in the regression.

Given training data from the series $i = 1,\ldots,N-1$ and a partial final series $\{y_t^N\}, t = 1,\ldots,M_n$, we forecast from the last known observation point, for all points going forwards until an endpoint. That is we are trying to learn the joint distribution $\{y_\tau^N\}, \tau = M_n+1,\ldots,M_N+H$. Sometimes mathematical notation can obfuscate rather than illuminate, Table 3.1 shows the actual simplicity of the functional representation in this context and shows target values $y_{249}^7,\ldots,y_3^8$.

Year	Day	Target Price
7	249	787
7	250	793
8	1	786
8	2	785
8	3	788

Table 3.1: Functional Training Data

Note that an anonymous follow up paper to Chapados, Long-Term forecasting using Gaussian Processes [1], stops at this point, using only the functional representation, they did not use augmentation which we will now describe.

3.1.2 Augmented-Functional

All we have done so far is chop time up in a different way, perhaps to help the GP learn repeated patterns across years. However in real applications, if we wish to forecast, then our training data is highly likely to include explanatory variables above and beyond time. Again following the notation and methodology of [6] we represent these by $\{\mathbf{x}_t^i\}$ where $\mathbf{x}_t^i \in R^d$.

Now we wish to find $\Pr(\{y_\tau^N\}_{\tau=M_n+1,\ldots,M_N+H} | \{\mathbf{x}_t^i, y_t^i\}_{t=1,\ldots,M_i}^{i=1,\ldots,N})$. Where τ ranges over our forecast horizon, i ranges over the different series and t ranges over the different observations within a series.

Chapados [6, 7] attempts to include these explanatory variables in a somewhat

unfamiliar way, indeed this method was part of our interest in both implementing that paper [6] and our experimental tests in Chapter 4.

Given a time series identity i, representing the time-series year and a time t within that year we have the model

$$\begin{aligned} E[y_t^i|\mathscr{I}_{t_0}^i] &= f(i,t,\mathbf{x}_{t|t_0}^i) \\ Cov[y_t^i, y_t^{i'\prime}|\mathscr{I}_{t_0}^i] &= g(i,t,\mathbf{x}_{t|t_0}^i, i', t', \mathbf{x}_{t'|t_0}^{i'}) \end{aligned} \tag{3.1}$$

We have now introduced, $\mathscr{I}_{t_0}^i$, this represents the information that was available for series i, at time t_0, noting that information from prior historic series is also available, our model is conditioned on this information. $f()$, $g()$ result from training our GP. We have included our explanatory variables through $\mathbf{x}_{t|t_0}^i$, which is a forecast of $\mathbf{x}_t^i$, given the information that was available at time t_0.

When it comes to forecasting, we now evaluate f, g with our series identity set to N, and the time index t now ranging over the elements of τ we wish to forecast. Thus, for prediction our information is fixed at $\mathscr{I}_{M_N}^N$.

As previously mentioned, the augmented approach is an attempt to include the explanatory variables $\mathbf{x}_{t|t_0}^i$. Of course over the forecasting period we do not know the values of $\mathbf{x}_\tau^N$, so in the next section we describe the mechanism by which this was addressed.

3.1.2.1 Observation time and Target time

We now get to the crux of the data representation used by Chapados [6, 7]. Observation time is defined to be the time at which input variables are observed, this corresponds to the time at which we have information upon which we condition, it is the last point at which any explanatory variables are known. From this point we forecast multiple points into the future. These points correspond to target times, they are times beyond the information known at observation time, the difference between the two times we represent by Δ.

We are now in a position to augment the training set, by including all pairs $< X_{\text{Observation time}}, y_{\text{ Target time }} >$, X being inputs corresponding to all explanatory variables actually known at observation time, and y corresponding to our target price at some point in the future (whose distance away is Δ).

After training on this set we can forecast by fixing our observation time to M_N, the time of our last observation, thus fixing our explanatory variables to $\mathbf{x}^N_{M_N}$ and just allow the target time to range over our forecast period τ. Thus, our model now becomes

$$\begin{aligned} E[y^i_{t_0+\Delta}|\mathscr{I}^i_{t_0}] &= f(i,t,\Delta,\mathbf{x}^i_{t_0}) \\ Cov[y^i_{t_0+\Delta}, y^{i'}_{t'_0+\Delta'}|\mathscr{I}^i_{t_0}] &= g(i,t,\Delta,\mathbf{x}^i_{t|t_0},i',t',\Delta',\mathbf{x}^{i'}_{t'_0}) \end{aligned} \tag{3.2}$$

Notation can mean that this unusual representation may be slightly tricky to grasp at first sight, Table 3.2 shows augmented training data using simplistic dummy data with a very short look ahead. Here for simplicity, we only look ahead to a maximum Δ of 10 days. In year 7 we show a snippet of training data for 2 observation days (day 5 and day 6). Our training features, Obs Last Price and Obs Value reflect the information available for training on those two days. However, we see our Δ increasing through time as we look forward to ever more distant target prices. Note, the target price in the first row matches the target prices in the last as both are looking at the target price on day 13. In the first row by looking ahead 8 days on observation day 5, and in the last row looking ahead 7 days on observation day 6.

3.1.3 Scaling

Augmentation introduces an issue with Gaussian Processes as they currently stand. Imagine a stock with 10 years of daily data. For simplicity imagine 250 trading days a year. Thus we have 2500 data points for a time series regression or a standard functional data representation. When it comes to the Augmentated-Functional view, we have over 300,000 data points (250 days x 250 days x 10 years, of course halved because we only forecast for positive Δ's).GP's have a computational complexity

Obs Year	Obs Day	Obs Last Price	Obs Value	Δ	Target Price
7	5	787	15.2	8	803
7	5	787	15.2	9	807
7	5	787	15.2	10	810
7	6	783	15.0	1	800
7	6	783	15.0	2	802
7	6	783	15.0	3	804
7	6	783	15.0	4	812
7	6	783	15.0	5	808
7	6	783	15.0	6	805
7	6	783	15.0	7	803

Table 3.2: Functional-Augmented Training Data

of $\mathcal{O}(N^3)$. As they currently stand this means that the functional-augmented data representation as presented above is currently intractable for GPs.

Scaling GPs is an active area of research with various methods available such as the Nystrom method [27]. Many induce some form of sparsity or sub-sampling [25, 43]. Also very recently String Gaussian Processes have been introduced by Kom Samo and Roberts [29, 28] as an alternative methodology for scaling GPs.

Chapados [6], used sub-sampling but the full implementation details were not completely laid out. We also used sub-sampling but found in practice that sampling was non-trivial and discuss our choices in the section on design decisions.

3.2 The Ornstein Uhlenbeck Stochastic Process

For the rest of the thesis, we diverge from Chapados [6]. In their work on commodity futures they modelled real time-series with a mean reversion aspect. Our choice is to look at simulated data, with variables we can control. One issue with time-series analysis, particularly very noisy time series is that it can be somewhat difficult to gain statistical certainty that one's models are genuinely viable. Financial time series are notorious in this respect, a sadly familiar story is the outstanding 'back-test' which ends up performing poorly when applied 'live'.

We will now very briefly move away from machine learning to finance. Within financial modelling it is commmon practice to use continuous time stochastic differential equations (SDEs) for modelling. For example, the famous Black-Scholes-Merton options pricing formula within options pricing or the Vasicek interest rate model [40]. SDEs are ubiquitous in finance but bar our Gaussian Processes somewhat outside the scope of this thesis. Good references may be found in Stochastic Calculus in Finance - Countinous-Time Models by Shreve [34] or Arbitrage Theory in Continuous-Time by Bjork [3]. For our simplistic purposes I will just introduce the Ornstein Uhlenbeck process [38].

3.2.1 Motivation

In finance there are various forms of investing, one is called Statistical Arbitrage, a style under whose umbrella one can find practitioners looking for combinations of instruments, which when combined are mean-reverting and stationary. The motivation being that strong movements (either positively or negatively) away from the mean can be traded profitably on average (given stable parameters). There are many references, but two widely known references laying out more detail may be found in Quantitative trading: How to build your own algorithmic trading business, Ernie Chan [4] and Algorithmic trading: winning strategies and their rationale [5], by the same author.

3.2.2 Simulating the OU

The SDE given by the OU process is given by

$$dS_t = \lambda(\mu - S_t)dt + \sigma dW_t \tag{3.3}$$

It is an example of a Brownian motion and the continous-time analogue of an AR(1) process. The parameter μ describes the mean and λ the 'speed' of mean revision. σ is a noise parameter, with W_t being a standard Brownian motion so that $dW_t \sim \mathcal{N}(0, \sqrt{dt})$. Its is commonly used in commodity yield modelling, for example the Gibson and Swartz model [13].

As is shown in [14] there is an exact method for modelling an OU process. A somewhat naive approach might be to look at the above equation and simply decide to discretise, for example

$$S_t = S_{t-1} + \lambda(\mu - S_{t-1})\Delta t + \sigma dW_t \tag{3.4}$$

Taking discrete steps Δt. However because the OU is a continuous time process this is only valid for very small Δt. Gillespie [14] gives the exact formula for simulating an OU which we use in this thesis.

$$S_{t+1} = S_t \exp(-\lambda\Delta t) + \mu(1 - \exp(-\lambda\Delta t)) + \sigma\sqrt{\frac{1-\exp(-2\lambda\Delta t)}{2\lambda}}\mathcal{N}(0,1) \tag{3.5}$$

3.2.3 OU Intuition

The intuition behind the OU is that if σ was set to zero then we have a very trivial realisation with all values equal to the mean (we use mean zero). With a positive σ the process will randomly move away from its mean but will over time wish to revert back. From a Gaussian Process viewpoint it is a very rough process. A very rough prior for the GP often leads to posteriors that do not generalise far away from the training data.

From a financial forecasting viewpoint, the OU is a fairly pessimistic model. It assumes we know nothing about the process other than a tendency to mean revert in a noisy fashion, thus has little structure. Clearly the practitioner would prefer plenty of noise and fast mean reversion, to enable multiple opportunities for profit.

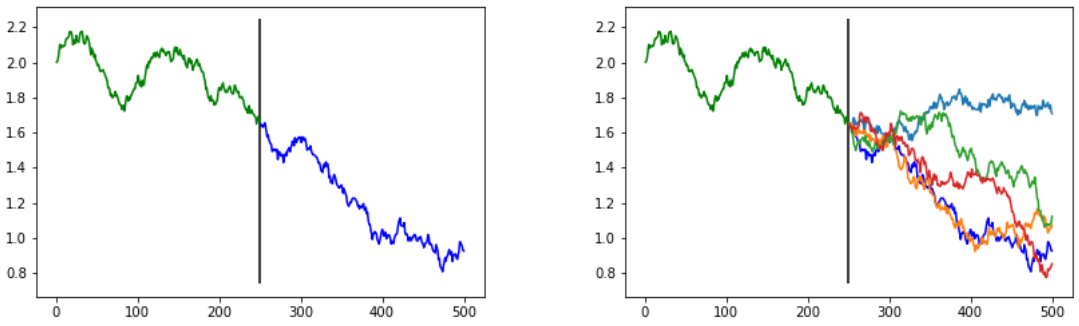

Figure 3.1: OU Simulation

In Fig 3.1, we illustrate a perhaps simple point, but one which is sometimes forgotten. In the left chart we have simulated a single realisation of an OU, with mean zero. It has a slow rate of mean reversion and a low volatility. We started this realisation a long way from its mean, thus the strong downward trend. Let us now imagine the green line was our training data and we are predicting from the black vertical bar. We can see the success of our prediction by measuring against the blue line, but is this correct?

In reality, if we predict and trade multiple times, this is not the truth. In the right chart I have rerun exactly the same OU process with the same parameters from our point of prediction. Any single one of these could have been our test set. It just so happens that we saw the blue line on the left.

The reality with a noisy process is that we would wish to measure our success against the distribution of the process. We do not have that usually, but a GP offers a prediction of this. And this example motivates the test methodology we shall be using in our experiments.

Finally, it is important to note that within this thesis the OU is only one part of our simulated data. To test the functional and augmented viewpoints we add this to a deterministic sinusoidal time-series which exhibits the repetitive behaviour

desired by the functional representation.

3.2.4 Max-Likelihood for the Ornstein Uhlenbeck process

In finance, when one models mean-reverting stationary time series - a commonly used model is an OU process, with fitted parameters. We have chosen to use Max-Likelihood, and this gives a simple and realistic model against which to test our GPs, in later experiments.

Recall that the OU is the continuous time version of the AR(1). This requires contiguous data, if data comes at uneven intervals then this model doesn't suit. However, a GP has no issues with unevenly spaced data. In practical contexts within finance, this can prove useful.

The maximum likelihood solutions for the parameters of our OU SDE,

$$S_{t+1} = S_t \exp(-\lambda \Delta t) + \mu(1 - \exp(-\lambda \Delta t)) + \sigma \sqrt{\frac{1 - \exp(-2\lambda \Delta t)}{2\lambda}} \mathcal{N}(0,1) \quad (3.6)$$

given empirical data, can be found in [39].

$$\begin{aligned} \mu &= \frac{S_y S_{xx} - S_x S_{xy}}{n(S_{xx} - S_{xy}) - (S_x^2 - S_x S_y)} \\ \lambda &= -\frac{1}{\delta} \log \frac{S_{xy} - \mu S_x - \mu S_y + n\mu^2}{S_{xx} - 2\mu S_x + n\mu^2} \\ \hat{\sigma}^2 &= \frac{1}{n}[S_{yy} - 2\alpha S_{xy} + \alpha^2 S_{xx} - 2\mu(1-\alpha)(S_y - \alpha S_x) + n\mu^2(1-\alpha)^2] \\ \sigma^2 &= \hat{\sigma}^2 \frac{2\lambda}{1-\alpha^2} \end{aligned} \quad (3.7)$$

Here

$$\alpha = \exp(-\lambda\delta), S_x = \sum_{t=1}^{n} S_{t-1}, S_y = \sum_{t=1}^{n} S_t, S_{xx} = \sum_{t=1}^{n} S_{t-1}^2, S_{xy} = \sum_{t=1}^{n} S_{t-1} S_t, S_{yy} = \sum_{t=1}^{n} S_t^2.$$

This we implemented in order to test against the GPs.

3.3 Sensibility check

3.3.1 Mauna Loa, Functional GPs

In order to aid intuition about the functional representation and as a brief check on the first part of our implementation. We apply the functional view to the Mauna Loa data used by Rasmussen [27].

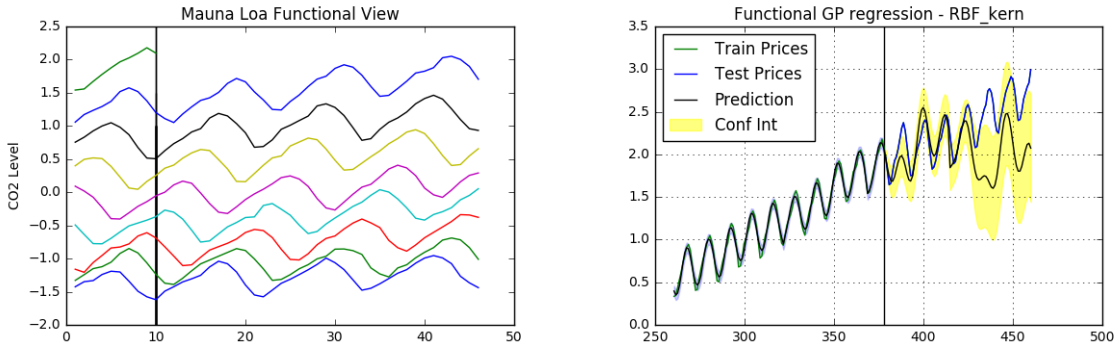

Figure 3.2: Functional GP Mauna

In fig 3.2, we see the functional view of the Mauna Loa data. Each year is now independent. As usual we will attempt to predict from the vertical black line going forwards for the last time series. In the left chart predicting the continuation of the green time series. In the right attempting to predict the blue line.

We trained using a simple RBF GP. The functional representation did appear to 'borrow' from previous times series, expressing more structure than the simple RBF kernel in Fig 2.2, however the seeming instability compared to the structured kernel example in Fig 2.2 raises questions, in the 2010 paper [1], we see similar pictures, however this is an anecdote and not a test statistic. We describe our test methodology in Chapter 4.

3.4 Sampling

The final design choice was as regards data augmentation. How much should we sample? how far ahead should we look in our data augmentation? And given data

augmentation requires features, what predictive features should we use?

In Chapados [6], the sampling strategy was biased towards a shorter delta (look-ahead) with regular observation points each week and multiple targets for each observation point.

Initially we looked at a purely random sampling strategy - with up to 2000 training set members. This produced terrible results. The issue being that often recent information that would have been known using our other data representations...was lost. For example, if we were in year 10, day 50 and trying to forecast from this point, the uniformly randomly sampled training set might have as its last point an observation point of year 9 day 247, looking ahead 3 days. Thus the GP would be trying to predict data in other contexts that would have been known.

Thus, the sampling strategy we chose was to include all the training set examples for each day with a zero delta (replicating normal time series approaches) and to then augment this data with 500 data points regularly sampled, but with a fixed maximum delta. In this case we chose a delta of 50, as we are really now trying to predict 10,20, and 30 days into the future. Even this practical compromise resulted in a heavy computational load, as for each matrix inversion we already have 3000 training set points and of course for optimisation purposes need multiple restarts for our optimisation.

3.4.0.1 Delta

For the augmented approach the Delta is important. This is how far forward from our observation day (and hence known features) the target price is. If we fix a maximum delta when sampling, then beyond this delta the GP has no training examples that are similar and will behave similarly to a GP where the test data has moved from the training data beyond its lengthscale, i.e. uninformed.

3.4.0.2 Features

The augmented approach has no real added value unless we have features. Features that are informative about future target prices. In the financial domain those features may be how cheap a stock is, whether it has positive momentum etc.

For the simulated time series, the features chosen were whether the observation point was in one of 3 regimes. High and going down, low and going up, or neither of the above. The high and low levels were set by the 50 day delta, ie are we low because over the enxt 50 days we will be at a higher level? As one can imagine for a sinusoidal time series this is a close to perfect feature, however as one adds noise this becomes far less perfect, creating false signals as the OU noise.

This was motivated by real life examples, where signals may have information content but can be highly imperfect and noisy.

3.4.0.3 Regime Implementation in a GP Kernel

The 3 regimes mentioned above are effectively categorical features. For a sensible interpretation in a multi-dimensional kernel, we decided to one hot-encode the regime, thus adding three features. These three features were then treated separately within the kernel implementation. We used a radial basis kernel for these 3 dimensions with automatic relevance determination. The rest of the dimensions we applied whatever kernel we were using for the particular experiment we were conducting.

To the author of this thesis this idea was not immediately obvious, and the idea was found in the Kernel Cookbook by David Duvenaud, [9].

Chapter 4

Experiments

4.1 Simulated Data

The functional data approach, would love to see regularities across years. A somewhat trivial way of inducing 'seasonal' regularities across years is a repeated sine wave. Even with the 'wrong' kernel one might hope the functional approach to be able to 'borrow' from prior years, where with a one dimensional data set it would quickly becoming uninformed.

For the core of our simulated experiments we added an OU process - this was to add noise as well as simulate mean reversion. As noise levels increased in the OU, our easy to predict seasonal time series would become corrupted and eventually dominated by the OU.

An initial motivation and indeed our first experiment was to assess the levels at which a standard AR(1) model (commonly used by practitioners) might dominate.

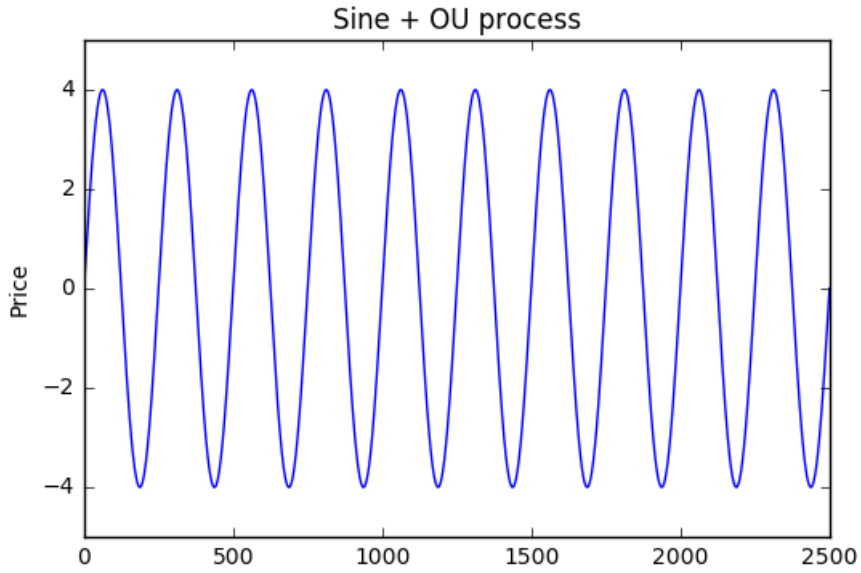

Figure 4.1: Sine wave - no OU noise

Fig 4.1, shows the simulated data with close to zero noise. Clearly unexciting.

However in Fig 4.2, we see the benefit of the functional view even with the 'wrong' kernel. We trained and optimised GPs with an OU kernel and as usual our last observation point is the black line, from which we attempt to predict the blue line. The GPs are trained on the green line. The top left chart shows a GP with an OU kernel, this does not generalise at all well and quickly becomes uninformed.

The top right chart uses the same kernel, but this time with the functional data representation. The prediction is well nigh perfect, in that one can barely distinguish between the blue test set and the black prediction, together with a very strong confidence interval. The bottom left chart shows the functional augmented view. This uses the functional view, but also adds our features, sampling with a maximum delta of 50. This is also a good predictor out to the max delta but then drops off. The final chart shows an OU kernel, without the benefit of the functional representation. This was not implemented by Chapados [6], but we chose to do so because we wished to separate the functional and the augmented views for test

purposes, it is better than the naive 1 dimensional view (top left chart) showing the features offer some predictive ability, but the kernel remains inappropriate for this time series.

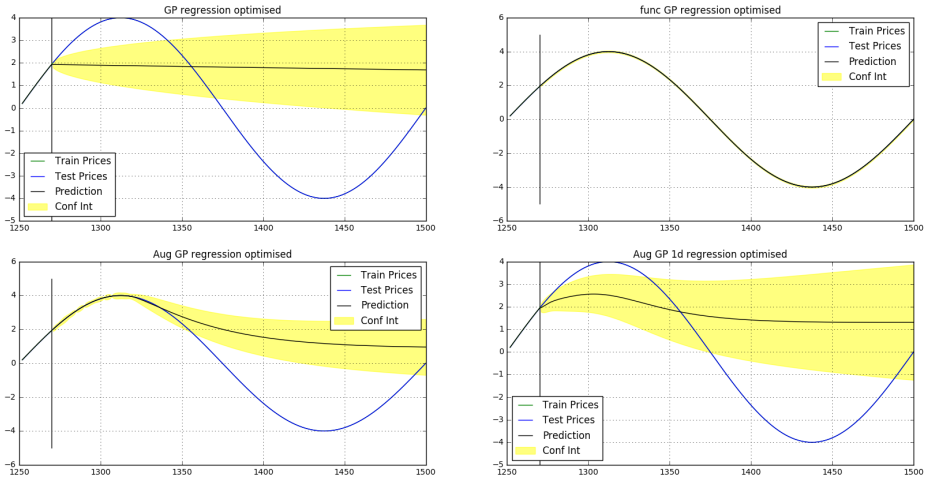

Figure 4.2: GP with OU Kernel- low noise

Fig 4.3, shows the simulated data where the added OU has a $\sigma = 1$, this is of course far more noisy. We will run the same GPs on this training set.

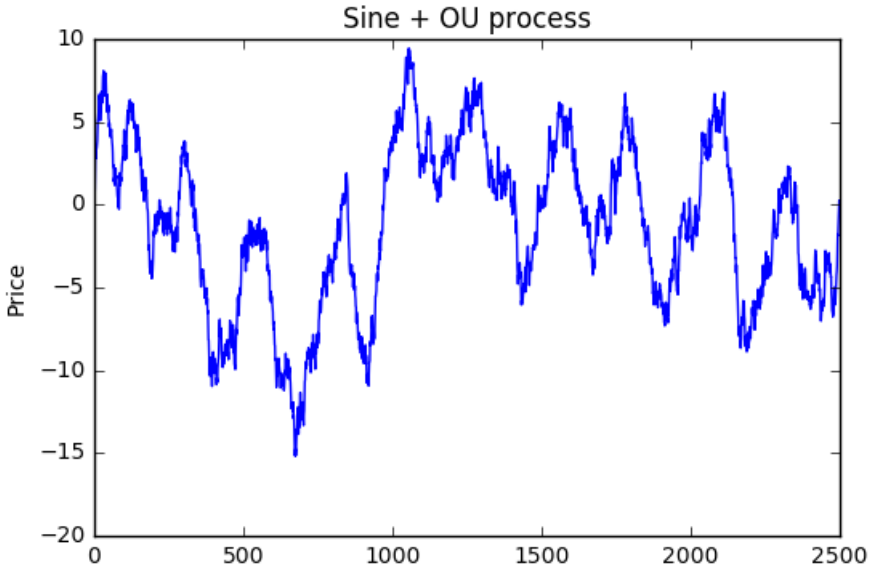

Figure 4.3: Sine + OU, $\sigma = 1$

Fig 4.4, illustrates the results, again using an OU kernel on the noisy data. The predictions seem similar in this case, however without a proper test statistic it is difficult to tell. I explain our test procedure in the following section.

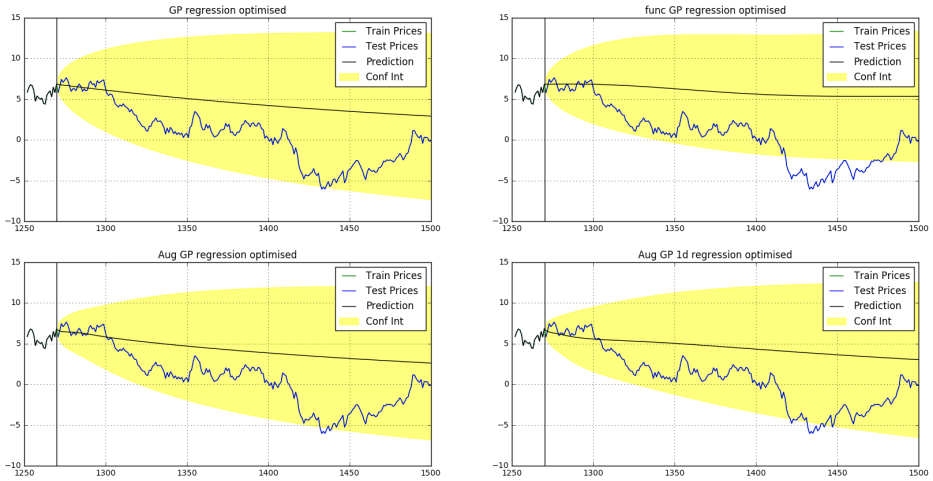

Figure 4.4: GP with OU kernel - high noise

4.2 Test Procedure

Tests were performed in order to assess the impact of models' performance as regards varying noise levels, different kernels, the impact of fat tails, and varying amounts of data in the training set.

From our last observation point the GP outputs a prediction of the mean of our predicted distribution together with its uncertainty. Because we are simulating our time series we can also generate a test distribution, thus we chose the following methodology for testing.

- Simulate training series

  Simulate one realisation of our time series with known parameters.

- Create training data for models

  The data representations are different for one dimensional, functional, aug-

mented and augmented-functional. In addition predictive features are created for the augmented approach as well as for sampling.

- Train Gaussian Process

  Train the one dimensional, functional, augmented-functional, augmented and AR(1) on this training set. Using 5 different restarts for each optimisation.

- Predict

  Predict the mean function and standard deviation going forwards.

- Create test set

  Simulate the test series 1000 times from the same observation point with the same parameters in order to create a distribution of test paths.

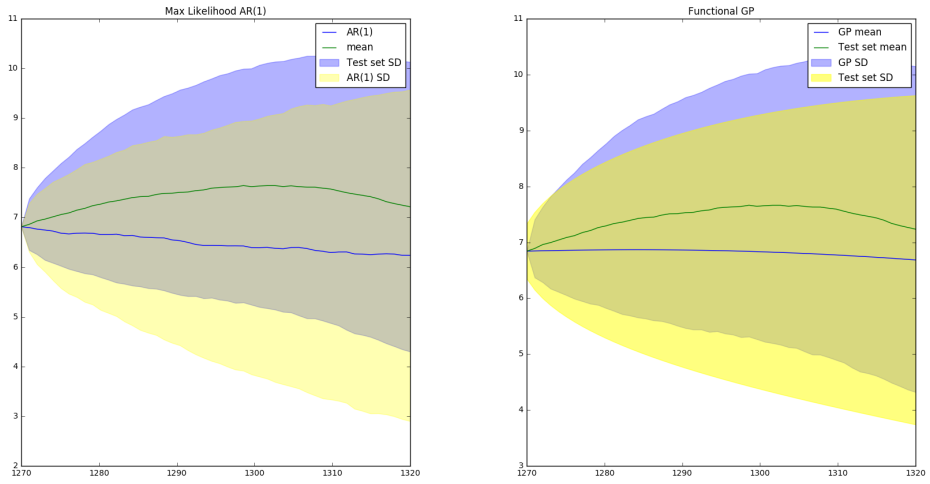

Figure 4.5: AR(1) v Functional - $\sigma = 1$

Fig 4.5, illustrates the idea in the case where we use a GP with an OU kernel to predict our noisy sine wave. The chart shows our GP prediction on the test set. In the right chart, we are comparing two distributions through time. The mean function for the GP (in blue), and its confidence interval (in yellow). But this time rather than comparing against one realisation of the test set, we instead rerun the test set 1000 times to create a test set distribution. The green line shows the mean function of this test set distribution and the blue error

bars the standard deviation.

A perfect predictor through time would have the green and blue lines matching, as well as the confidence intervals matching exactly.The chart on the left shows the test set distribution but this time against the standard model practitioners often use. This is the AR(1) calibrated via maximum likelihood as described earlier. This would normally give a single prediction, but again because we know the parameters of this model it is simple to simulate a distribution, as we have done here.

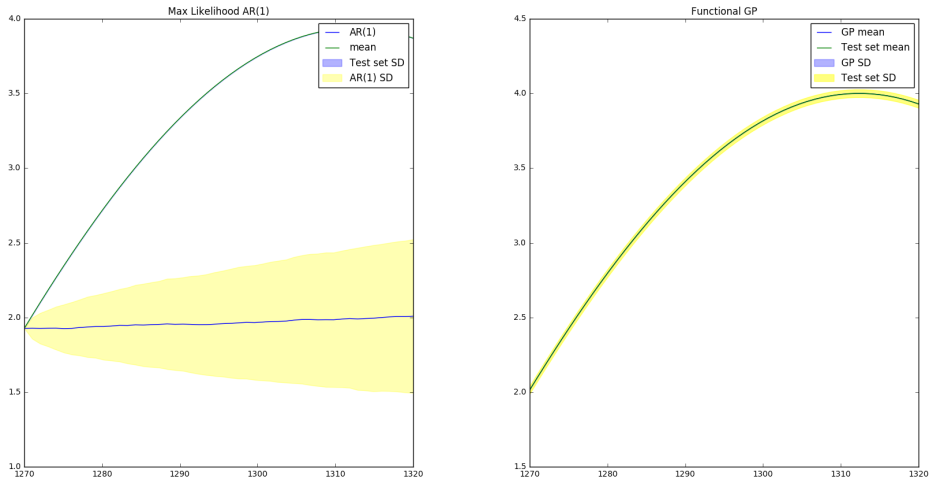

Figure 4.6: AR(1) v Functional - $\sigma = 0.0001$

Fig 4.6, shows the same models (AR(1) and Functional GP with OU kernel), but this time where the OU attached to the sine wave has next to no noise. Both are effectively inppropriate models. In the left chart the AR(1) is an extremely poor predictor. In the right chart however, despite using an OU kernel, the model is saved by the functional representation and is close to a perfect predictor.

- Create different starting levels

  In the charts above we are starting from a particular observation point, in that

particular case the observation point was positive and thus the tendency of the model to mean revert back to 0. Of course this need not have been the case. The observation point could have been near 0, in which case we may have a very flat prediction, or it could have been some way from the 0, in which case we might expect it to mean revert. Thus, we rerun the above 10 times in order to account for these differences, we would wish to do more but when we later vary parameters this experiments become computationally expensive time-wise.

- Compare Means

  We can now compare each prediction at fixed points in the future. We chose 10,20 and 30 days, but indeed this is simply for reporting purposes as we have the whole trajectory for free. We know the mean prediction at these points, and we can calculate the mean from our simulated test set. The more accurately we can predict this mean, the more accurately we can predict our actual returns. We wish to penalise an incorrect prediction over the whole curve, so we calculate the mean squared error over the whole trajectory. This has the disadvantage that each metric is not independent, but the advantage of picking up a prediction that happens to say coincide at 10 days forward but is actually a poor predictor for the earlier part of the prediction.

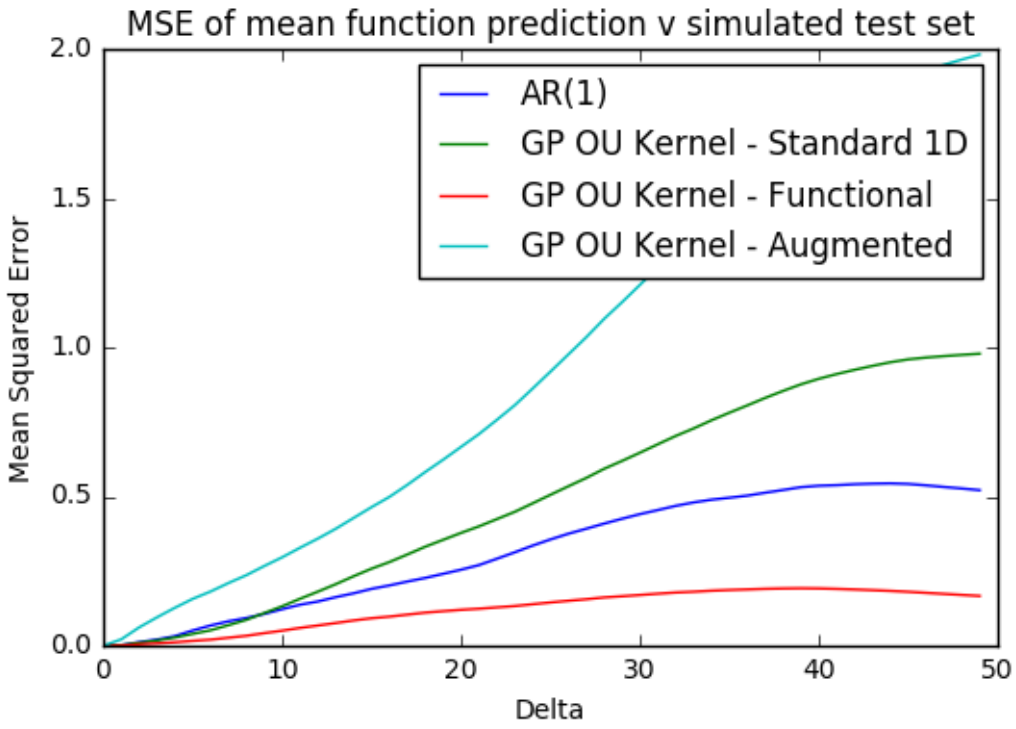

Figure 4.7: Mean Squared Error of mean function predictions with increasing delta

Fig 4.7, illustrates the idea behind our test statistic. We have moved from a GP giving a prediction of a noisy time series, to comparing the GP distribution against a test set distribution through time. Now we wish to see which models predict better. The chart shows the AR(1), the functional GP, the augmented GP, and standard GP. The GPs have an OU kernel, the mean squared errors are against the actual test distribution and are averaged (so for example the MSE at day 10 takes into account the errors of the whole path up to day 10).

We can see that the functional GP was a better predictor, than the AR(1), even with a relatively high noise level. Given the high noise level it appears the signals given to the augmented view did more harm than good.

- Compare Standard deviations

  If we are able to also predict the standard deviation then we would be able to

size our positions correctly, thus reducing the risk of unforeseen capital loss. So we also compare the standard deviations of the predictions 10, 20 and 30 days into the future, this is exactly the same methodology as for the mean functions described above.

- Change parameters

  For example, if we are assessing the impact of noise on the different model's ability to predict, we rerun all the above for multiple noise levels. Note that we would use more starting levels, but even assessing 10 different noise levels for the above procedure takes several hours on a quad core 3Ghz machine.

4.3 Experiment 1-Noise

4.3.1 How do the GPs and different data representations cope with noise?

Chapados and Bengio [6], attempted to prove the efficacy of the augmented-functional approach applied to commodity futures using a Rational Quadratic Kernel with automatic relevance determination.

If we recall the idea behind the functional approach. It is to enable forecasting by 'borrowing' information from prior years. The idea behind augmentation is to also enable prediction of our features, multiple time points in the future.

To simulate the Chapados calendar spreads, we first wanted our test series to exhibit mean reversion and so used an OU-process. However we also wished to create regularities across years. Thus, we added the OU process to a sine wave.

This enabled us to simulate the functional view, but then to test augmentation we need features. The features chosen were categorical features describing the 'regime' that the last observation point was in for the underlying sine wave (flat regime, low and going up, high and going down).

Chapados tested functional-augmentation and functional only. So it is unclear how much of their positive results came from augmentation, or from the ability to 'borrow' from previous years through the functional approach. In our tests we also included augmented data in one dimension, without the functional structure, to clarify this.

4.3.2 Noise

When there is very little noise, we see considerable structure and would hope the functional view can learn this, ideally forecasting for longer than a standard GP regression would be capable of (particularly given we did not use a periodic ker-

nel). As noise is added, the structure is degraded, as are our signal/ features - where price levels may be set by random noise and not the underlying process, giving misleading signals. Eventually we would expect the pure OU process to dominate the sine wave.

The test calibrated the performance of our data representations using the same kernel that Chapados and Bengio used [6]. The aim being to compare the repesentations against one another and against the AR(1) commonly used in industry.

Chapados used an RQ kernel, yet we are adding OU noise, and that is on top of a periodic time series. Given the relative 'smoothness' of an RQ kernel, we might expect it to under-perform the calibrated AR(1) above a certain noise level. And lower than a certain noise level, the process becomes deterministic but periodic - so the RQ kernel is imperfect both ways. Our secondary interest was to see if the data representations used by Chapados mitigated this imperfection in anyway. After all, in real life we don't actually have an Oracle to point out the correct kernel prior.

4.3.3 Results - Predictive power under varying Noise levels

The points below summarise the results of Table A.1, and Fig 4.8, which show the results for a 10 day forecast. The results are rather considerable so full results may be found in appendix A

1. Standard Gaussian Process versus Functional

   Across all noise levels, functional was a better predictor than a standard GP. Moreover at very low noise levels the functional approach could predict correctly for far longer than the standard GP (not shown in the table below, but in the appendices).

2. Standard Gaussian Process versus Augmented

   Again across all noise levels the sampling approach used by the augmented

representation was a better predictor than a standard GP. Indeed augmented was a better predictor than functional. Begging the question of whether in Chapados [6], it was actually the quality of their predictive features that might have had a greater effect than the pure functional approach?

3. Functional-Augmented

   Across all noise levels the Functional-Augmented representation was a better predictor than a standard GP, a functional GP and an augmented GP. This agrees with the conclusions in Chapados in our simulated context.

4. Noise Levels

   As noise increased then the different representations predictive ability deteriorated at different rates. The worst was a naive standard GP, next the functional with no features, next augmented and best the functional-augmented. The data representations did mitigate the impact of starting with an inaccurate kernel.

5. High noise levels and AR(1)

   The best of the approaches, functional-augmented was a better predictor than AR(1) below noise levels with a $\sigma = 0.4$. When noise was greater than this, none of the structure embedded in the functional view, nor the features given by augmentation had value above and beyond the simple AR(1) model. This is for the RQ kernel. We test different kernels in the next experiment.

6. Low noise levels and AR(1)

   At low noise levels all of the GP approaches, outperform the AR(1). Indeed the functional predicts close to perfectly for longer than our 50 day horizon, with the augmented-functional close to perfect out to its maximum delta. It may appear that the AR(1) still maintains good predictive power at low noise levels, however this is somewhat illusory. By this stage we are looking at a close to deterministic process. The error being the difference on average between the AR(1) and a sine wave i.e. simply the difference between exponential decay and the movement of a sine-wave from whatever point we

start. At low noise levels we see the GP approaches are actually orders of magnitudes better predictors than an AR(1).

7. Consistency of Prediction across time

   As the delta increased we found consistency of results. In that the best predictor over 10 days, was also the best predictor over 20 and 30. The complete results may be found in appendix A, including the results for the volatility predictions which were less conclusive.

The summary is that as the noise level increases, the structure that the GP could benefit from is lost and a simple model suffices. We have calibrated the levels at which this occurs and shown how the functional-augmented does improve predictive power compared to a naive GP model, in this particular context. Below a $\sigma = 0.4$, the functional-augmented is a better predictor than an AR(1) as more structure is apparent.

We are adding OU noise and only have a very simple and smooth kernel, thus following Chapados by using an RQ kernel. This begs the question as to whether alternatives kernels might have done better.

4.4 Experiment 2-Kernel choice

In this thesis we are only using simple kernels. We have followed [6, 7, 1], who used a rational quadratic and a squared exponential kernel respectively. In this experiment we tested the results on other kernels.

Engineering kernels is an active area of research, for example David Duvenaud, Automatic model construction with Gaussian processes [10]. It is certainly less complex than the functional-augmented approach explored in this thesis. It is also an approach that doesn't drastically increase the training set, resulting in the sampling that the methodology followed in this thesis required.

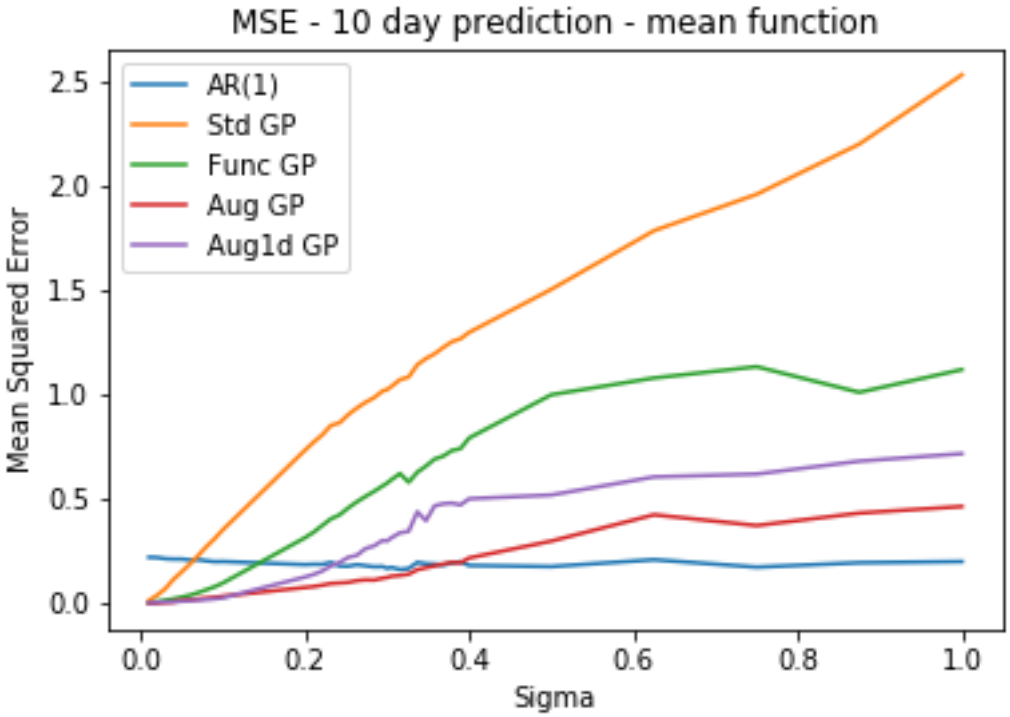

Figure 4.8: Mean Squared Error, 10 day prediction

4.4.1 How much does kernel choice matter?

In the first experiment we followed Chapados [6], by using a Rational Quadratic kernel. Here we explore simple alternatives, including the Squared Exponential kernel applied in [1]. The latter applied the functional approach to Hewlett Packard, Yahoo and Starbucks stocks, using adjusted closing prices with an RBF kernel (Squared Exponential). We attempted to replicate this using the same data, however the paper did not provide summary statistics. But even so, we were able to closely replicate their predictions in their paper. However, we could only do this by artificially constraining the length-scale of the RBF kernel to force the year in the functional approach to be taken into account. Otherwise the RBF GP simply reverted back to its prior.

Indeed in the results below we see the RBF kernel (which really expects smooth functions) gives by far the worst results.

We tested the RQ, RBF, and Matern3/2 kernel. The final kernel we tested is really the appropriate kernel for the type of stochastic process we are adding, which is an OU kernel.

Because we are now testing across kernels, we need to select a certain noise level. We chose a $\sigma = 1$. At this level the AR(1) strongly outperformed the functional-augmented RQ kernel in the last experiment. We wished to see whether alternative kernels could close the predictive gap under such a high noise level.

4.4.2 Results- Predictive power of different kernels

Kernel	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Rat-Quad	0.200159	2.529615	1.119038	0.463764	0.716609
OU	0.241812	0.385954	0.415322	0.321764	0.575269
Matern3/2	0.192320	3.173271	2.581655	0.440301	0.799059
RBF	0.315257	13.075297	11.401516	0.933333	2.083939

Table 4.1: Kernel Test MSE- Sigma=1, Multiple Kernels - 10 day forecast

Kernel	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Rat-Quad	0.739826	6.439397	2.871008	1.714489	2.179615
OU	0.753923	1.298962	1.377844	1.053084	1.641579
Matern3/2	0.681386	13.530849	9.950751	1.689285	2.820553
RBF	1.048223	34.307963	28.219531	2.985530	6.778340

Table 4.2: Kernel Test MSE- Sigma=1, Multiple Kernels - 20 day forecast

Kernel	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Rat-Quad	1.337654	9.701094	4.303638	3.086880	3.595598
OU	1.343196	2.389121	2.539594	1.990475	2.796935
Matern3/2	1.335548	24.904670	17.563196	3.825518	6.448994
RBF	1.992848	43.881841	35.140114	5.501070	14.011281

Table 4.3: Kernel Test MSE- Sigma=1, Multiple Kernels - 30 day forecast

1. Best Kernel

   The kernel with the best predictive power was the OU kernel. At high noise levels, in terms of statistics the predictive power of an OU kernel is close to indistinguishable compared to an AR(1) model (which of course we might

expect!). At lower noise levels the benefits of the functional and augmented approaches are gained because the GP is still able to predict the underlying structure and thus have greater predictive power than an AR(1) (as shown in experiment 1).

2. Worst Kernel

   The squared exponential kernel had by far the worst predictive power. We believe it is simply too smooth for the functions we are dealing with here. Interestingly the performance of this incorrect kernel was drastically improved when the functional-augmented structure was used. Again lending weight to Experiment 1, where the poor predictive power of using the wrong kernel was mitigated by the func-aug representation. This also lends weight to our struggle to replicate the paper, Long Term Stock Market Forecasting using Gaussian Processes[1], where we could only achieve similar pictures to the authors by artificially constraining the year length-scale in the RBF kernel

3. The Matern3/2 and Rational Quadratic kernels

   The results here were fairly similar to one another, indeed the kernel chosen by Chapados, the RQ is not too far away in predictive power from the ideal kernel for this process, the OU.

4. Functional and Functional-Augmented

   The results were consistent with experiment 1, the data structures improved predictive power for all kernels, with the best being the functional-augmented. The only case where there was no improvement in predictive power was for high levels of noise with the OU kernel. Here the std GP performed just as well as the clever data representations, which added no benefit.

5. Consistency across Deltas

   The results were again consistent across time. The best predictor at 10 days was also the best at 20 and 30.

In Fig 4.9, we have shown a typical simulation of our training time series. This has been simulated at a $\sigma = 0.5$. For the best of our kernels, the OU, above

this noise level we see no significant difference in predictive power compared to an AR(1)

In Fig 4.10, we have shown the predictions for each of the four data representations for the same process at an observation day of 1270 (5 years and 70 days). The predictions are to the end of that year.

In Fig 4.11, we have shown the test results of two predictions. On the left the AR(1) which was calibrated on the training data and then a predictive distribution given against the test set distribution. In the right hand figure we see the same for the functional-augmented GP with an OU kernel. The key lines to examine are the blue and green lines. Even at this high noise level the functional-augmented GP is able to give better predictions than the AR(1), one can see this by noting that the AR(1) is mean reverting back slowly, whereas the GP has learned that the process will in future curve up slightly before mean reverting.

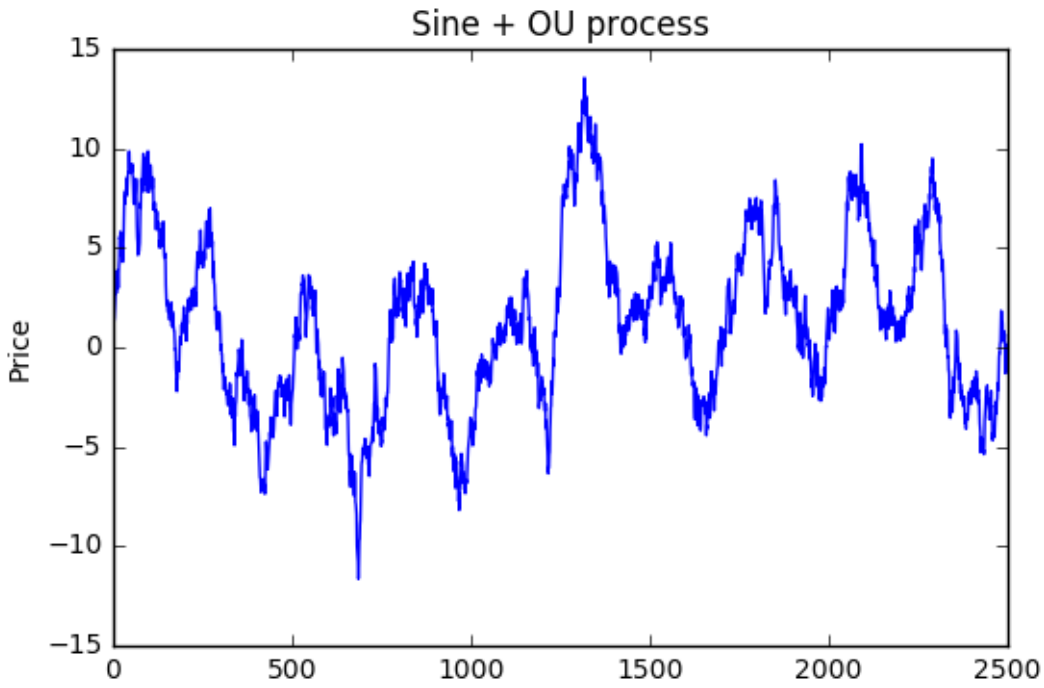

Figure 4.9: Price Process - $\sigma = 0.5$

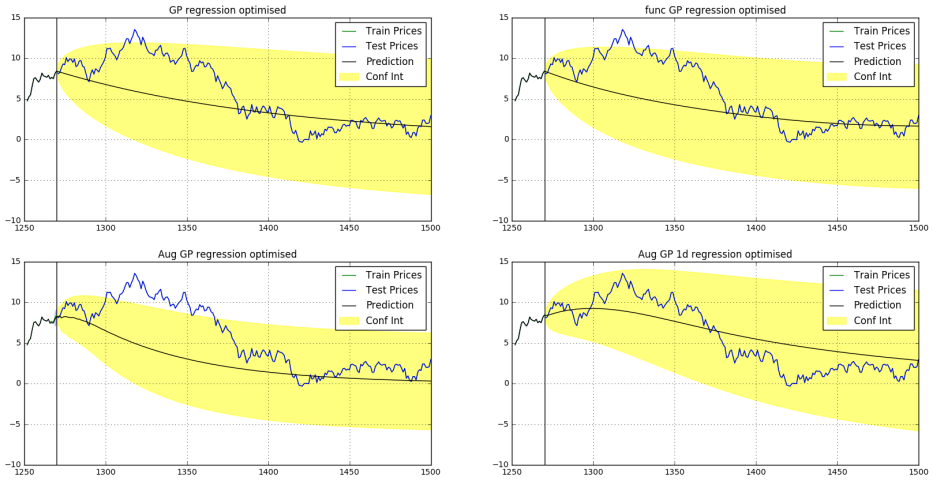

Figure 4.10: GP(OU) predictions - $\sigma = 0.5$

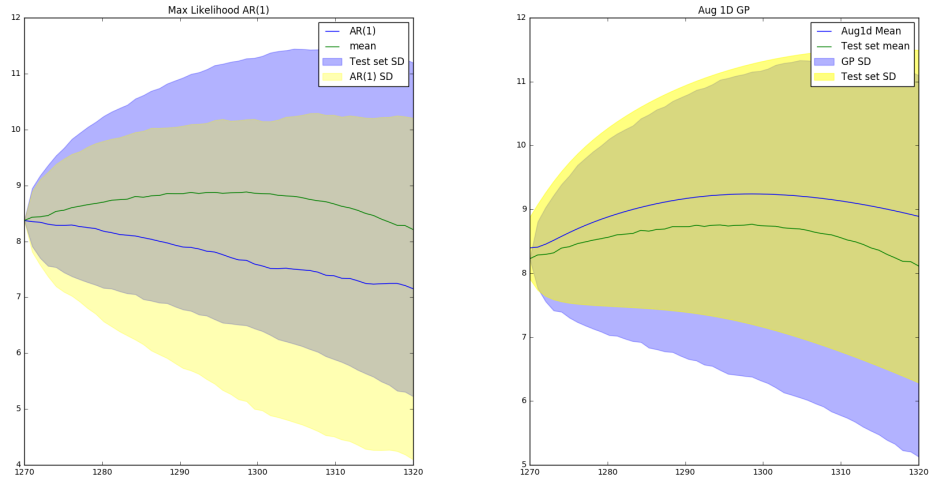

Figure 4.11: GP(OU) results - $\sigma = 0.5$

In summary, the OU kernel predicts more accurately than the Matern3/2 and Rational Quadratic kernels, with the squared-exponential kernel predicting poorly. The functional-augmented structure did improve the predictive power of each kernel, drastically improving the struggling squared-exponential kernel.

Further work could be done in this area. We alluded to the fact the squared-exponential kernel is too smooth. In fact in Rasmussen and Williams, [27]. The authors discuss more precise methods of assessing the 'smoothness' of a kernel using the fourier transform and looking at the eigenvalues. This is more concrete.

4.5 Experiment 3-Fat Tails

4.5.1 What if we introduce 'fat tails'?

It is commonly known that financial time series rarely seem to exhibit Gaussian noise, but have fatter tails where the probability of extreme loss is much lower than a Gaussian might suggest.

We wished to test the impact of fat tails on Chapados' [6] chosen kernel the Rational Quadratic. In experiment 1, we saw that above a noise level of 0.4 the AR(1) tended to be a better predictor regardless of the data representations. We chose a fixed $\sigma = 0.38$. At this noise level the RQ with a functional-augmented structure is a better predictor than the AR(1).

We chose to use the Student-t distribution and varied the degrees of freedom in order to add fatter tailed noise. With a large number of degrees of freedom the Student-t approaches a Gaussian, with a low number then extreme moves become more likely. The equation for the pdf is given below.

$$\Pr(x, df) = \frac{\Gamma\left(\frac{df+1}{2}\right)}{\sqrt{\pi df}\,\Gamma\left(\frac{df}{2}\right)} \left(1 + \frac{x^2}{df}\right)^{-\left(\frac{df+1}{2}\right)}$$

(4.1)

The Student-t pdf for the degrees of freedom we tested is given in figure 4.12.

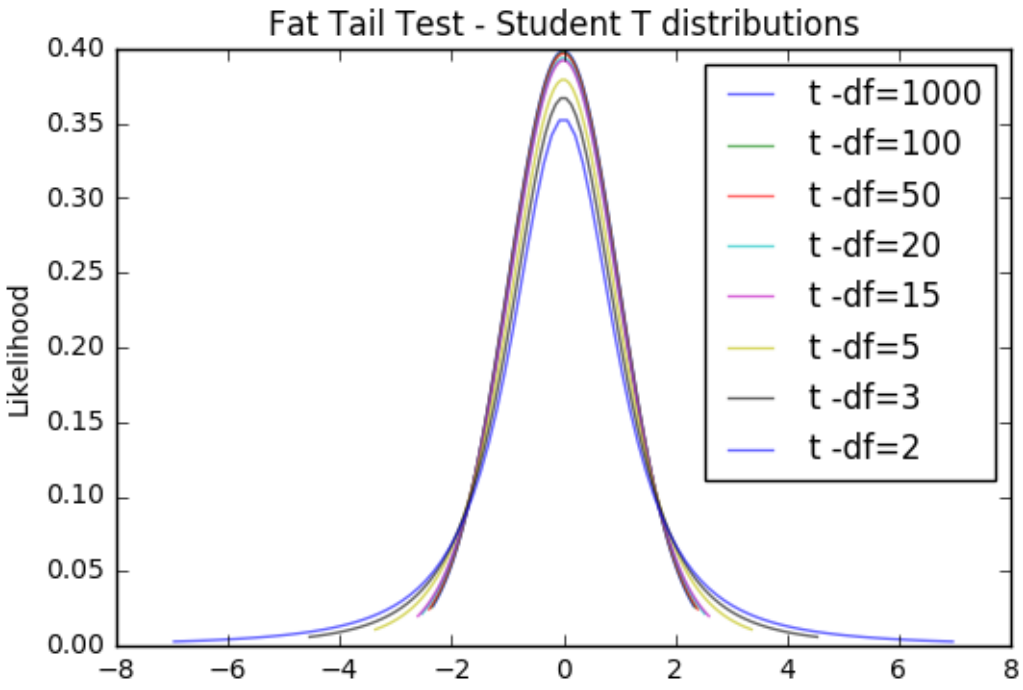

Figure 4.12: Student-T pdfs - Fat Tail test

4.5.2 Results

Tables 4.4,4.5, show the results for the 10-day predictions with varying degrees of freedom. The remaining results are in Appendix B.

1. Most Robust Predictor

   The best predictor as we made tails fatter was again the functional-augmented GP. The results were in-line with other experiments in that functional-augmented predicted best, then augmented, then functional and finally a standard GP.

Degrees of Freedom	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
1000	0.230755	0.979164	0.730086	0.162114	0.357263
100	0.246249	1.17295	0.782307	0.200303	0.360546
50	0.237387	0.978202	0.756199	0.249668	0.432981
20	0.271697	2.764549	0.548723	0.353936	0.381440
15	0.269132	2.91509	0.673308	0.238837	0.463434
5	0.285076	0.98209	0.575406	0.359739	0.583631
3	0.236719	1.862816	1.317582	0.548689	0.763959
2	263.41	32653.4	76371.9	740.3	48116.9

Table 4.4: RQ- Sigma=0.38, Student-t - 10 day forecast Mean Function

Degrees of Freedom	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
1000	0.001789	0.116764	0.097042	0.048652	0.020827
100	0.001256	0.109116	0.088696	0.044788	0.018523
50	0.001643	0.097567	0.082645	0.05625	0.016674
20	0.001624	0.117463	0.099621	0.050229	0.022806
15	0.001814	0.122605	0.109712	0.043295	0.021840
5	0.148879	0.054682	0.053846	0.092781	0.034454
3	0.436201	0.688179	0.665197	0.381423	0.244926
2	1.143485	1.362697	1.329857	1.65723	1.482673

Table 4.5: RQ- Sigma=0.38, Student-t - 10 day forecast Std

2. AR(1)

   The AR(1) was initially a poorer predictor than the functional-augmented GP. However as we reduced the degrees of freedom below 20, the AR(1) had the lowest MSE of its predictions. The results and experiments were similar in nature to experiment 1 - adding noise. As noise increased then structure was lost and the AR(1) was the best predictor. In this case we started at a level of noise where the RQ kernel was a better predictor, but as the tails became fatter the AR(1) took over.

3. Standard Deviation

   For the standard GP, Functional, Augmented and Functional-Augmented the author could see differences, but would hesitate to make any conclusive findings in terms of the different representations ability to predict the standard deviation of the test set distribution.

4. Pathological Degrees of Freedom

With two degrees of freedom all the models had no essentially no predictive ability. We haven't included the time series here, but they appear pathological with massive jumps and don't really look like financial time-series.

In summary, adding fat tails instead of Gaussian noise behaved very much like simply increasing the volatility of the OU process, as we did in experiment 1. We started at a level of volatility where the RQ-GP was a better predictor than an AR(1) but as we reduced the degrees of freedom in the Student-t below 20, the AR(1) became the better predictor.

4.6 Experiment 4-Small data

4.6.1 How much data do we need for these models to learn the process?

Chapados [6, 1], used 2500 data points for their intial data set. This is 10 years of 250 days of data. In practical situations we may not have the luxury of so much history but may be modelling a fairly recent time series. We wished to examine the robustness of the above results in the face of less data.

Figure 4.13, shows an instance of a reasonably good prediction for both the functional and functional-augmented RQ-GP. The observation point is only 20 days into year 2, so only one year of history. Yet both performed well. Rather than looking at particular instances we will test as before using simulations of test distributions. Given we now have smaller data sets we have the luxury of running more simulations. As before we used multiple restarts for our GP, and simulate the test distribution 1000 times, but this time we re-ran everything 50 times, to give more price levels at our last observation point.

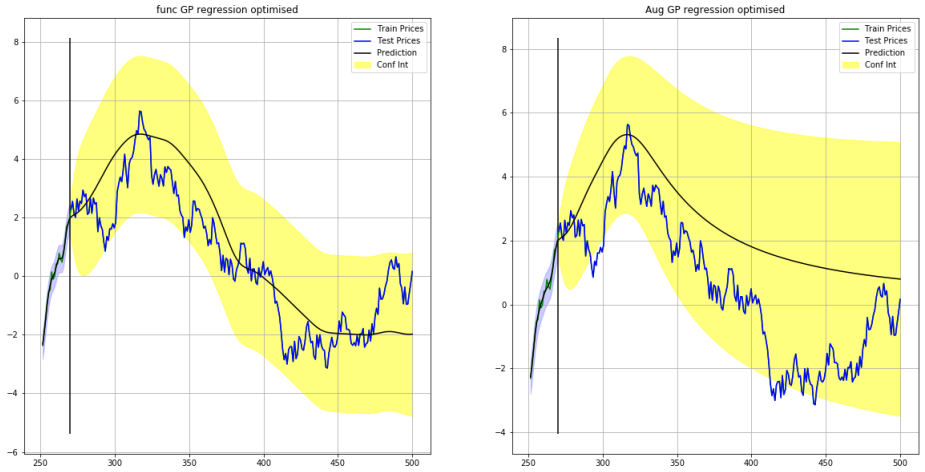

Figure 4.13: RQ - Functional and Augmented, Low data Predictions

Tables 4.6, 4.7, show the mean squared errors for the mean function predictions for 10 and 20 days respectively. The full results are in Appendix C. We began our simulations in year 2, at observation day 50, (i.e. 550 days of data) and worked our way backwards until we had only 50 days of data. As in the previous experiment we used a $\sigma = 0.38$, the point at which the RQ kernel was the best predictor but where if noise increased or tails became too fat it would be beaten by the AR(1).

4.6.2 Results

Obs Year	Obs Day	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
2	50	0.020602	0.573366	0.469432	0.087942	0.177772
1	200	0.168553	0.113113	0.252316	0.086598	0.075516
1	150	0.150945	1.109502	0.898004	0.233877	0.293358
1	100	0.428651	0.316786	0.480829	0.225174	0.278258
0	150	0.218372	1.318972	1.319897	0.293534	0.293641
0	100	1.894154	0.39154	0.391543	0.285445	0.285234
0	50	1.609344	0.449006	0.448999	0.182029	0.179624

Table 4.6: RQ- Sigma=0.38, OU - 10 day forecast MSE - Small Data

1. Best Predictors

   With less and less training data, the functional-augmented and augmented GPs were the best predictors. In hindsight this is logical as they have the

Obs Year	Obs Day	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
2	50	0.0086741	1.383307	1.151605	0.226623	0.327411
1	200	0.194001	0.544586	0.256715	0.256715	0.241836
1	150	0.488949	3.248612	2.529134	0.766008	0.988561
1	100	1.74482	1.409138	0.480829	0.855082	1.081898
0	150	0.776458	3.740487	3.740489	0.988972	0.990674
0	100	5.890957	1.349572	1.349574	1.247833	1.247454
0	50	3.513869	0.949658	0.949644	0.318234	0.31382

Table 4.7: RQ- Sigma=0.38, OU - 20 day forecast MSE - Small Data

Obs Year	Obs Day	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
2	50	0.217699	1.884776	1.648823	0.392047	0.396859
1	200	2.111592	0.241403	0.749169	0.579606	0.656602
1	150	0.856949	5.345155	4.081297	1.38428	1.836873
1	100	3.991562	2.454904	2.63404	1.934716	2.528708
0	150	1.647748	6.096296	6.096301	1.839766	1.842913
0	100	11.131405	2.901266	2.901266	2.941594	2.941013
0	50	5.584797	1.178199	1.17818	0.369026	0.36372

Table 4.8: RQ- Sigma=0.38, OU - 30 day forecast MSE - Small Data

benefit of sampling features and thus the augmented approach came into its own in this particular test.

2. AR(1)

   All GP approaches predicted better than the AR(1) as data became very sparse. With less than 100 days of data the AR(1) was the worst predictor.

3. Functional

   The standard GP and the functional give the same answers when we are in year 0. Obvious in hindsight but there is no benefit to the functional approach as it has no previous years to look at. This was of course the same for the functional and augmented-functional approaches.

In all our experiments so far, functional-augmented GPs have been the best predictors with the difference becoming marginal between an OU kernel and an AR(1) at high noise levels, and worse if one has the 'wrong' kernel in the first place. Augmentation has given benefits such as being able to use signals/ features, but it has also introduced problems - too much data for a GP to handle due to

matrix inversion. This in turn introduces scaling issues where we had to resort to sub-sampling.

The results from our final experiment show a benefit to the augmented approach where we only have mall amounts of training data. Augmentation increased the predictive power of the GP, above and beyond a standard GP and certainly beyond an AR(1).

4.7 Experiment 5-Real Data

In order to demonstrate how GPs could be put to use in a real trading strategy, we tested viability on what one would expect to be a highly stationary series in real life. The premium to NAV of two IShares MSCI ETFs. The first trade uses real data, but is a fantasy trade and not realisable in practise, the second trade is executable.

4.7.1 Premium to NAV EEM ETF

The IShares MSCI Emerging market ETF, better known as the EEM ETF, is a basket of stocks packaged as an exchange traded fund by MSCI. It is made up of emerging market stocks. This is a highly liquid ETF. We used the daily price series from 2000, until the present.

The idea behind the trade is that the underlying basket of stocks has a specific Net Asset Value (NAV), and the ETF itself also can be traded on a live basis. The trade idea is that one could thus trade deviations between the underlying NAV and the ETF price. This would be enabled by requesting a broker trade buy the underlying basket of stocks and short the ETF when the NAV is at a discount. Of course, one would do the opposite trade when the NAV is at a premium. We constructed the EEM NAV premium from the year 2000 onwards. (Many thanks to Ahrash and Zais for this construction).

One subtlety that makes the idea nice in theory but wrong in practise is time-zones and foreign exchange. Whereby the ETF is traded in dollars with closing

price times set in the United States, and the underlying basket of stocks closing at various earlier times of the day depending upon their location. Thus, the spectacular sharpe ratios shown below are not realisable in practice.

4.7.2 Gaussian Process Training and prediction

We chose to train using a rolling window and a training history of 500 days. Using a Matern5/2 kernel we then trained our GP on this series with one dimensional data augmentation. Our features being the day, the delta and the level of the VIX (an implied volatility indicator). We optimised the GP and predicted from one to four days forwards. (The series is quickly mean reverting, thus we considered that larger deltas were unecessary. Clearly the more structured the data and the more powerful our features then the more the GP and augmentation would add value. In this case this is still a very simple construction).

We rolled our window forward every day thus repeating the process. We did this for over 2500 days (starting at day 1250). Due to the costs of optimisation we actually re-optimised our GP every 10 days and used the saved hyper-parameters for predictions on intermediate days (being careful to refresh our kernel every day given that GP's are non-parametric and require every training point, and both the training data and prediction window is changing each day).

Thus, for every trading day the GP gave us a prediction from one to four days forwards of the EEM NAV premium, as well as its uncertainty.

Fig 4.14, shows examples of paths predicted by the GP (mean function only). Clearly flat paths imply a lower predicted return whereas paths that start at a high or low level and move to a different level imply the GP sees a potentially profitable trade.

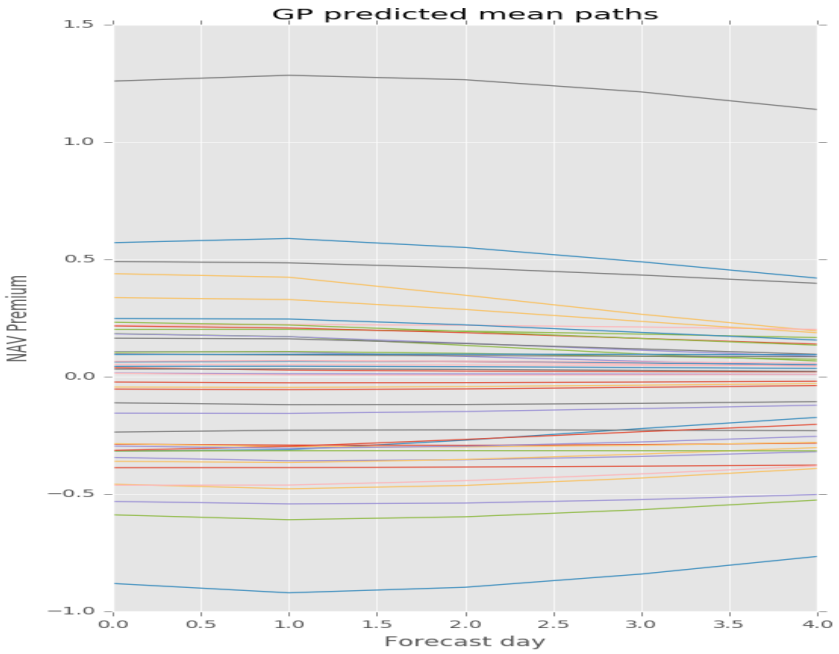

Figure 4.14: GP paths for NAV premium

4.7.3 Trade Implementation

Many thanks to Ahrash for his significant help on the trade implementation. At Zais' request, this part of the code is not included in the distributed code for the thesis.

Every day we had a trade decision to make, and every day we have from the GP a forecast of the NAV premium for the next four days as well as our uncertainty of that prediction for each of the four days. From this we constructed an expected annualised 'Sharpe' ratio for each of the next four days. (The ratio of expected return minus the risk free rate to expected volatility. Given it makes no difference to our trade decision we simply assumed risk free was zero).

Every day a trading decision is made. The basis for this decision is the maximum expected Sharpe, for the next four days. If this exceeds a predefined threshold including trade costs we will go long or short the NAV premium. The full round

trip trade costs are included in the initial decision making process. If we already hold a position, then the position is maintained if the maximum expected Sharpe remains positive. The position is reversed if the maximum expected Sharpe exceeds the Sharpe threshold, again after full round trip costs, but where the trade is of the opposite sign. In our case we simply set the threshold to zero, thus the GP traded whenever it felt it may be profitable, of course the threshold could be set to far more demanding predicted Sharpe ratios.

4.7.4 GP Benefits

The trade implementation thus makes direct use of some extra benefits of GPs. First multi-point predictions, together with the uncertainty about those predictions. This enables multi-point predictions of the Sharpe ratio, which including trading costs can be directly included into trade decision making. We also know at the outset how long we expect the trade to last. This latter information enables the model to include transaction costs in an innovative way by choosing trades with a longer expected duration and of course to trade less frequently. The Bayesian treatment also offers the extra possibility of optimising hyper-parameters using all the training data by maximising the Marginal Likelihood (as we did here), as well of traditional cross-validation.

4.7.5 Results EEM ETF

Unlike the more rigorous methodology in the previous experiments, results for this 'experiment' are really showing how one can use GPs and augmented data in a real life setting for decision making. The metric for our result is thus simply the realised profitability of the trading strategy together with the actual realised Sharpe ratio and recall that for reasons given at the outset, this arbitrage is not possible in practise.

Fig 4.15, shows the actual NAV premium for the whole time period in red, together with the GP prediction and its uncertainty, both in blue. Noteworthy is the

increased uncertainty of the GP during and after periods of high volatility

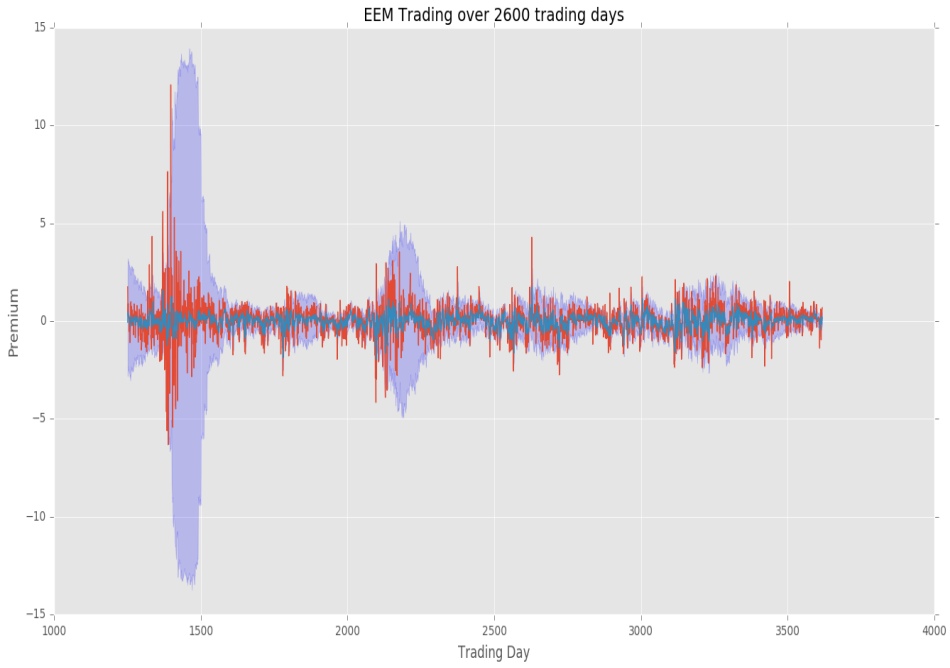

Figure 4.15: GP predictions for EEM NAV premium

Fig 4.16 shows the profitability of the strategy. The GP trades whenever it believes it will be profitable after costs. Transaction costs are set to an unrealistic zero cost. The strategy in this case has a realised Sharpe ratio of 6.4. Fantasy world for a daily trading strategy and a clear red flag to a practitioner, but as discussed the strategy is not actually one practitioners can implement.

Figure 4.16: Profitability of strategy - no costs

With zero trading costs, the GP expected to make a profitable trade every single trading day. The GP expected to hold for 1 day on 2263 occasions, 2 days on 3 occasions, 3 days 1 occasion and 4 days on 101 occasions. Thus with no costs the GP was happy to expect to make a huge number of one day trades and was always trading.

In Fig 4.17, we have now included full round trip costs of 75 bps (in reality trading costs would include commission, spread, market impact and time delays from idea to execution). The Sharpe ratio falls to 3.5. Still spectacular and unfortunately not possible in reality.

Noteworthy is the difference in profitability through time periods. Liquidity increased drastically through the 2000s and volatility levels changed drastically during the financial crisis.

Figure 4.17: Profitability of strategy - Round trip costs 75bps

With trading costs of 75bps, we can see the impact not just on profitability but also the decision making for the GP. It traded less often and expected trades to last longer. The GP did not trade at all for 1390 days, and expected trades to last 1 day on 867 occasions, the number of expected 4 day trades increased to 108. Thus the GP has automatically incorporated round trip transaction costs into its initial decision making and clearly migrated to trades with expected longer duration. It is more cautious.

In Fig 4.18, we again include trading costs of 75 bps. However this time although the GP forecasts expected Sharpe ratios, we do not include round trip transaction costs into its initial decision making. The impact of this is far worse profitability. The GP is too aggressive, makes worse trades and is essentially unprofitable for most of the period.

Figure 4.18: Profitability - Round trip costs 75bps, Costs not included in GP decision

Finally, in Fig 4.19, we show the length-scales through time of the Matern5/2 kernel for the day dimension and delta look-ahead dimension. The length-scale for the VIX was orders of magnitude higher and is not included. This highlights the benefits of ARD for feature selection. The GP simply did not need the VIX to predict the NAV premium (as the extremely high Sharpe ratio pre-costs demonstrates).

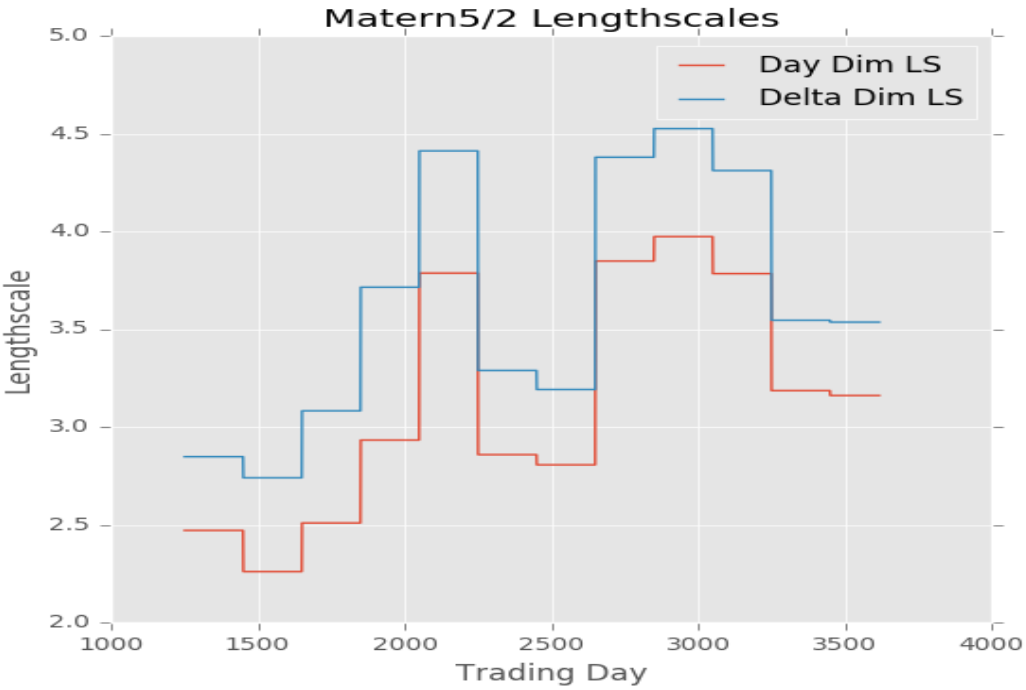

Figure 4.19: Matern5/2 lengthscales - Day/Delta

4.7.6 Premium to NAV SPY ETF

In contrast to our previous experiment where a 'time machine' was implicitly embedded in the strategy. The same strategy executed on the SPY ETF is completely realisable. The SPY ETF is a United States quoted ETF, with an underlying basket of U.S. stocks. Thus, there are no issues of different closes and different foreign exchange movements. We will run exactly the same strategy using our trained GPs on this ETF.

4.7.7 Results SPY ETF

Again our result metric is the realised profitability of the trading strategy together with the actual realised Sharpe ratio .

In Fig 4.20 we run the strategy at zero trading costs. The Sharpe ratio is a spectacular 5.6. Again we see the marked difference in profitability pre and post

crisis. However the GP has learned and predicted well. Clearly in real life, trading costs exist, so we shall do the same analysis as we did for the unrealisable EEM trade.

Figure 4.20: GP predictions for SPY NAV premium

In Fig 4.21 we rerun the strategy with a tiny 10 bps of round trip trading costs. The Sharpe ratio now falls to 1.0. Still respectable, but notice that the strategy has made no money post financial crisis and again that liquidity levels for SPY have changed drastically over time as the ETF became more popular. Thus, trading costs were likely higher pre-crisis. Despite the predictive ability of the GP. With only 10 bps of costs, all profits for the strategy have essentially been arbitraged away.

Figure 4.21: GP predictions for SPY NAV premium, 10bps trade costs

We see from our final experiments that GPs offer potential enhancements to trade strategies not just through the ability to incorporate multiple predictions into the future, but also its uncertainty about those predictions. This can expand trading strategies by including the direct forecasting of multiple expected Sharpe ratios after round trip transaction costs at the initial decision making stage. We showed extreme profitability for a very simple but unfortunately not realisable strategy as proof of principle and also that as costs increased the GP would choose fewer trades with a longer expected duration. Then, we showed how ARD can aid feature selection in practise. Finally we showed the GP at work on a realisable strategy in the markets, where it again showed again very strong performance, but that that performance is completely dependent upon the cost of trade execution.

Chapter 5

General Conclusions

This thesis was about long term forecasting of time series using Gaussian Processes. To that end the thesis first introduced familiar machinary and intuitions about GPs and then introduced and implemented the less familiar functional, augmented and functional-augmented data representations. Data augmentation multiplies the training set and thus scales up GPs, necessitating sampling - we implemented this and introduced signals to enable proper testing of augmentation. Finally we implemented a real life trading strategy utilising both the GP mean function prediction and its uncertainty about that prediction directly into the trade decision making process.

The motivation for the thesis came from Forecasting and Trading Commodity Contract Spreads with Gaussian Processes, and Augmented functional time series representation and forecasting with Gaussian processes, by Chapados and Bengio [6, 7], as well as Long Term Stock Market Forecasting using Gaussian Processes, [1], where the potential benefits of using GP's for financial forecasting seem very desirable. For example, having a prediction of the whole future distribution, a Bayesian treatment, the use of kernels, and the potential of maximising the marginal likelihood for hyper-parameter optimisation rather then / (or as well as) cross-validation.

The thesis shadows [6] to a certain extent, at least in terms of implementa-

tion. We vary when it comes to testing. We built a testbed to avoid the common uncertainties when forecasting time-series with a stochastic element. One can be guilty of viewing a prediction of a time series and noticing that it appears to match a realisation of a test set time-series (indeed [1], do precisely this). Chapados goes further and uses a valid test statistic. But really there is a whole possible distribution of outcomes, and we would ideally wish to see this. However with time series, by definition we get only one realisation of our process through time, with enough data and unchanging parameters this needn't be a problem of course. But this cannot be guaranteed.

Our approach to testing takes advantage of the fact we are simulating our time-series. We first engineered characteristics matching those one might have expected to find in [6]. In particular structure for seasonality through the years, and stochastic mean-reversion by adding an Ornstein Uhlenbeck process. For a single training example, we trained our GP and received a predictive distribution. Instead of comparing this to one test realisation, we instead simulated 1000 paths of the test set, from our prediction point. This gave us a distribution of possible outcomes which we could compare now to the distribution predicted by our GP. This was done many times to enable multiple training examples.

In real life financial time series there is often considerable noise and fat tails. Also one may be dealing with a reasonably new stock or instrument and thus have a small amount of history. We tested the representations introduced by Chapados against increasing noise, fatter tails and a short training history. Finally Chapados [6] used a rational quadratic kernel with automatic relevance discrimination, whereas [1], used a squared exponential kernel, neither elucidated a strong justification. These kernels actually have quite different characteristics, so we tested the different representations using different kernels. The reality is that a kernel is a very important prior to a GP, but in real life we have no guarantees we picked the right one for the data.

After all our experiments on simulated data, we moved on to real life problems. Trading the NAV premiums on the MSCI IShares ETFs. We showed how a GP can be used to make multiple forecasts, not just of future returns but also the risk associated with these returns. This gave us multiple expected Sharpe ratio forecasts which adjusted for round trip transaction costs, we used to implement a trading strategy.

5.1 Main Conclusions

5.1.1 Representations

We showed the functional representation is able to 'borrow' from previous years and augmentation was able predict more accurately than a standard GP using our features. In our experiments the functional-augmented representation was the best predictor, better than a standard GP with the same kernel and better than an augmented GP without the functional approach. The representation was also more robust under increasing noise and fatter tails.

There is a price to pay for both representations. Data augmentation, helps training features to predict multiple targets ahead, but it also squares ones data-set. A high price for GPs where an $O(N^3)$, matrix inversion is integral to their solution. With augmentation, we were forced to use sub-sampling.

The functional representation dissects years into independent inputs, enabling the GP to use information from previous years as it moves further from its training data in the current year. This is all very well, but it also imposes a very strong 'prior' on ones data. The GP is forced to believe that the last day of the previous year is about as far away as one can imagine from the first day of the next year, which is actually the very next day. The only thing that matters is how close our prediction point is to our current years training data and to previous years data around the same time of year. This may be true, but if it is seasonality we are after then can't

we let the data speak for itself and build a structured kernel that captures these same properties in the data if they exist, without imposing such sharp discontinuities. See Fig 2.2, where we introduced kernels and showed a simple structured kernel for the classic Mauna Loa CO2 data.

5.1.2 Noise

We calibrated the impact of noise on the representations and kernels. At high noise levels structure matters less and the OU process begins to dominate, also our signals may give false positives and be less certain. When the OU noise was above 0.4, a standard AR(1) predicted just as well or better than the Gaussian Processes (without an OU kernel), regardless of the data representation. GPs with an OU kernel of course predicted as well as an AR(1).

At high noise levels, the AR(1) would perform better than the GPs if our kernel didn't match our noise process (i.e an OU kernel). This makes logical sense, but also shows the importance of selecting the correct kernel in the first place. In Chapados [6], they used a Rational Quadratic kernel. We found this was not a bad choice and the RQ as well as the Matern3/2 under a functional-augmented representation were reasonably robust despite adding a large amount of OU noise.

The worst performing kernel was the squared exponential kernel. This is a very 'smooth' kernel and the one used by [1] for predicting the Hewlett Packard stock price. We were unable to replicate their images using the same data without adding additional constraints to the optimisation. To us it seems that OU noise is simply too rough for an RBF kernel to cope with. The experiments did however show the value of the functional-augmented approach even with this poor kernel choice, where the mean squared error of the prediction was sometimes improved by a factor of 10. Even with this improvement though the RBF kernel still lagged behind.

5.1.3 Small Datasets

We were reasonably surprised by how little data the models were able to learn from. Short time series of less than a couple of hundred data points led to no degradation in predictive power. With such short time series the functional collapses into the standard approach and the functional-augmented into the augmented (we have no prior years). But, here the augmented approach had an advantage. As we continued to shorten the training set, all the other models deteriorated, but the extra sampling used in the augmented approach made it more robust and it continued to predict well.

5.1.4 Fat Tails

Fat tails gave very similar test results to increasing noise. As noise increased we found a simple AR(1) model or a GP with an OU kernel suffices. There was little structure for the models left to grasp. We tested fat tails by deliberately choosing a volatility level where the GP (with Chapados' RQ kernel) outperformed an AR(1), but only just. As the tails got fatter, in a similar way to the volatility test the AR(1) began to dominate (for GPs without the OU kernel).

5.1.5 Trading Strategy

We showed some of the advantages of the GP approach in a real life trading strategy. Our model could look forward multiple periods and forecast an expected return for each period, together with its uncertainty about that return. This together with round trip transaction costs and a Sharpe ratio hurdle rate was used for trade decision making.

In addition the GP could give a forecast of how long it expected each trade to last - thus enabling trade decision making to be tailored not just to an expected return but also directly to costs and uncertainty at the outset. This is of course information rich compared to single period forecasts and forecasts without confidence

intervals. We showed that by doing this as trading costs increased the GP, explicitly chose fewer trades and trades of longer expected duration.

We also showed how ARD could be used for feature selection by demonstrating the VIX to be an unnecessary predictive feature in this example.

Finally we applied the GP to a realisable trading strategy and showed the level of round trip transactions costs that would arbitrage profits away.

5.1.6 Summary

For a practitioner the pessimistic answer for mean reverting time series if the noise level is very high (and in this context we have calibrated how high), has fat tails, and characteristics like an OU, that simple models suffice.

However it is not all gloom. We showed practical advantages to the functional-augmented approach, particularly with the correct kernel. First it does not need evenly spaced training data, which is integral to an AR(1) model. Second, if there is structure in a functional sense, or useful features in the augmented sense, then at lower noise levels the GP will learn these and predict far better than the simpler model.

In short, the GP is no worse at high noise levels and much better at low noise levels, and can cope with missing data. If rather than missing data, we have a very short training history, then augmentation proved to improve predictive power. Finally, if the kernel proves inappropriate for the test data, then if there is value in the features and the data really does suit a functional representation, then the functional-augmented representation can mitigate the poor predictive performance from using the wrong kernel.

As regards trading, the extra information provided by a GP. Not just a mean function forecast, but its uncertainty about that forecast at multiple future time pe-

riods, can be directly implemented into trade decision making, which we demonstrated. This opens new avenues to the trader, whereby the model can look multiple points into the future and assess not only when a trade should be executed, but how long it is likely to last and also whether it might be better to wait until a future moment before executing the trade. Where that trade is assessed not only by a return forecast, but its uncertainty about its own prediction.

5.2 Further Directions

5.2.1 Scaling

Beyond several thousand training points training GP's is slow and beyond tens of thousands not possible. Augmentation, which was key to our experiments, squares ones data-set. Not just in this thesis, but for other applications of GPs scaling up would be an advantage. This is an active area of research and we alluded to sparsity methods such as [25], and the very recent string GPs [29]. Our solution to scaling up, was subsampling, which introduced other issues such as how does one sample? This was unsatisfying, where what one really wishes to do was just let the GP optimisation handle the larger dataset.

5.2.2 Kernel Construction

We showed that kernel selection is important. Rasmussen and Williams [27], take the Fourier transform of a kernel and look at the eigenvalues, a more systematic way of framing smoothness. Duvenaud, 'Automatic model construction with Gaussian processes' [10] looks at methods of automatic kernel construction. To the author, it feels that kernel construction is a bit of a dark art, with heuristics to be learned. In this thesis the author dealt with fairly complex data representations using simple kernels, it would have been nice to have been able to let the data inform a perhaps more structured complex kernel in a semi-automatic way.

5.2.3 Extensions to Gaussian Processes

Given time the author would have liked to have implemented the t-processes first introduced by Shah, Wilson and Ghahramani [31]. It is claimed that they offer

advantages where there are changes in covariance structure. Something of interest in the finance domain.

Appendix A

Experiment 1- Noise full results

	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Sigma					
0.010000	0.220593	0.013694	0.003401	0.000437	0.002433
0.020000	0.219471	0.038941	0.008570	0.001714	0.004902
0.030000	0.213787	0.070448	0.015151	0.005015	0.006666
0.040000	0.212049	0.115376	0.021935	0.007283	0.008040
0.050000	0.212205	0.151248	0.030128	0.014104	0.010357
0.060000	0.207549	0.188496	0.039127	0.017914	0.012684
0.070000	0.209093	0.228439	0.051469	0.021387	0.015307
0.080000	0.203865	0.271618	0.065390	0.025550	0.018270
0.090000	0.199517	0.309322	0.079557	0.028767	0.021631
0.100000	0.200298	0.352120	0.097457	0.034476	0.026147
0.200000	0.185716	0.732964	0.314130	0.076224	0.126087
0.210526	0.187499	0.771415	0.340670	0.080409	0.138566
0.221053	0.187106	0.806492	0.372898	0.087120	0.155141
0.231579	0.196021	0.851621	0.404676	0.095209	0.179038
0.242105	0.179093	0.864284	0.423445	0.097789	0.195577
0.252632	0.177839	0.903170	0.456627	0.100341	0.220290
0.263158	0.185587	0.934761	0.485843	0.108817	0.230073
0.273684	0.180286	0.962230	0.511847	0.112848	0.262147
0.284211	0.175226	0.984338	0.535974	0.111792	0.275335
0.294737	0.176381	1.017258	0.562487	0.120672	0.301758
0.300000	0.168284	1.022669	0.575587	0.123905	0.298496
0.305263	0.172299	1.038199	0.591821	0.130197	0.314155
0.315789	0.162348	1.070852	0.621453	0.133737	0.338026
0.326316	0.166675	1.083216	0.579904	0.139669	0.344999
0.336842	0.196482	1.142052	0.627731	0.161482	0.438443
0.347368	0.189662	1.171762	0.657105	0.170569	0.395920
0.357895	0.188033	1.194631	0.690702	0.179509	0.467140
0.368421	0.183715	1.224806	0.704778	0.188498	0.476895
0.378947	0.193523	1.252238	0.731687	0.196191	0.479729
0.389474	0.199464	1.266732	0.740882	0.194419	0.471575
0.400000	0.181562	1.297105	0.791657	0.217592	0.500806
0.500000	0.176241	1.503204	0.998696	0.298238	0.518785
0.625000	0.209112	1.783785	1.078736	0.424253	0.605328
0.750000	0.173646	1.957816	1.132822	0.372649	0.619115
0.875000	0.194750	2.198240	1.010134	0.432247	0.680701
1.000000	0.200159	2.529615	1.119038	0.463764	0.716609

Table A.1: Mean Squared Error of Mean Function prediction - 10 day forecast

	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Sigma					
0.010000	0.730761	0.125793	0.010674	0.001232	0.012444
0.020000	0.726652	0.300263	0.028354	0.004690	0.026401
0.030000	0.709802	0.485444	0.050045	0.013906	0.030253
0.040000	0.703160	0.723298	0.073430	0.020358	0.032310
0.050000	0.700835	0.889943	0.100788	0.045114	0.040758
0.060000	0.686604	1.048449	0.132968	0.058478	0.046796
0.070000	0.684130	1.203994	0.173376	0.070901	0.053136
0.080000	0.675348	1.361548	0.221735	0.086385	0.061105
0.090000	0.659614	1.484911	0.267759	0.097415	0.072574
0.100000	0.666331	1.623360	0.328846	0.117545	0.087383
0.200000	0.624147	2.590166	1.006970	0.280834	0.436751
0.210526	0.629637	2.673973	1.088434	0.298508	0.478415
0.221053	0.625126	2.747645	1.182012	0.326844	0.520296
0.231579	0.642095	2.844578	1.267768	0.349921	0.604865
0.242105	0.592551	2.867349	1.328777	0.370450	0.684995
0.252632	0.584407	2.933050	1.408476	0.391612	0.760732
0.263158	0.611499	3.005809	1.494239	0.411606	0.811626
0.273684	0.594017	3.085741	1.593542	0.450187	0.905084
0.284211	0.588274	3.114346	1.651165	0.449530	0.939123
0.294737	0.591738	3.194663	1.733369	0.485911	1.003529
0.300000	0.581993	3.209241	1.771967	0.493819	1.027167
0.305263	0.566058	3.237208	1.814263	0.519938	1.042454
0.315789	0.570611	3.296449	1.882395	0.545950	1.119075
0.326316	0.563170	3.336818	1.753585	0.571354	1.150860
0.336842	0.625247	3.444392	1.862700	0.632728	1.346877
0.347368	0.608142	3.468509	1.906336	0.654077	1.246315
0.357895	0.604086	3.522415	2.010252	0.698883	1.418952
0.368421	0.581700	3.589429	2.040347	0.736412	1.453819
0.378947	0.594357	3.657866	2.116451	0.779099	1.463539
0.389474	0.615529	3.697010	2.140382	0.781148	1.479899
0.400000	0.565428	3.725004	2.253912	0.823390	1.518440
0.500000	0.565911	4.175917	2.777610	1.156415	1.609264
0.625000	0.655202	4.734133	2.886076	1.526918	1.781849
0.750000	0.592794	5.094599	2.990398	1.417119	1.827504
0.875000	0.641416	5.636119	2.575283	1.600615	1.997921
1.000000	0.739826	6.439397	2.871008	1.714489	2.179615

Table A.2: Mean Squared Error of Mean Function prediction - 20 day forecast

	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Sigma					
0.010000	1.328625	0.422355	0.017375	0.002027	0.039546
0.020000	1.324700	0.868882	0.047041	0.007384	0.080486
0.030000	1.298117	1.289022	0.083437	0.021553	0.084075
0.040000	1.281004	1.771459	0.124258	0.032055	0.079354
0.050000	1.273072	2.078365	0.172439	0.086053	0.087852
0.060000	1.246637	2.355559	0.228914	0.112125	0.097049
0.070000	1.242892	2.616996	0.299614	0.139083	0.106607
0.080000	1.227533	2.866891	0.384618	0.170329	0.121150
0.090000	1.206336	3.065151	0.468005	0.194087	0.146650
0.100000	1.217802	3.274248	0.572663	0.233313	0.180438
0.200000	1.136921	4.641310	1.740858	0.589166	0.883128
0.210526	1.144231	4.751961	1.875980	0.625783	0.958547
0.221053	1.135219	4.856267	2.030505	0.677128	1.041982
0.231579	1.164891	4.977360	2.168598	0.726893	1.194438
0.242105	1.099619	5.033595	2.295331	0.780604	1.372175
0.252632	1.078977	5.105959	2.420927	0.822970	1.501812
0.263158	1.110710	5.198233	2.561052	0.872374	1.593355
0.273684	1.066209	5.280926	2.707394	0.938774	1.726749
0.284211	1.070762	5.335843	2.825697	0.961047	1.797372
0.294737	1.068747	5.425262	2.944789	1.028202	1.879779
0.300000	1.081833	5.477919	3.030103	1.055764	1.961875
0.305263	1.028064	5.488411	3.075598	1.090334	1.946151
0.315789	1.039110	5.546207	3.172489	1.148831	2.060580
0.326316	1.018984	5.593255	2.937140	1.197320	2.104467
0.336842	1.115353	5.772579	3.126132	1.318033	2.426569
0.347368	1.091054	5.763221	3.172972	1.366952	2.248704
0.357895	1.073246	5.845071	3.352044	1.452798	2.532594
0.368421	1.031061	5.903218	3.364959	1.510725	2.568363
0.378947	1.068653	6.025229	3.501914	1.628658	2.602333
0.389474	1.101647	6.097813	3.553090	1.632893	2.653459
0.400000	1.028592	6.113700	3.724148	1.659701	2.686579
0.500000	1.019156	6.719878	4.508544	2.289826	2.830487
0.625000	1.170988	7.503816	4.630354	2.983760	3.095864
0.750000	1.098992	8.008021	4.794210	2.788637	3.174944
0.875000	1.136060	8.670080	3.949269	3.050445	3.358819
1.000000	1.337654	9.701094	4.303638	3.086880	3.595598

Table A.3: Mean Squared Error of Mean Function prediction - 30 day forecast

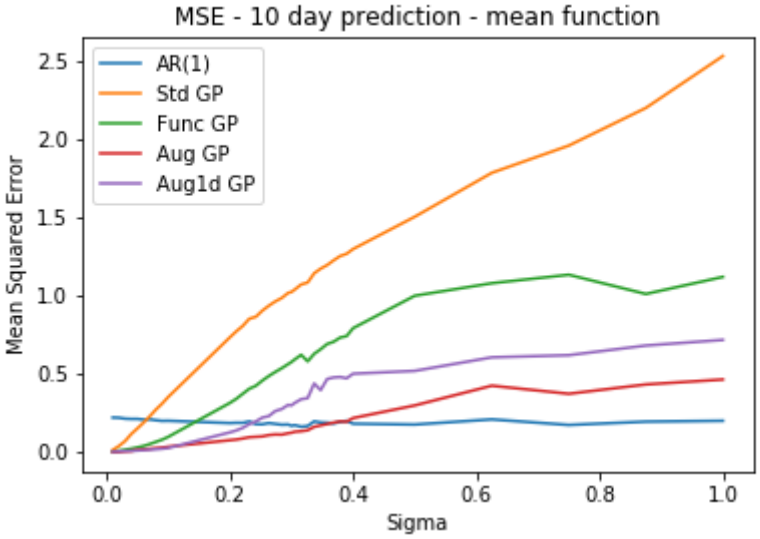

Figure A.1: Mean Squared Error, 10 day prediction

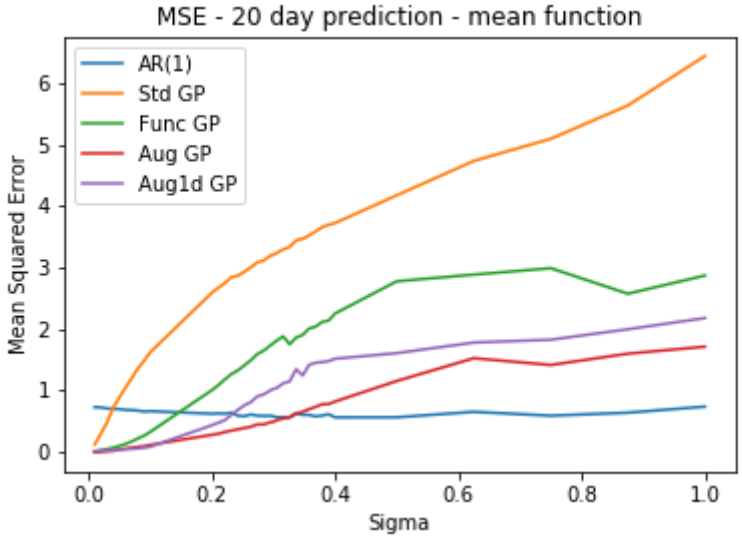

Figure A.2: Mean Squared Error, 20 day prediction

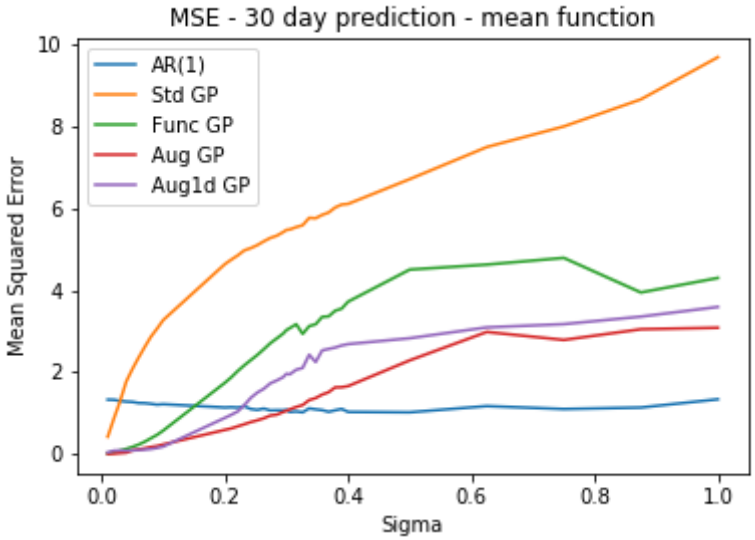

Figure A.3: Mean Squared Error, 30 day prediction

	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Sigma					
0.010000	0.019882	0.008430	0.001171	0.000203	0.001173
0.020000	0.015130	0.019081	0.004142	0.000619	0.003471
0.030000	0.011216	0.033274	0.007366	0.000991	0.004977
0.040000	0.009616	0.048161	0.010896	0.001431	0.006320
0.050000	0.007281	0.058509	0.014503	0.001791	0.007326
0.060000	0.006397	0.068298	0.018504	0.002225	0.008385
0.070000	0.005272	0.076734	0.021960	0.002677	0.009000
0.080000	0.004096	0.083080	0.025128	0.003094	0.008967
0.090000	0.003121	0.087238	0.027429	0.003487	0.009285
0.100000	0.003127	0.093146	0.031046	0.004182	0.009720
0.200000	0.001144	0.100871	0.047663	0.011502	0.009751
0.210526	0.001419	0.100236	0.049136	0.012070	0.010043
0.221053	0.000884	0.101026	0.051411	0.013459	0.010216
0.231579	0.000889	0.098292	0.051704	0.015080	0.010520
0.242105	0.001598	0.098561	0.053727	0.015736	0.011364
0.252632	0.001005	0.097381	0.054008	0.016329	0.012173
0.263158	0.001103	0.095825	0.055799	0.018059	0.011551
0.273684	0.001085	0.096314	0.057709	0.019856	0.012566
0.284211	0.000911	0.095546	0.058843	0.021316	0.012162
0.294737	0.001013	0.092859	0.058346	0.022420	0.013261
0.300000	0.000779	0.093091	0.059228	0.023651	0.012888
0.305263	0.001473	0.094644	0.061857	0.023802	0.013231
0.315789	0.000487	0.090601	0.059456	0.025568	0.012130
0.326316	0.001095	0.093757	0.064002	0.026405	0.013119
0.336842	0.000619	0.091505	0.063674	0.028350	0.013393
0.347368	0.000659	0.086379	0.059825	0.030593	0.012875
0.357895	0.000599	0.088912	0.062727	0.032124	0.013522
0.368421	0.001116	0.091008	0.066129	0.033948	0.014565
0.378947	0.001336	0.091764	0.066795	0.034138	0.014338
0.389474	0.000590	0.088593	0.064823	0.036947	0.014773
0.400000	0.001244	0.089461	0.068016	0.038328	0.015477
0.500000	0.001279	0.095360	0.080802	0.053158	0.023585
0.625000	0.003844	0.106170	0.091498	0.082800	0.041332
0.750000	0.003698	0.134119	0.127166	0.099858	0.061826
0.875000	0.004226	0.152653	0.136908	0.155557	0.096509
1.000000	0.006348	0.181395	0.168239	0.200559	0.125736

Table A.4: Mean Squared Error of SD Function prediction - 10 day forecast

	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Sigma					
0.010000	0.040019	0.078546	0.003753	0.000443	0.006684
0.020000	0.031106	0.133643	0.010156	0.001486	0.017111
0.030000	0.023102	0.184578	0.016217	0.002166	0.023850
0.040000	0.019472	0.221776	0.022229	0.003068	0.029362
0.050000	0.015689	0.239547	0.028030	0.004300	0.034670
0.060000	0.013043	0.249648	0.033457	0.005168	0.037106
0.070000	0.011157	0.255520	0.038063	0.005807	0.039274
0.080000	0.009769	0.258043	0.042750	0.006742	0.039761
0.090000	0.007136	0.253045	0.044573	0.006917	0.039487
0.100000	0.007787	0.254248	0.049298	0.008024	0.043115
0.200000	0.003859	0.173422	0.054248	0.013546	0.023261
0.210526	0.004040	0.164795	0.053494	0.012817	0.021124
0.221053	0.002828	0.157458	0.053881	0.014875	0.018657
0.231579	0.003559	0.151851	0.055241	0.015854	0.018861
0.242105	0.005035	0.145634	0.055638	0.017422	0.017868
0.252632	0.004002	0.138910	0.054477	0.016237	0.016439
0.263158	0.004206	0.133206	0.055747	0.017348	0.014782
0.273684	0.002605	0.120300	0.051934	0.019888	0.011702
0.284211	0.002438	0.118082	0.053917	0.020909	0.010872
0.294737	0.002281	0.106284	0.050543	0.020079	0.010083
0.300000	0.002533	0.107102	0.050771	0.022685	0.010640
0.305263	0.003471	0.109325	0.055586	0.022349	0.010520
0.315789	0.002417	0.100260	0.051146	0.023015	0.009356
0.326316	0.003189	0.099037	0.054190	0.022011	0.009262
0.336842	0.002529	0.095628	0.054298	0.023959	0.009381
0.347368	0.002308	0.085734	0.047570	0.025711	0.009250
0.357895	0.001483	0.082034	0.048057	0.026042	0.010964
0.368421	0.002142	0.082902	0.049585	0.028936	0.011624
0.378947	0.003415	0.084214	0.050629	0.027934	0.011841
0.389474	0.002086	0.075131	0.046310	0.029154	0.014018
0.400000	0.003105	0.074825	0.049867	0.032073	0.015450
0.500000	0.002334	0.060776	0.052006	0.043463	0.041387
0.625000	0.007363	0.068432	0.063761	0.075820	0.094918
0.750000	0.006685	0.092178	0.096045	0.087296	0.168542
0.875000	0.011688	0.131282	0.139606	0.175416	0.297023
1.000000	0.016342	0.167358	0.176634	0.224259	0.402403

Table A.5: Mean Squared Error of SD Function prediction - 20 day forecast

	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
Sigma					
0.010000	0.060333	0.237285	0.005381	0.000845	0.021779
0.020000	0.047302	0.333370	0.012688	0.002998	0.050822
0.030000	0.036349	0.397393	0.019314	0.004370	0.069451
0.040000	0.030141	0.428217	0.025427	0.006143	0.083383
0.050000	0.025012	0.435976	0.031389	0.009496	0.096689
0.060000	0.021156	0.432692	0.036868	0.011631	0.102195
0.070000	0.017665	0.420528	0.040399	0.012581	0.104544
0.080000	0.016667	0.411291	0.045394	0.014861	0.106389
0.090000	0.012621	0.391417	0.046504	0.015150	0.104170
0.100000	0.014410	0.382959	0.051164	0.017514	0.110499
0.200000	0.008548	0.205717	0.047291	0.023486	0.046154
0.210526	0.008113	0.188050	0.044789	0.021026	0.038568
0.221053	0.006724	0.178030	0.044962	0.024331	0.034375
0.231579	0.008488	0.168297	0.047100	0.025649	0.032285
0.242105	0.009283	0.155456	0.046134	0.026032	0.025699
0.252632	0.007830	0.144303	0.044298	0.023542	0.021258
0.263158	0.008522	0.135575	0.044721	0.024131	0.018907
0.273684	0.005870	0.119542	0.042434	0.027990	0.013890
0.284211	0.005016	0.112291	0.043417	0.027545	0.011570
0.294737	0.006139	0.100185	0.042485	0.027103	0.010056
0.300000	0.006102	0.099202	0.039996	0.027242	0.010380
0.305263	0.005993	0.099474	0.047092	0.028398	0.010895
0.315789	0.005442	0.088649	0.041872	0.028467	0.008929
0.326316	0.006151	0.085476	0.045193	0.026432	0.009519
0.336842	0.007044	0.082494	0.047673	0.030073	0.011041
0.347368	0.006239	0.070488	0.040001	0.029800	0.014082
0.357895	0.003086	0.064466	0.040832	0.028232	0.018864
0.368421	0.004492	0.065328	0.042222	0.032013	0.020973
0.378947	0.006416	0.066693	0.044248	0.032088	0.023275
0.389474	0.006006	0.058703	0.042589	0.030877	0.029875
0.400000	0.005715	0.057208	0.045698	0.034273	0.034233
0.500000	0.004916	0.052773	0.061100	0.041425	0.101551
0.625000	0.015282	0.091229	0.110076	0.083134	0.225535
0.750000	0.013844	0.163964	0.187364	0.104395	0.408562
0.875000	0.021678	0.259862	0.299217	0.210837	0.645399
1.000000	0.034774	0.376285	0.410858	0.290252	0.907243

Table A.6: Mean Squared Error of SD Function prediction - 30 day forecast

Appendix B

Experiment 3, Fat Tails - Full Results

Full results for the Fat tail experiment.

Degrees of Freedom	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
1000	0.230755	0.979164	0.730086	0.162114	0.357263
100	0.246249	1.17295	0.782307	0.200303	0.360546
50	0.237387	0.978202	0.756199	0.249668	0.432981
20	0.271697	2.764549	0.548723	0.353936	0.38144
15	0.269132	2.91509	0.673308	0.238837	0.463434
5	0.285076	0.98209	0.575406	0.359739	0.583631
3	0.236719	1.862816	1.317582	0.548689	0.763959
2	263.41	32653.4	76371.9	740.3	48116.9

Table B.1: RQ- Sigma=0.38, Student-t - 10 day forecast Mean Function

Degrees of Freedom	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
1000	0.001789	0.116764	0.097042	0.048652	0.020827
100	0.001256	0.109116	0.088696	0.044788	0.018523
50	0.001643	0.097567	0.082645	0.05625	0.016674
20	0.001624	0.117463	0.099621	0.050229	0.022806
15	0.001814	0.122605	0.109712	0.043295	0.02184
5	0.148879	0.054682	0.053846	0.092781	0.034454
3	0.436201	0.688179	0.665197	0.381423	0.244926
2	1.143485	1.362697	1.329857	1.65723	1.482673

Table B.2: RQ- Sigma=0.38, Student-t - 10 day forecast Std

Degrees of Freedom	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
1000	0.744355	3.0712	2.27584	0.638833	1.201206
100	0.764476	3.526791	2.342166	0.64193	1.193606
50	0.814635	3.108013	2.420939	0.850552	1.458465
20	0.271697	2.764549	0.548723	0.353936	0.38144
15	0.92146	2.91509	1.98096	0.815004	1.416591
5	0.966406	2.946265	1.810121	1.09672	1.748844
3	0.806526	4.91751	3.431405	1.823446	2.174833
2	747.1	57730.3	110208	6028	78702

Table B.3: RQ- Sigma=0.38, Student-t - 20 day forecast Mean Function

Degrees of Freedom	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
1000	0.004191	0.0.119411	0.088466	0.052952	0.015466
100	0.004687	0.11814	0.084742	0.052722	0.013656
50	0.000501	0.098231	0.077081	0.063401	0.014113
20	0.005421	0.121857	0.099621	0.066593	0.023381
15	0.004217	0.121808	0.109712	0.047585	0.017705
5	0.305415	0.041169	0.041087	0.103773	0.093517
3	0.752925	0.689267	0.642053	0.429857	0.355513
2	1.840288	1.920719	1.953693	2.870584	2.789316

Table B.4: RQ- Sigma=0.38, Student-t - 20 day forecast Std

Degrees of Freedom	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
1000	1.344587	5.23551	3.885728	1.37779	2.269541
100	0.764476	3.526791	2.342166	0.64193	1.193606
50	1.530333	5.39021	4.225021	1.631246	2.74023
20	1.626072	4.700875	2.958772	2.253778	2.443279
15	1.701395	4.976242	3.383994	1.587026	2.619834
5	1.824731	5.059022	3.200931	1.98966	3.130902
3	0.806526	4.91751	3.431405	1.823446	2.174833
2	2235	67620	119767	14076	95614

Table B.5: Matern3/2- Sigma=0.38, Student-t - 30 day forecast Mean Function

Degrees of Freedom	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
1000	0.01001	0.0.101816	0.075019	0.0675548	0.017877
100	1.373689	5.953987	3.963088	1.242159	2.287314
50	0.009352	0.082645	0.065173	0.060628	0.020561
20	0.012969	0.107505	0.080363	0.081297	0.03432
15	0.00701	0.101639	0.086985	0.061086	0.024971
5	0.441676	0.069454	0.072077	0.10733	0.203371
3	1.477403	7.652786	5.368949	3.157255	3.723196
2	2.463715	2.923867	3.011824	3.847704	4.381655

Table B.6: RQ- Sigma=0.38, Student-t - 30 day forecast Std

Appendix C

Experiment 4, Small Data - Full Results

Full results for the small data experiment.

Obs Year	Obs Day	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
2	50	0.020602	0.573366	0.469432	0.087942	0.177772
1	200	0.168553	0.113113	0.252316	0.086598	0.075516
1	150	0.150945	1.109502	0.898004	0.233877	0.293358
1	100	0.428651	0.316786	0.480829	0.225174	0.278258
0	150	0.218372	1.318972	1.319897	0.293534	0.293641
0	100	1.894154	0.39154	0.391543	0.285445	0.285234
0	50	1.609344	0.449006	0.448999	0.182029	0.179624

Table C.1: RQ- Sigma=0.38, OU - 10 day forecast MSE - Small Data

Obs Year	Obs Day	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
2	50	0.0086741	1.383307	1.151605	0.226623	0.327411
1	200	0.194001	0.544586	0.256715	0.256715	0.241836
1	150	0.488949	3.248612	2.529134	0.766008	0.988561
1	100	1.74482	1.409138	0.480829	0.855082	1.081898
0	150	0.776458	3.740487	3.740489	0.988972	0.990674
0	100	5.890957	1.349572	1.349574	1.247833	1.247454
0	50	3.513869	0.949658	0.949644	0.318234	0.31382

Table C.2: RQ- Sigma=0.38, OU - 20 day forecast MSE - Small Data

Obs Year	Obs Day	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
2	50	0.217699	1.884776	1.648823	0.392047	0.396859
1	200	2.111592	0.241403	0.749169	0.579606	0.656602
1	150	0.856949	5.345155	4.081297	1.38428	1.836873
1	100	3.991562	2.454904	2.63404	1.934716	2.528708
0	150	1.647748	6.096296	6.096301	1.839766	1.842913
0	100	11.131405	2.901266	2.901266	2.941594	2.941013
0	50	5.584797	1.178199	1.17818	0.369026	0.36372

Table C.3: RQ- Sigma=0.38, OU - 30 day forecast MSE - Small Data

Obs Year	Obs Day	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
2	50	0.002175	0.104336	0.072829	0.031347	0.019039
1	200	0.002098	0.104108	0.073451	0.023038	0.019935
1	150	0.002211	0.09925	0.067789	0.025226	0.02176
1	100	0.001892	0.086778	0.057845	0.031237	0.025189
0	150	0.004579	0.08647	0.086471	0.032115	0.032055
0	100	0.015186	0.053324	0.053325	0.055492	0.055471
0	50	0.02796	0.085667	0.085667	0.069743	0.069866

Table C.4: RQ- Sigma=0.38, OU - 10 day forecast STD - Small Data

Obs Year	Obs Day	AR(1)	Std GP	Func GP	Aug GP	Aug GP-1D
2	50	0.005767	0.114354	0.066373	0.042055	0.028056
1	200	0.005715	0.116186	0.0715	0.041603	0.032019
1	150	0.006381	0.106553	0.067789	0.050069	0.038358
1	100	0.005861	0.093886	0.065543	0.064255	0.043737
0	150	0.006381	0.106553	0.067789	0.050069	0.0328358
0	100	0.005861	0.093886	0.065543	0.064255	0.043737
0	50	0.114718	0.148598	0.148596	0.182355	0.18263

Table C.5: RQ- Sigma=0.38, OU - 20 day forecast STD - Small Data

Appendix D

Colophon

(Gaussian Process packages) We used both the GPy package from the Sheffield machine learning group and GPFlow [22]. For coding we used Python 3.6, together with the Anaconda environment. For optimisation purposes the package uses the SciPy implementation of L-BFGS. Small scale optimisations were performed on a quad core 2.8Ghz laptop. Larger optimisations were performed using cloud computing resources provided by Zais Group, who the author wishes to thank.

Bibliography

[1] Authors, A. (2010). Long term stock market forecasting using gaussian processes. Technical report.

[2] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[3] Björk, T. (2009). *Arbitrage theory in continuous time*. Oxford university press.

[4] Chan, E. (2009). *Quantitative trading: how to build your own algorithmic trading business*, volume 430. John Wiley & Sons.

[5] Chan, E. (2013). *Algorithmic trading: winning strategies and their rationale*. John Wiley & Sons.

[6] Chapados, N. and Bengio, Y. (2007). Forecasting and trading commodity contract spreads with gaussian processes. In *13th International Conference on Computing in Economics and Finance*.

[7] Chapados, N. and Bengio, Y. (2008). Augmented functional time series representation and forecasting with gaussian processes. In *Advances in neural information processing systems*, pages 265–272.

[8] Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431.

[9] Duvenaud, D. The kernel cookbook. `http://www.cs.toronto.edu/~duvenaud/cookbook/index.html`. Accessed: 2017-08-02.

[10] Duvenaud, D. (2014). *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge.

[11] Farrell, M. T. and Correa, A. (2007). Gaussian process regression models for predicting stock trends. *Relation*, 10:3414.

[12] Ghoshal, S. and Roberts, S. (2016). Extracting predictive information from heterogeneous data streams using gaussian processes. *Algorithmic Finance*, 5(1-2):21–30.

[13] Gibson, R. and Schwartz, E. S. (1990). Stochastic convenience yield and the pricing of oil contingent claims. *The Journal of Finance*, 45(3):959–976.

[14] Gillespie, D. T. (1996). Exact numerical simulation of the ornstein-uhlenbeck process and its integral. *Physical review E*, 54(2):2084.

[15] Girard, A., Rasmussen, C. E., Candela, J. Q., and Murray-Smith, R. (2003). Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Advances in neural information processing systems*, pages 545–552.

[16] Grinold, R. C. and Kahn, R. N. (2000). Active portfolio management.

[17] Hamilton, J. D. (1994). *Time series analysis*, volume 2. Princeton university press Princeton.

[18] Hastie, T., Tibshirani, R., and Friedman, J. (2002). The elements of statistical learning: Data mining, inference, and prediction. *Biometrics*.

[19] Johansen, S. (1991). Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica: Journal of the Econometric Society*, pages 1551–1580.

[20] MacKay, D. J. (1999). Comparison of approximate methods for handling hyperparameters. *Neural computation*, 11(5):1035–1068.

[21] Markowitz, H. (1952). Portfolio selection. *The journal of finance*, 7(1):77–91.

[22] Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., Leon-Villagra, P., Ghahramani, Z., and Hensman, J.

[23] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

[24] Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

[25] Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959.

[26] Ramsay, J. O. and Silverman, B. W. (2002). *Applied functional data analysis: methods and case studies*, volume 77. Springer New York.

[27] Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT press Cambridge.

[28] Samo, Y.-L. K. and Roberts, S. (2015). Generalized spectral kernels. *arXiv preprint arXiv:1506.02236*.

[29] Samo, Y.-L. K. and Roberts, S. J. (2016). String and membrane gaussian processes. *The Journal of Machine Learning Research*, 17(1):4485–4571.

[30] Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

[31] Shah, A., Wilson, A., and Ghahramani, Z. (2014). Student-t processes as alternatives to gaussian processes. In *Artificial Intelligence and Statistics*, pages 877–885.

[32] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

[33] Shi, J. Q. and Choi, T. (2011). *Gaussian process regression analysis for functional data*. CRC Press.

[34] Shreve, S. E. (2004). *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media.

[35] Smith, W. (2010). On the simulation and estimation of the mean-reverting ornstein-uhlenbeck process. *Commodities Markets and Modelling*.

[36] Snelson, E. L. (2008). *Flexible and efficient Gaussian process models for machine learning*. University of London, University College London (United Kingdom).

[37] Swastanto, B. A. (2016). Gaussian process regression for long-term time series forecasting.

[38] Uhlenbeck, G. E. and Ornstein, L. S. (1930). On the theory of the brownian motion. *Physical review*, 36(5):823.

[39] van den Berg, T. Calibrating the ornstein-uhlenbeck (vasicek) model. `https://www.sitmo.com/?p=134`. Accessed: 2017-08-02.

[40] Vasicek, O. (1977). An equilibrium characterization of the term structure. *Journal of financial economics*, 5(2):177–188.

[41] Williams, C. K. (1998). Prediction with gaussian processes: From linear regression to linear prediction and beyond. *Nato asi series d behavioural and social sciences*, 89:599–621.

[42] Williams, C. K. and Rasmussen, C. E. (1996). Gaussian processes for regression. In *Advances in neural information processing systems*, pages 514–520.

[43] Williams, C. K., Rasmussen, C. E., Schwaighofer, A., and Tresp, V. (2002). Observations on the nyström method for gaussian processes.

[44] Williams, C. K. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688.

[45] Yan, W., Qiu, H., and Xue, Y. (2009). Gaussian process for long-term time-series forecasting. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 3420–3427. IEEE.