Indian Institute of Technology, Kanpur

MTH 517

Course Project

Bayesian Time Series Modelling

Author:
Shubhanshu Khandelwal(15807705)
Chitrak Raj Gupta(15917207)
Kumar Raj (160350)
Hemant Singh (150286)

Instructor:
Prof. Amit Mitra

November 12, 2019

Contents

1 Introduction

A time series is a sequential set of data points which are arranged in chronological order. We define it mathematically as a set of vectors x(t),t = 0,1,2,... where t represents the time elapsed. Here the variable x(t) is treated as a random variable. A time series is made up of 4 components viz Trend, Cyclical, Seasonal and Noise components. The models used to analyze these components are either assumed to be multiplicative or additive.

There are various types of stochastic models([1]) used to study time series. White Noise(WN) processes, Moving average processes(MA), Auto regressive processes(AR) and Auto regressive Moving average(ARMA) are linear models, and are the simplest of them and have been studied most deeply. Moving upward we encounter ARIMA(Auto regressive Integrated Moving average) and SARIMA(Seasonal ARIMA) which are still Linear models.

Adding one more level of sophistication, we have Non-Linear models like ARCH(Autoregressive Conditional Heteroskedasticity), GARCH(Generalized ARCH) and EGARCH(Exponential GARCH), which are quiet ubiquitous in Financial data modelling.

Next in the hierarchy comes Artificial Neural Networks(ANN) which are further Non-linear models and are motivated by biological neural network. The basic idea behind ANN is to have input as some previous values of time series($y_{t-1}, y_{t-2}, \ldots, y_{t-p}$) pass through some non-linear transformation giving a hidden layer which is then used to predict the output y_{t-1}. Some of the ANN's that are commonly used are TLNN(Time Lagged Neural Networks) and SANN(Seasonal Artificial Neural Networks).

The parameters in all of these models can be estimated using MLE and MAP. However it is sometimes more useful to have distribution of parameters which can then be used to predict the output with uncertainty i.e. with some confidence interval. In those cases we perform bayesian estimation.

2 Bayesian Time Series Modelling

Bayesian Modelling comes very handy when we not only want to do prediction but also want to know the uncertainty of prediction, since when uncertainty is known we can say that there is only a particular amount of surety associated with the prediction. This feature would come very handy when we want to know about the stock, will it increase or decrease and with what surety.

There are 3 main concepts associated with Bayesian Modelling. First is prior (the prior belief we have about the parameters, $p(\theta)$), Second is likelihood (what is the probability we have with a particular instance of parameters, $p(y|\theta)$) and the third is posterior (the belief we have after we have collected all the data, $p(\theta|y)$). It is sometimes straightforward to have the posterior when the prior and the likelihood are conjugate. But most of the times this does not happen and we have to stick to approximate methods.

Various methods are employed to do Bayesian modelling. Some of them are Expectation maximization, Variational Bayes, Markov Chain Monte Carlo Methods (MCMC) which are all parametric methods. Some Non-parametric modelling techniques([3]) also exist which

include Gaussian Processes, Poisson Processes, Dirichlet processes etc. Time series modelling involves regression, for which Gaussian Processes are particularly suitable.

2.1 Bayesian Modelling - A case of AR(p) process

A Linear Dynamical System(LDS)([2]) on variables $x_{1:T}, y_{1:T}$ has the following form:

$$\begin{aligned} x_t &= Ax_{t-1} + \bar{x}_t + \epsilon_t^x, \quad \epsilon_t^x \sim \mathcal{N}(\epsilon_t^x|0, Q), \quad x_1 \sim \mathcal{N}(x_1|\mu, P) \\ y_t &= Cx_t + \bar{y}_t + \epsilon_t^y, \quad \epsilon_t^y \sim \mathcal{N}(\epsilon_t^y|0, R) \end{aligned} \tag{1}$$

with transition matrix A and emission matrix C. The terms $\bar{x}_t, \bar{y}_t$ are often defined as $\bar{x}_t = Bz_t$ and $\bar{y}_t = Dz_t$, where z_t is a known input that can be used to control the system. Thus, the complete parameter set is therefore $\{A, B, C, D, Q, R, \mu, P\}$.
In particular, AR models can be formulated in the LDS form as:

$$\underbrace{\begin{pmatrix} y_t \\ y_{t-1} \\ \vdots \\ y_{t-m+1} \end{pmatrix}}_{x_t} = \underbrace{\begin{pmatrix} a_1 & a_2 & \dots & a_m \\ 1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} y_{t-1} \\ y_{t-2} \\ \vdots \\ y_{t-m} \end{pmatrix}}_{x_{t-1}} + \underbrace{\begin{pmatrix} \epsilon_t \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{\epsilon_t^x},$$

$$y_t = \underbrace{\begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}}_{C} x_t + \epsilon_t^y,$$

where, $y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_m y_{t-m} + \epsilon_t$ is an AR(m) process.
Thus, we have transformed an m_{th}-order Markov model into a constrained first-order latent Markov model. Now, we can use EM or variational Bayes or MCMC algorithms can be used to infer the predictions and the associated parameters like A and C.

2.2 Why we chose GP

Gaussian Processes have several advantages in modelling Time Series. Some of them are-

1. First and foremost advantage is that GP can be used for modelling data in Bayesian sense, no matter how much complicated the data is. Further, the implementation is also easier and flexible.
2. Given the values of their hyper-parameters(which can even be estimated via MLE and MAP), Gaussian Processes like other kernel methods, can be optimized exactly, and they allow for accurate trade-off between bias(fit) and variance(smoothness) in the fitting.

3. On small datasets, like time series, they are very good because of this well-tuned smoothing and because they are still computationally affordable. Even if the dataset is large, there are many techniques in the literature which can be used to remove the redundant data from the train set(like Orthogonal matching pursuit and small rank approximations of kernel matrix etc), consequently reducing the complexity. This advantage makes GP more handy to use than Neural Net which are harder to train(However Now-a-days automatic differentiation techniques and other libraries make ANN more convenient and user-friendly).

3 Gaussian Process

We have seen that modelling with Gaussian processes([4]) covers non-linear functions also. Further, it also ensures modelling of the noise component.
A Gaussian Process,

$$f \sim \mathcal{GP}(\mu, \kappa) \tag{2}$$

defines a distribution over functions, where μ is the mean function and κ is the covariance/kernel function.

- Mean function μ models the "average" function f from $\mathcal{GP}(\mu, \kappa)$.
- Covariance function κ models "shape/smoothness" of these functions.

For $f \sim \mathcal{GP}(\mu, \kappa)$, f' s values at any finite set of input $x_1, \ldots, x_N$ are jointly Gaussian

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_N) \end{bmatrix}, \begin{bmatrix} \kappa(x_1, x_1) \ldots \kappa(x_1, x_N) \\ \kappa(x_2, x_1) \ldots \kappa(x_2, x_N) \\ \vdots \\ \kappa(x_N, x_1) \ldots \kappa(x_N, x_N) \end{bmatrix} \right)$$

Compactly, $p(\mathbf{f}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$, where $\boldsymbol{f}$ and $\boldsymbol{\mu}$ are $N \times 1$ and $\mathbf{K}$ is $N \times N$
Now, we will compute $f_* = f(x_*)$ for a new input x_*. To see this, note that for $\mu = 0$

$$p\left(\begin{bmatrix} f \\ f_* \end{bmatrix} \right) = \mathcal{N}\left(\begin{bmatrix} f \\ f_* \end{bmatrix} | \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{K} & k_x \\ k_x^\top & \kappa(x_*, x_*) \end{bmatrix} \right)$$

where $\mathbf{k}_x = [\kappa(x_*, x_1), \ldots, \kappa(x_*, x_N)]^\top$
Now we can apply the Gaussian conditioning and the marginal distribution expressions to get $p(f_* | \boldsymbol{f}) = \mathcal{N}(\mu_*, \sigma_*^2)$, where

$$\begin{aligned} \mu_* &= \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{f} \\ \sigma_*^2 &= \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_x^\top \mathbf{K}^{-1} \mathbf{k} \end{aligned}$$

Here, if we have no trend in the data then $\mu = 0$, for the case when there is trend present then the related description is given in next subsection. So, the major challenge is the kernel function which in reality depends primarily on our time series given. In the following subsection we briefly present some commonly used kernels which are suitable for analysis of time series. An important point to note is that sums (and products) of valid covariance

kernels give valid covariance functions (i.e. the resultant covariance matrices are positive (semi-) definite) and which comes in handy when the data shows complex behaviour or a combination of multiple curves. However it comes with no free lunch. The price paid lies in the extra complexity of handling the increased number of hyperparameters.

3.1 Estimation of trend

There are 2 possible ways to extract the trend component

1. *MLE*: If we have a trend component we could extract it first before modelling. Let $\mathbf{K}$ be the associated kernel matrix, y_N be the vector of observations $[y_1, y_2, \ldots, y_N]'$ and let μ be given by $T\beta$, where T is the matrix of features such that $T[i,:] = [t_{i,1}, t_{i,2}, \ldots, t_{i,d}]$ where d is the order of the polynomial fit that is desired. β is d-dimensional vector. Now the MLE estimate of β is -

$$\begin{aligned}\beta_{MLE} &= \arg\min_{\beta}(y_N - T\beta)^\top \mathbf{K}^{-1}(y_N - T\beta) \\ &= (T^\top \mathbf{K}^{-1} T)^{-1} T^\top \mathbf{K}^{-1} y_N\end{aligned}$$

2. Kernel approach: We can instead consider a polynomial kernel(described in next subsection) in addition with the kernel associated with the seasonal kernel and do the inference of GP as usual to incorporate the polynomial trend.

To determine the order of polynomial d we will use Z-test. Here, we are estimating under 95% confidence interval.

3.2 Kernel Functions

A valid covariance/Kernel function under any arbitrary (smooth) map remains a valid covariance function. For any function $u : x \to u(x)$, a covariance function $k()$ defined over the range of x gives rise to a valid covariance $k'()$ over the domain of u. Thus we can use simple, stationary covariance in order to construct more complex (possibly non-stationary) covariance. So basic kernel functions are:

- The squared exponential (SE) or radial basis function(rbf) kernel is given by:

$$\kappa(x_i - x_j) = h^2 \exp(-(x_i - x_j)^2/\lambda)$$

  here h is an output-scale amplitude and λ wavelength parameter. If λ is larger then the function modeled by GP is smooth and if it is small then the function can capture a swiftly varying data. Functions drawn from this type of GP are infinitely differentiable.

- The periodic kernel is given by:

$$k_{\text{per}-\text{SE}}(x_j, x_j; h, w, T) = h^2 \exp\left(-\frac{1}{2w^2}\sin^2\left(\pi\left|\frac{x_j - x_j}{T}\right|\right)\right) \tag{3}$$

  In this case the output scale h serves as the amplitude and T is the period. The hyperparameter w is a "roughness" parameter that serves a similar role the input scale

λ in rbf kernels. With this formulation, we can perform inference about functions of arbitrary roughness and with arbitrary period.

- The polynomial kernel is given by:

$$K(x, y) = (\langle x, y \rangle + c)^d \tag{4}$$

where x and y are vectors in the input space, i.e. vectors of features computed from training or test samples and $c \geq 0$ is a free parameter trading of the influence of higher-order versus lower-order terms in the polynomial.

- The rational quadratic (RQ) kernel is given by:

$$k_{\mathrm{RQ}}(x_i, x_j) = h^2 \left(1 + \frac{(x_i - x_j)^2}{\alpha \lambda^2}\right)^{-\alpha}$$

where α is known as the index. This is equivalent to a scale mixture of squared exponential kernels with different length scales, the latter attributed according to a Beta distribution with parameters α and λ^{-2}. This gives variations with a range of time-scales, the distribution peaking around λ but extending to significantly longer period (but remaining rather smooth). When $\alpha \to \infty$, the RQ kernel reduces to the SE kernel with length scale λ.

4 Simulations

To check the performance of GP, we run three simulations on synthetic dataset and then one using a real life dataset for evaluation.

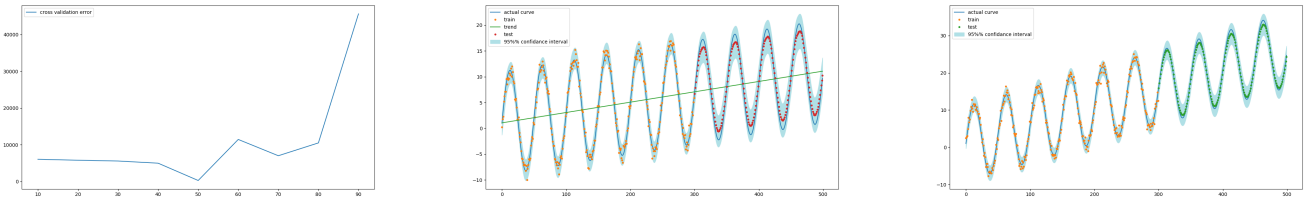
4.1 Sinusoidal seasonality and linear trend

$$y_t = m_t + s_t + e_t \tag{5}$$

where

$$m_t = a + b * t; \qquad s_t = 10 * sin(\frac{2 * \pi * t}{k}); \qquad \epsilon_t = N(0, \sigma^2)$$

We ran cross-validation for seasonality and found the error for validation dataset to be least for the desired value of 50 only.



(a) Cross Validation Error for s (b) Prediction for trend using kernels (c) Prediction(trend using MLE)

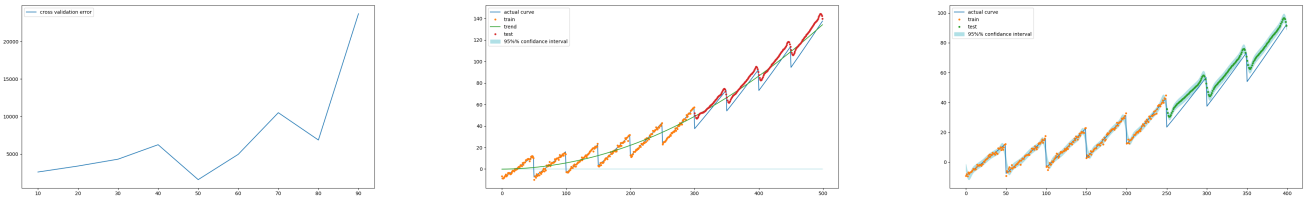
Figure 1: Sinusoidal Seasonality with Linear Trend

4.2 Sawtooth seasonality and quadratic trend

$$y_t = m_t + s_t + e_t \tag{6}$$

where

$$m_t = a + b * t + c * t^2; \qquad s_t = 10 * \text{sawtooth}(\frac{2 * \pi * t}{k}); \qquad \epsilon_t = N(0, \sigma^2) \tag{7}$$

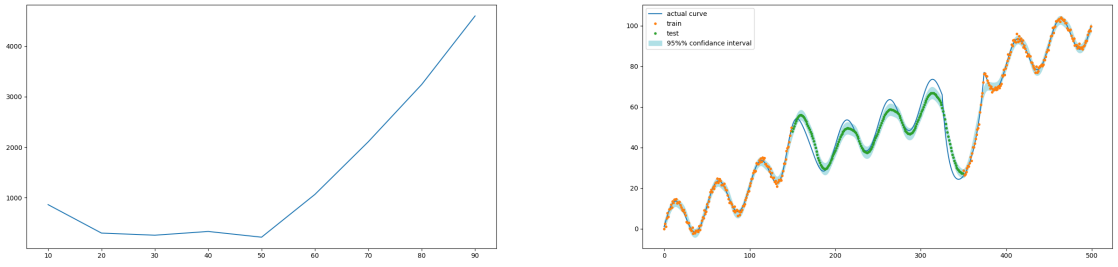


(a) Cross Validation Error for s (b) Prediction for trend using kernels (c) Prediction

Figure 2: Cross-validation and prediction for quadratic-sawtooth data(synthetic dataset)

4.3 Sinusoidal seasonality, linear trend and local trend

In this model, we take a sinusoidal linear sample and add local trends in them.

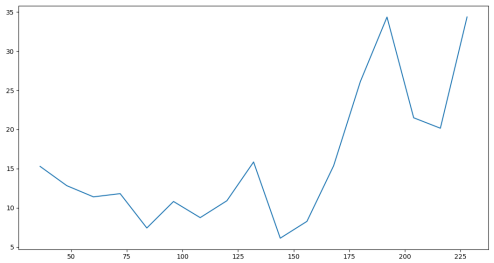


(a) Cross Validation Error for seasonality (b) Prediction

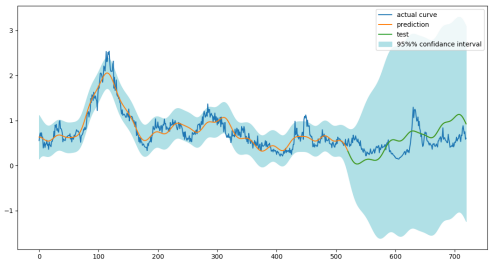
Figure 3: Cross-validation and prediction for linear-sinusoidal with random bumps in between(for local trend) data(synthetic dataset)

4.4 Real Dataset

We use the tide height data of 10 days obtained from chimet.co.uk. The data provided is sampled once every 5 minutes. We Take samples once every 20 minutes by averaging 4 points.



(a) Cross Validation Error for seasonality



(b) Prediction

Figure 4: Cross-validation and prediction for real-world dataset

For trend component, $\mid \hat{z_1} \mid = 0.79 < 1.96$. Hence, at 95% significance, we reject linear component. $m_t = 0.7258$.

5 Future Work

In this project we could only consider time series with white noise stochasticity. However we would like to extend our knowledge and simulations to include arbitrary noise models like AR, MA and ARMA. Also, we only tried simulations for additive models using kernel additive property, for multiplicative models GP with kernels multiplied with each other can be tried.

** Link to Code: https://github.com/chitrak7/Gaussian-Processes-Time-Series

References

[1] Ratnadip Adhikari and Ramesh K Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.

[2] David Barber, A Taylan Cemgil, and Silvia Chiappa. *Bayesian time series models*. Cambridge University Press, 2011.

[3] Samuel J Gershman and David M Blei. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

[4] Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, 2013.