ISYE-6420 Course Project

Seasonal Time Series Prediction With Gaussian Process

Weijia Zhao

This version: April 24, 2022

1 Introduction

Seasonality is a common feature of time series data in real life as many of the events experience regular and predictable changes over time. Examples include financial series(eg: corn futures price, volatility data), demographic series (eg: birth records), geophysical data (eg: temperature, sea level), and many others. In traditional (frequentist) statistics, we reply on time series regression techniques such as ARIMA model, GARCH model, or filtering techniques such as STL decomposition, Hodrick-Prescott filter to analyze such time series and make predictions. In the meantime, the cycle is in many cases large compared to the sample size for such time series (especially true for social-economic data), making the prediction with traditional analysis less accurate and robust (as lagging will further decrease the sample size can be used in regressions). In contrast, Gaussian Process is less subject to such problems and might be used to generate good predictions.

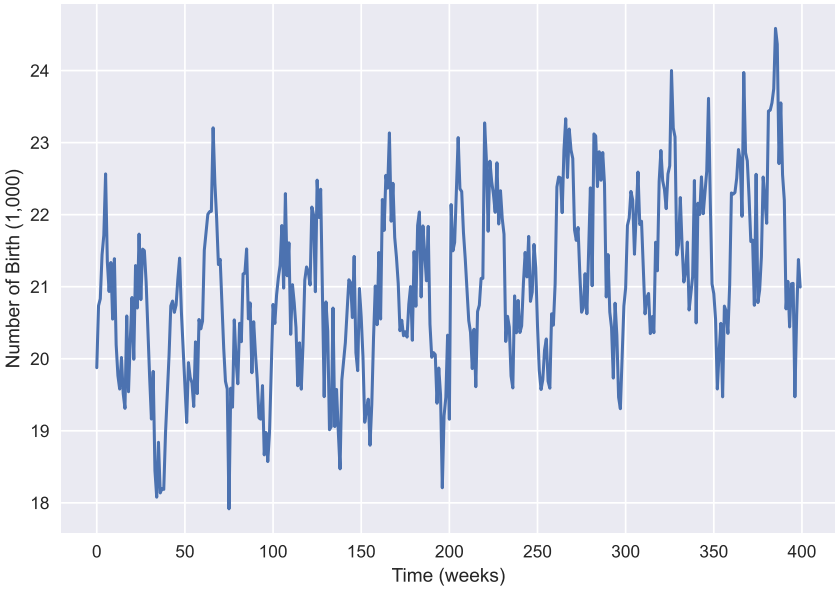
I use the proprietary high frequency (400 weekly observations) birth registration data for 24 selected provinces[1] and compare the predictability of traditional models and Bayesian Gaussian Processes. I split the sample into training (75% of the sample) and testing data (25% of the sample), train the model on the training data and then make predictions on both training and testing data. I compare the prediction accuracy of these two models with R^2. Based on the analysis, in my setting, Bayesian Gaussian Process Model yields a much better prediction for both the training dataset and the testing dataset. The difference in R^2 can be as large as 1.0 the traditional STL decomposition approach has negative R^2 if we define $R^2 = 1 - RSS/TSS$)

2 Data

We obtain the proprietary high frequency (weekly) birth registration data from China National Health and Family Committee. The following graph plots the raw data over 400 weeks with every data point to be the number of births[2] We can see several patterns from the graph

[1]This dataset is collected by China National Health and Family Committee and used as part of my own current research project. The raw dataset include each province as a separate table and I first aggregate them together before the analysis

[2]Technically it is not the actual number of new births but instead the number of new registers in a given week, as there might be a difference between the birth time and registration time, which is especially prevalent in rural areas.



- There is an overall trend of an increasing number of births over the sample period.
- There is clear seasonality with a cycle approximately equal to 50 weeks. This cycle roughly corresponds to the calendar year as the peak of birth in March and October is consistently documented in the literature [Bobak and Gjonca, 2001, Yang, 2021]

For the following analysis, we split out data (400 observations in total) into training sample (the first 300 observations) and testing sample (the last 100 observations). For both STL and Bayesian models, we train the model with the training sample and then evaluate the model with both training sample and testing sample.

3 Model and Results

We would like to compare the performance of the traditional time series model and bayesian model.

3.1 STL Model

Among numerous traditional time series forecasting model STL (Seaonsla and Trend decomposition using Loess[3]) [Cleveland et al., 1990] has been shown to be a robust versatile method with good performance in practice.

$$Y_t = T_t + S_t + R_t$$

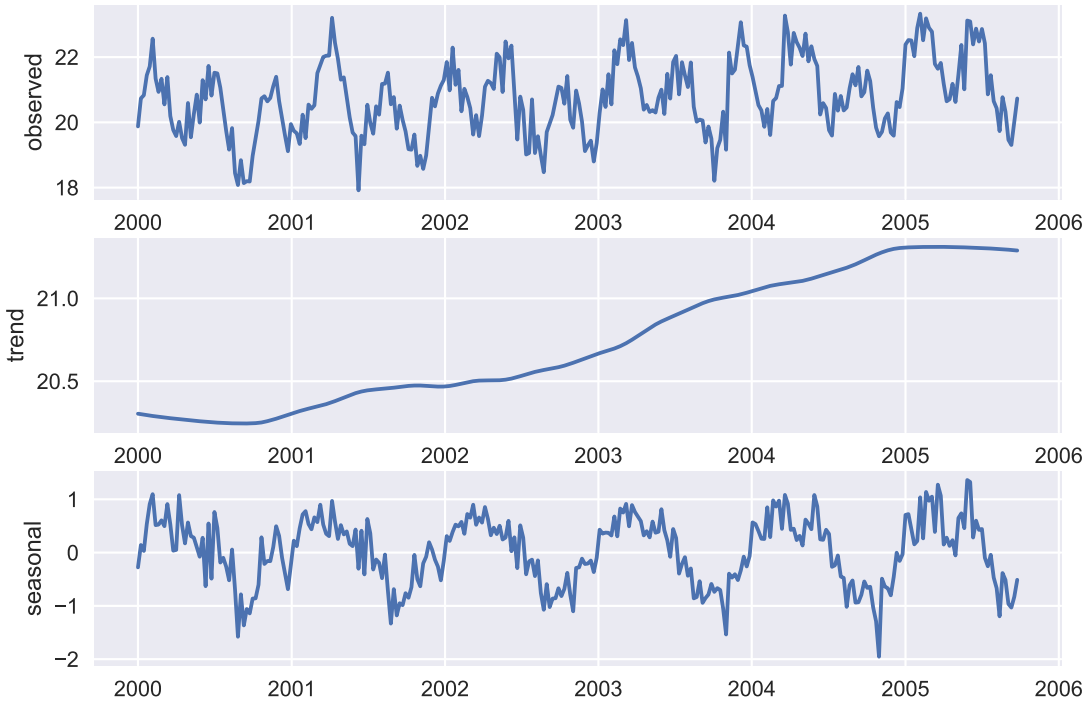
[3]Loess is a method for estimating nonlinear relationships by doing local weighted regression

STL model basically assumes that the data Y_t can be decomposed into three components, trend T_t, seasonal S_t and remainder R_t and the three components can be computed from the following steps [4]

- Find the approximate trend line fitting the time series data using, which can be obtained using OLS regression
- Find the de-trended series by subtracting time series data with approximate trend line
- Find the seasonal component by grouping the de-trended series by the cycle
- Populate this component across the whole data
- Find the de-seasonalized time series by subtracting time series data with seasonal data
- Find the trend by fitting the de-seasonalized data to a polynomial model

By calling STL class in *statsmodels* package, we get the following decomposition results[5]:

Figure 1: STL Decomposition



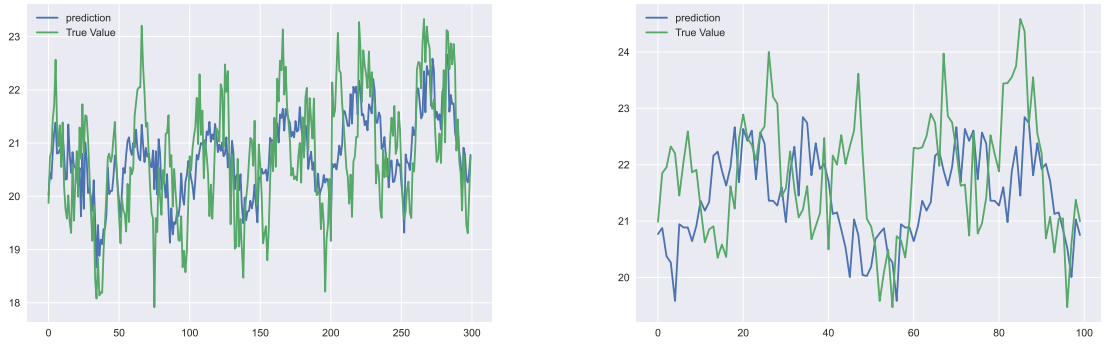
[4]Here we only provide a naive STL implementation as the full model involves a lot of smoothing using local regression, moving average, adaptively adjusting weight. See HERE for more details. STL is already implemented in Python's statsmodels library

[5]Note: the x-axis is only for calculation purpose as the STL model requires the index of series to be in Timestamp format. It is not the actual time of the data, which is omitted due to data usage restriction

As we can see clearly from the above results: the overall trend is increasing and there is a strong seasonality with a cycle of approximately one year. [6]

We add the trend and seasonality component to get predictions from STL model, as presented below

Figure 2: STL Model Prediction



(a) Training Sample

(b) Testing Sample

3.2 Bayesian Gaussian Process

Guassian process is a stochastic process [7] such that every finite collection of those random variables at different points of time has a multivariate normal distribution. The Gaussian Process problem is that:

We are given a time series $f(t)$ along the time dimension t, we want to make predictions $f(t')$ given some t'. The main estimation involves Bayesian updating (see Williams and Rasmussen [2006] for more details, PyMC has already implemented this): suppose the prior distribution of $f(t)$ and $f(t')$ are given by

$$f(t) \sim N(\mu_t, K_{t,t})$$
$$f(t') \sim N(\mu_{t'}, K_{t',t'})$$

Where μ_t, $K_{t,t}$ are mean and variance functions evaluated at t and $\mu_{t'}$, $K_{t',t'}$ are the same functions evaluated at t' . The join distribution of $(f(t), f(t'))$ is given by

$$\begin{bmatrix} f(t) \\ f(t') \end{bmatrix} \sim \mathbf{N}(\begin{bmatrix} \mu_t \\ \mu_{t'} \end{bmatrix}, \begin{bmatrix} K_{tt} & K_{tt'} \\ K_{t't} & K_{t't'} \end{bmatrix})$$

[6] Based on the residual plot, there might be another seasonality component with a smaller cycle. However STL does not allow users to directly specify multiple seasonality components. We can probably work on this in the future to allow for more flexible multi-cycle structure

[7] A stochastic process is a (might be uncountable) collection of random variables along the (time) dimension

Then based on Bayesian formula, the posterior distribution is given by

$$f(t')|f(t) \sim \mathbf{N}(K'_{t't}K_{tt}^{-1}(f(t)-\mu_t)+\mu_{t'}, K_{t't'}-K'_{tt'}K_{tt}^{-1}K_{tt'})$$

For Bayesian Gaussian process model, we specify the following model:

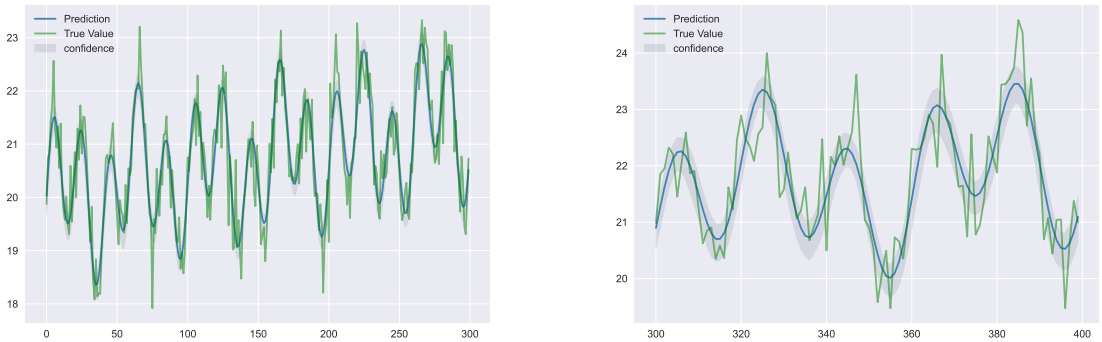
- Component 1 to capture the seasonality with cycle 52: periodic covariance function with lengthscale sampled from a *Gamma(2,2)* prior distribution and period sampled from a *Gamma(52,1)* prior distribution[8]
- Component 2 to capture the seasonality with the shorter cycle: periodic covariance function with lengthscale sampled from a *Gamma(2,2)* prior distribution and period sampled from a *Gamma(15,1)* prior distribution
- Component 3 to capture the trend: linear covariance function with the constant term sampled from *Normal(0.1,1)* distribution

The results after updating is listed below:

	mean	sd	hdi_2.5%	hdi_97.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
linear	0.047	0.993	-1.906	1.977	0.01	0.01	9900.0	6447.0	1.0
sns1	1.438	0.635	0.706	2.508	0.294	0.222	6.0	104.0	1.73
cycle1	78.138	25.967	51.67	104.971	12.956	9.923	6.0	112.0	1.73
sns2	1.662	0.232	1.198	2.095	0.018	0.012	170.0	5884.0	1.02
cycle2	20.009	0.026	19.956	20.059	0.0	0.0	8785.0	6239.0	1.0
sigma	0.492	0.021	0.451	0.531	0.0	0.0	9875.0	5924.0	1.0

To make predictions, we sample 500 processes from the trace: the point prediction is calculated by taking the average of the 500 stochastic processes, the credible region of the process is obtained by taking the credible region at each individual time point. Predictions for the training sample and the testing sample are displayed below:

Figure 3: Gaussian Process Prediction



(a) Training Sample

(b) Testing Sample

[8]Relatively informative prior is used based on our knowledge/literature

It is quite obvious that the Bayesian Gaussian Process approach generates better predictions based on a glance of the chart. To quantify the prediction accuracy, we calculate the R^2[9] in the following table

	Training Sample	Testing Sample
STL Model	0.418	-0.318
Bayesian Gaussian	0.821	0.733

As we can see from the table, the Bayesian Gaussian process model significantly outperforms the STL Model in both training sample and testing sample and the difference is even larger for testing sample (out of sample prediction)

4 Conclusion

From the analysis above, we can see that the Bayesian model significantly outperforms the traditional STL time series forecasting model with for both in and out of sample predictions and the difference is even larger for out of sample predictions.

Of course, my conclusion only holds for this particular analysis and should be interpreted with caution. For future work, as we can see from the prediction residuals of STL model, there is still some kind of seasonality with a smaller cycle, we may choose to add another layer to further decompose the residual to improve the performance of this model, making the comparison more convincing. In addition, we can probably try to tune the parameters in both the STL and Bayesian Gaussian Process model to better fit data and generate better predictions.

[9]For Bayesian Gaussian Process, we use the average of all sampling processes as our prediction

References

Martin Bobak and Arjan Gjonca. The seasonality of live birth is strongly influenced by socio-demographic factors. *Human reproduction*, 16(7):1512–1517, 2001.

Yuxiang Yang. Analysing the seasonality of births in mainland china. *Journal of Biosocial Science*, 53(2):233–246, 2021.

Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition. *J. Off. Stat*, 6(1):3–73, 1990.

Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pymc3 as pm
import arviz as az
from statsmodels.tsa.seasonal import STL
from statsmodels.tsa.api import STLForecast
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
plt.style.use('seaborn')
np.random.seed(42)

if __name__=='__main__':
    #read_data
    data=pd.read_csv('birth.csv')
    data=data[['t','num']]
    #plot graph for the data
    plt.plot(data['t'],data['num'])
    plt.xlabel('Time (weeks)')
    plt.ylabel('Number of Birth (1,000)')
    plt.savefig('graph1.pdf')
    plt.show()
    #split training and testing sample
    num_train=300
    data_train,data_test=data[:num_train],data[num_train:]
    #use frequelist regression
    data_train['time']='2000-01-01'
    data_train['time']=pd.to_datetime(data_train['time'])
    data_train['time']=data_train['time']+pd.to_timedelta(data_train['t']*7,u
    data_train_freq=data_train.set_index('time')
    data_train_freq=data_train_freq[['num']]
    # decomp = seasonal_decompose(data_train_freq["num"])
    # decomp.plot()
    # plt.savefig('decompose.pdf')
    # plt.show()
    stl = STL(data_train_freq, seasonal=13)
    res=stl.fit()
    fig,ax=plt.subplots(3)
    ax[0].plot(res.observed)
    ax[0].set_ylabel('observed')
    ax[1].plot(res.trend)
    ax[1].set_ylabel('trend')
    ax[2].plot(res.seasonal)
```

```
ax[2].set_ylabel('seasonal')
# fig=res.plot()
plt.savefig('decompose.pdf')
plt.show()
stlf = STLForecast(data_train_freq, ARIMA, model_kwargs=dict(order=(1, 1,
stlf_res = stlf.fit()
forecast = stlf_res.forecast(100)
forecast=pd.DataFrame(forecast)
forecast.reset_index(inplace=True)
forecast.rename(columns={0:'num'},inplace=True)
forecast=forecast[['num']]
data_test_freq=data_test.reset_index()
data_test_freq=data_test_freq[['num']]
#training r2
train_pred=res.seasonal+res.trend
r2_train=1-np.sum((train_pred-data_train_freq['num'])**2)/np.sum((data_tr
#testing r2
r2_test=1-np.sum((forecast['num']-data_test_freq['num'])**2)/np.sum((data
train_pred=pd.DataFrame(train_pred)
train_pred.reset_index(inplace=True)
train_pred.rename(columns={0:'num'},inplace=True)
train_pred=train_pred[['num']]
#plot: training forecast
plt.plot(train_pred,label='prediction')
plt.plot(data_train['num'],label='True Value')
plt.legend()
plt.savefig('prediction_training.pdf')
plt.show()
#plot: testing forecast
plt.plot(forecast['num'],label='prediction')
plt.plot(data_test_freq['num'],label='True Value')
plt.legend()
plt.savefig('prediction_testing.pdf')
plt.show()
print('training r2 of frequenlist model is given by '+str(r2_train))
print('testing r2 of frequenlist model is given by '+str(r2_test))
```

```
#bayesian model
with pm.Model() as m:
```

```
    #gaussian process: two seasonality term with one linear trend
    sns1=pm.Gamma(name='sns1',alpha=2,beta=2)
    cycle1=pm.Gamma(name='cycle1',alpha=52,beta=1)
    gauss1=pm.gp.Marginal(cov_func=pm.gp.cov.Periodic(input_dim=1,ls=sns1
    sns2=pm.Gamma(name='sns2',alpha=1,beta=1)
    cycle2=pm.Gamma(name='cycle2',alpha=15,beta=1)
    gauss2=pm.gp.Marginal(cov_func=pm.gp.cov.Periodic(input_dim=1,ls=sns2
    linear=pm.Normal(name='linear',mu=0.1,sigma=1)
    gauss3=pm.gp.Marginal(cov_func=pm.gp.cov.Linear(input_dim=1,c=linear)
    gauss=gauss1+gauss2+gauss3
    #observation
    sigma=pm.Gamma(name='sigma',alpha=0.1,beta=0.1)
    y_prediction = gauss.marginal_likelihood('y_prediction', X=data_train
    trace = pm.sample(draws=2000, chains=4, tune=500)
with m:
    #print results
    print(az.summary(trace,hdi_prob=0.95))
    az.summary(trace,hdi_prob=0.95).to_csv('bayesian_summary.csv')
    #plot results
    az.plot_trace(trace)
    plt.savefig('bayesian_trace.pdf')
    plt.show()
    #generate prediction
    x_train_values = gauss.conditional('x_train_values', data_train['t'].
    y_train_pred = pm.sample_posterior_predictive(trace, var_names=['x_tr
    x_test_values = gauss.conditional('x_test_values', data_test['t'].val
    y_test_pred = pm.sample_posterior_predictive(trace, var_names=['x_tes
#plot predictions
y_train_pp,y_test_pp=y_train_pred['x_train_values'],y_test_pred['x_test_v
y_train_mean,y_train_025,y_train_975,y_test_mean,y_test_025,y_test_975=y_
#plot: training sample
fig, ax = plt.subplots()
ax.plot(data_train['t'], y_train_mean, color='steelblue',label='Predictio
ax.fill_between(data_train['t'], y_train_025, y_train_975, color='lightsl
ax.plot(data_train['t'],data_train['num'],color='green',alpha=0.5,label='
ax.legend()
plt.savefig('bayesian_prediction_training.pdf')
plt.show()
#plot: testing sample
fig, ax = plt.subplots()
ax.plot(data_test['t'], y_test_mean, color='steelblue',label='Prediction'
ax.fill_between(data_test['t'], y_test_025, y_test_975, color='lightslate
ax.plot(data_test['t'],data_test['num'],color='green',alpha=0.5,label='Tr
ax.legend()
plt.savefig('bayesian_prediction_testing.pdf')
```

```
plt.show()
#calculate training r2
br2_training=1-np.sum((y_train_mean-data_train['num'])**2)/np.sum((data_t
#calculate testing r2
br2_testing=1-np.sum((y_test_mean-data_test['num'])**2)/np.sum((data_test
print('training r2 of bayesian model is given by '+str(br2_training))
print('testing r2 of bayesian model is given by '+str(br2_testing))


print('finished!')
```