# Summer 2019/ June 18, 2019/ FNAL

Rachael Keller. Memo: Init.

## Abstract

In which we discuss autoregressive and gaussian process modeling of time series data.

## 1 Introduction

Consider a space of variables $\mathcal{P} \subset \mathbb{R}^{|\mathcal{P}|}$ that are settings, $\mathcal{Q} \subset \mathbb{R}^{|\mathcal{Q}|}$ that are observations, and $\mathcal{E} = \{0,1\}^{256}$ that are indicators of event $\mathcal{E}_i, i \in \{1,2,\dots,256\}$ occuring. We are working on a problem with some input parameters $\vec{p} \in \mathbb{R}^{m_p}$, $\vec{q} \in \mathbb{R}^{m_q}$, and $\vec{E} \in \{0,1\}^{256}$. Parameters $\vec{p} \in \mathbb{R}^{m}$ are real-valued, independent variables specifying settings for a machine; $\vec{q} \in \mathbb{R}^{m_q}$ are observable variables; and $\vec{E}$ is a binary vector mapping to events, where a value of 1 means an event is on and 0 off[1].We hope to learn a function $f$:

$$f : \mathbb{R}^m \times \{0,1\}^{256} \to \mathbb{R},$$

where $m \le m_p + m_q$, to predict some observable $q_i$, $i \in \{1,2,\dots,m_q\}$ at future time steps. In particular, we let $q_0$ be the primary quantity of interest ($q_0 = \Delta V$).

For now, we have a large dataset over time of parameters $\vec{p}$ and $\vec{q}$. These samples are in discrete time, and for the remainder of this paper we let $t$ index the (discrete) time, so that $\vec{p}_t, \vec{q}_t, \vec{E}_t$ refer to the parameters at time $0 \le t < T < \infty$.

Ultimately, we hope to determine the input space $\mathcal{X} \subset \mathcal{P} \cup \mathcal{Q} \cup \mathcal{E}$, so that we may refer to our samples as $\mathbf{x} \in \mathcal{X}$ and approximate some function $f$ where $f(\mathbf{x}) = y \in \mathbb{R}$. We have yet to specify precisely which variables should be chosen. We are in the process of obtaining the data $\vec{E}_t$, and we are still in the process of selecting parameters from $\mathcal{P}$ and $\mathcal{Q}$.

Note that parameters from $\mathcal{Q}$ are observations, so they may not be given in real-time to predict a parameter in it. Therefore, one would first specify that no $\vec{q}$ may enter the learning problem as an input. However, the parameters $\vec{p}$ are specified settings and are rarely changed, so that the data one generates over time of these inputs is only a rank 1 matrix. One can represent this matrix by a single row vector; it is therefore singular, and so we have problems with learning techniques that require full rank such as kernel methods.

To cut out kernel methods would eliminate a wide variety of robust methods for learning, so that is not a route we hope to take. Instead, we might predict other parameters in $\mathcal{Q}$ and use them to inform a model to predict the primary quantity of interest.

To understand if time series analysis is even required, or if the problem is merely static, we use `pandas`' `lag_plot` function to see the time dependence on a few of the variables of interest: two observation variables `B:IMINER` and `B:VIMIN`, and a setting `B_VIMIN`. We show the results in Figure 1. If the data were truly random, there would be no pattern showing dependence of the two. One can see clearly that the variables `B:IMINER` and `B:VIMIN` display linear time dependence; interestingly, the observation variables `B:VIMIN` seems to be reset and evolves linearly for some clusters of data. Therefore, the plots are compelling for studying different types of time series models.

[1] Let $i \in \{0, 1, m_p - 1\}$ and $j \in \{0, 1, \dots, m_q - 1\}$. We abuse notation and let $p_i \in \vec{p}$ or $q_j \in \vec{q}$ refer to both the real-valued variable it represents and what it represents. i.e. $q_0$ is the value $q_0$ takes and what it represents $\Delta V$.

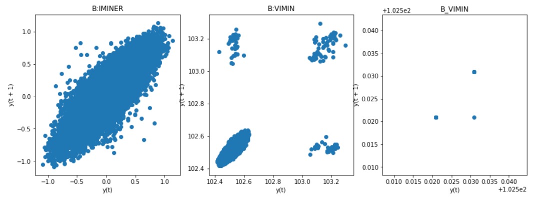


Figure 1: Lag plot of quantities of interest. The x-axis is the data sampled at time $t$ and the y-axis is the observed state at time $t+1$.

## 2 AR Models

### 2.1 Introduction

Autoregressive (AR) models are a form of linear regression models where the time series is regressed on previous time steps. Suppose we have a discrete time series $\{\mathbf{x}_t\}_{t=0}^{T}$, and we hope to predict $\mathbf{x}_T$. Consider the model

$$\mathbf{x}_T = \alpha_0 + \alpha_1 \mathbf{x}_{T-1}.$$

This model is an AR(1), model, where the argument 1, the *order* of the model, indicates the most distant previous time step used in the model. Generally speaking, for $0 < T' < T$

$$\mathbf{x}_T = \alpha_0 + \sum_{k=1}^{T'} \alpha_j \mathbf{x}_{T-k}$$

is an AR(T') model of order T'.

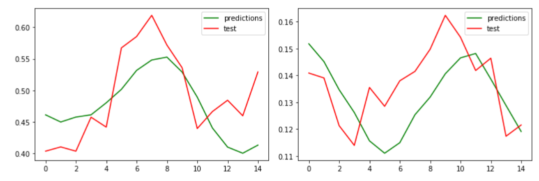
There are several advantages to using AR models. The coefficient of correlation between data of two different time steps, known as the autocorrelation function, gives a quick measure of the linear relationship between observations at different times. The difference in the time steps is known as the *lag*. One can use the lag in the autocorrelation function to deduce the optimal order of an AR model.

In addition to the nice, though simple, properties of the model, there are many computer programming (coding) packages available to implement an AR model and optimize an AR model for the dataset. The python packages `pandas` and `scipy` may be used to build models and to predict a future time series over a specified time window. In the following section, we show some results over a one-hour time period of data.

### 2.2 Results

In this section, we show the results of some experiments with a 60-minute window of samples. We fit an AR model

Figure 2: AR Models for `B:IMINER` and `B:VIMIN`



(a) `B:IMINER` 1 second Prediction MSE: 3.2e-3 (b) `B:VIMIN` 1 second Prediction MSE 2.0e-4

using `scipy`'s AR package. Qualitative results are shown in Figure 2.

Our first goal is to predict a cycle – one second, or 15 time steps – in advance. We have a total of 48108 time steps of data to train it and, as a test, save the final 15 pieces of data to serve as a test. The prediction for the final 15 time steps is plotted alongside the observed values. For both models, the optimal lag computed and used was 56. The training for `B:VIMIN` was an order of magnitude closer to the true result compared with `B:IMINER`, which is a derived quantity and dependent on `B:VIMIN`.

An important aspect of this experimentation is to note the smoothness of the predictions versus the true observation. The autoregressive model fails to capture the noise and instead plots a smoothed curve. Since the data we deal with is highly noisy, these models might not be the best choice for modeling.

## 3 Gaussian Processes

### 3.1 Introduction

Gaussian Processes are a Bayesian kernel regression method that combines the nice properties of Gaussian functions with the richness of kernel methods. In Bayesan linear regression, one assumes a prior distribution over parameers and outputs a probability distribution over possible outputs. Gaussian processes generalize the output to a random functions– an infinite dimensional space.

Let $\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$ be a collection of random variables. A collection of such variables is said to be drawn from a Gaussian process with mean $m : \mathcal{X} \to \mathbb{R}$ and covariance $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ if for any finite input set $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathcal{X}$, the associated set $f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_m)$ satisfies:

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_m) \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} m(\mathbf{x}_1) \\ \vdots \\ m(\mathbf{x}_m) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \boldsymbol{K}(\mathbf{x}_1, \mathbf{x}_m) \\ \vdots & \ddots & \vdots \\ \boldsymbol{K}(\mathbf{x}_m, \mathbf{x}_1) & \cdots & \boldsymbol{K}(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix} \right)$$

The covariance function may be any function that satisfies Mercer's conditions. In the discrete case, if we let $K$ refer to the matrix of $k(\cdot, \cdot)$ values specified above, any kernel function statisfying Mercer's condition is positive semidefinite.

A well-known, popular kernel is the gaussian kernel:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}'; \sigma) = \exp\left( -\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} \right),$$

where $\sigma \in \mathbb{R}$. We will use this for now in our preliminary experiments.

Python has several packages for Gaussian Processes. We use a nice package `GPFlow`, with a backend in Tensorflow, that we use for our experiments.

### 3.2 Results

We use the same data set as before, except now we only hope to train the model to predict values of `B:IMINER`. We use the gaussian kernel, as described above, and use `gpflow` to build a model [2]. A qualitative view of the results on this preliminary experiment are shown in Figure 3.

### 3.3 Discussion for Time Series

With Bayesian inference, we're building a probability distribution over functions

$$p(\boldsymbol{f}(\mathbf{x})) = \mathcal{N}(\boldsymbol{m}(\mathbf{x}), \boldsymbol{K}(\mathbf{x}, \mathbf{x})),$$

where $\boldsymbol{f}(\mathbf{x}) = \{f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_m)\}$ are dependent function values evaluated at $\mathbf{x}_1, \mathbf{x}_2, ldots, \mathbf{x}_m$. $m(\mathbf{x})$ is a mea nfunction and $\boldsymbol{K}(\mathbf{x}, \mathbf{x})$ is our covariance matrix. If we assume that the data has some random, uncorrelated noise from sample to sample, then the variance:

$$\mathbb{V}(\mathbf{x}) = \boldsymbol{K}(\mathbf{x}, \mathbf{x}) + \sigma^2 \boldsymbol{I},$$

where $\boldsymbol{I}$ is the identity matrix and $\sigma^2$ is a hyperparameter of the variance of the added noise.

Now, if we seek to use a GP over some test sample $(\mathbf{x}_*, y_*)$, then we begin with the training data and then combine with the test set:

$$p\left( \begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \right) = \mathcal{N}\left( \begin{bmatrix} \boldsymbol{m}(\mathbf{x}) \\ m(x_*) \end{bmatrix} \begin{bmatrix} \boldsymbol{K}(\mathbf{x}, \mathbf{x}) & \boldsymbol{K}(\mathbf{x}, x_*) \\ \boldsymbol{K}(x_*, \mathbf{x}) & k(x_*, x_*) \end{bmatrix} \right)$$

Then, we may deduce the posterior distribution over $y_*$ is Gaussian with mean and covariance given by

$$\begin{aligned} m_* &= m(x_*) + K(x_*, \mathbf{x}) \boldsymbol{K}(\mathbf{x}, \mathbf{x})^{-1} (\boldsymbol{f}(\mathbf{x}) - m(\mathbf{x})) \\ \sigma_*^2 &= K(x_*, x_*) - \boldsymbol{K}(x_*, \mathbf{x}) \boldsymbol{K}(\mathbf{x}, \mathbf{x})^{-1} K(x_*, \mathbf{x})^T, \end{aligned}$$

where the superscript $T$ denotes the transpose.

We then extend this for a set of input outside our observations $\mathbf{x}_*$ to obtain a posterior distribution over $\boldsymbol{f}_* \equiv \boldsymbol{f}(\mathbf{x}_*)$ described by:

$$p(\boldsymbol{f}_*) = \mathcal{N}(\boldsymbol{m}_*, \boldsymbol{C}_*),$$

where

$$\begin{aligned} \boldsymbol{m}_* &= \boldsymbol{m}(\mathbf{x}_*) + \boldsymbol{K}(\mathbf{x}_*, \mathbf{x}) \boldsymbol{K}(\mathbf{x}, \mathbf{x})^{-1} (\boldsymbol{f}(\mathbf{x}) - m(\mathbf{x})) \\ \boldsymbol{C}_* &= \boldsymbol{K}(\mathbf{x}_*, \mathbf{x}_*) - \boldsymbol{K}(\mathbf{x}_*, \mathbf{x}) \boldsymbol{K}(\mathbf{x}, \mathbf{x})^{-1} \boldsymbol{K}(\mathbf{x}, x_*)^T. \end{aligned}$$

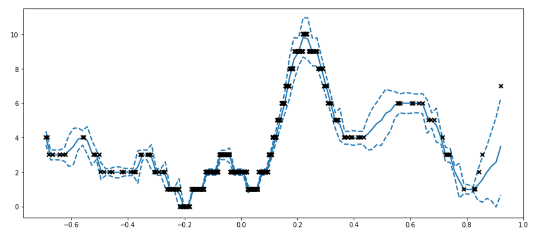


Figure 3: Gaussian Process Model of Predicted Values (black x) within a window of two standard deviations from the expected value.

## References

[1] Do, Chuong and Lee, Honglak. *Gaussian processes*. Course Lecture Notes. Stanford University, Spring 2019. `http://cs229.stanford.edu/section/cs229-gaussian_processes.pdf`.

[2] GPFlow. `https://gpflow.readthedocs.io/en/latest/notebooks/ordinal.html`

[3] Roberts, Stephen, et al. "Gaussian processes for time-series modelling." Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 371.1984 (2013): 20110550.