Time Series Project

Jacob Clinton George Smith-Kolff

2024-10-14

Contents

Chapter 1

Introduction

(Probably work on this section last)

1.1 What is missing data

1.2 Issues caused by missing data

1.3 Structure of the report

Chapter 2

Theory

2.1 Time series definition

Missing at random. This basically means that the missingness in the data is not imformative

We begin by providing definitions for both uni variate and multivariate time series:

A univariate time series $X = \{x_1, x_2, ..., x_t\} \in \mathbb{R}^t$ is a sequence t observations on a single variable.
This can be extended to multivariate time series:
$X = \{x_1, x_2, ..., x_t\} \in \mathbb{R}^{t \times d}$ where each x_i is a d dimensional vector.

2.2 Missing data Mechanisms

Data missingness is a common occurrence that is common in working with real world data. Missing data can arise from device failures such as measureing equipment failing, or from data censoring (such as from governments) [1]. Missing data typically falls into one of three categories:

2.2.1 Missing Completely at Random (MCAR)

Data is said to be missing completely at random if the distribution that describes how missingness occurs is independent of both the observed and unobserved values in the time series.

2.2.2 Missing at Random (MAR)

Missing at random is when missingness is related to the observed data, but is independent of the unobserved data. This means that there is some external factor that is causing the missingness. An example given in [2] is that data from a sensor is more likely to be missing on weekends since on some weekends the sensor is shutdown.

2.2.3 Not Missing at Random (NMAR)

Not missing at random means that the missingness is related to the value of the observation itself. An example of this is a sensor that will return a missing value if the recorded value is above 100°.

Come back and include probability notation

2.3 Missing data imputation

Most missing data imputation methods require MCAR or MAR, this is because these systems of missingness are not informative towards the missing values themselves.

Can extend this writeup later.

Idea is to start with the most basic techniques, then more the more advanced ones

2.3.1 Univariate data imputation

2.3.1.1 The struggle with univariate data imputation

Unlike with a multivariate time series, a univariate time series is only one sequence of observations $\{y_1, y_2, ..., y_n\}$. This simplier form actually leads to increased difficulty when it comes to imputing a missing value of y, since we can no longer exploit any dependencies between the predictor variables in the series. With univariate imputation we can only rely on the previous (or future) values for y along with the implicit time variable [1].

2.3.1.2 Last Observed Carried Forward (LOCF) Next Observation Carried Backward (NOCB)

[2]

2.3.1.3 Mean/Median imputation

This is another simple approach for filling in a missing value. We take the mean over the entire series and impute the missing value with the series mean:

$$\hat{\mu} = \sum_i \frac{y_i}{n} \tag{2.1}$$

If we replace Equation 2.1 with the median, then we have median imputation.

[1] moritz2015comparison
[2] ahn2022comparison

Keep talking about why this method is not great in general and why it could be worse for series with a trend

2.3.1.4 Spline interpolation

A split interpolation of missing data assumes a linear relationship between the data points. The equation is given by:

$$f(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_1 - x_0) \tag{2.2}$$

2.3.1.5 Kalman filter

2.3.2 Multivariate data imputation techniques

2.3.2.1 K-nearest neighbours

K-Nearest neighbours is a simple non-parametric machine learning algorithm, it can be used for either classification or regression. It can be applied in a time series data imputation context to estimate missing values in a time series.

$$\frac{1}{K}\sum_{j=1}^{K} Y_j$$

Consider a missing value x_i in a time series, k-nearest neighbours will impute a value for the missing value by calculating the mean of the data points in the neighbourhood determined by a distance metric such as Euclidean distance. The algorithm is simple and is shown to be quite effective [3]. K-nearest neighbours algorithm has a hyper-parameter K, the number of nearest (non missing) data points to consider. A choice of K that is too large may lead to smoothing over temporal patters (under fitting) this has the effect of ignoring potentially important seasonal patterns. On the other hand a value of K that is too small could lead to being overly sensitive to the noise. The value for K needs to be carefully chosen, typically by cross validation, but in a time series context it could be chosen using sliding windows and forward chaining. The curse of dimensionality is an issue for K-nearest neighbours. With higher dimension it becomes harder to find data points that are closer, this can lead to an increased sensitivity to noise and misleading imputations. The computational cost also increases rapidly with more dimensions in the data making the algorithm less efficient. Another issue related to computational cost, K-nearest neighbours is a lazy learning algorithm so the complete dataset ends up being stored. For large datasets this can make the algorithm slow, or even infeasible

Come back to this section later

2.3.2.2 Multivariate Imputation by Chained Equations (MICE)

MICE is a common imputation technique that is quick to incorporate and yields effective results. The algorithm begins by using something simple like mean imputation, to fill in initial missing values.

For each variable where missingness is present, we build a regression model. From there, lagged values $y_{t-1}, y_{t-2}..., y_{t-n}$ as predictors. This is to say that if $y(t)$ is missing. predict using y_{t-1} and other variables.

impute the missing values for one variable at a time. After that variable is imputed, use the updated data and move on to the next variable. We repeat this process until we reach convergence for our given variable.

This chained process is typically repeated 5-10 times to generate multiple datasets that contain a variation of results.

After, each data set that has undergone imputation is analysed. Then, we combine the results using Rubin's rule.

Rubin's rule is as follows:

1. **Pooled Estimate** (e.g., mean, regression coefficient):

$$\hat{\theta} = \frac{1}{m} \sum_{i=1}^{m} \theta_i$$

where $\hat{\theta}$ is the pooled estimate, m is the number of imputed datasets, and θ_i is the estimate in each imputed dataset i.

2.3.2.3 General Adversarial Networks (GANs)

GANs consist of two neural networks. A generator (G), and a discriminator (D) . These two networks are trained simultaneously through a game theory approach. It is the generators job to try and generate realistic data points, while the discriminator tries to distinguish whether the values are true values or values generated from the generator. The generator is tasked to try and fool the discriminator. In turn, attempting to learn the underlying distribution of the data. This class of machine learning models were first purposed in 2014 [4].

The typical process of using a GAN is as follows:

1) The generator produces an estimate for the missing values conditioned on **all** available observed data points.
2) The discriminator evaluates the quality of the imputed value compared to real observed values.
3) Both networks are trained to improve the generators ability to fool the discriminator.

The training process continues until the generator can emulate realistic data that will fool the discriminator and in turn, replicate the distribution of the observed data. Given this, GANs can provide a powerful architecture for imputing missing values. Particularly in cases where imputing techniques can struggle to model the complex relationships within the data [4].

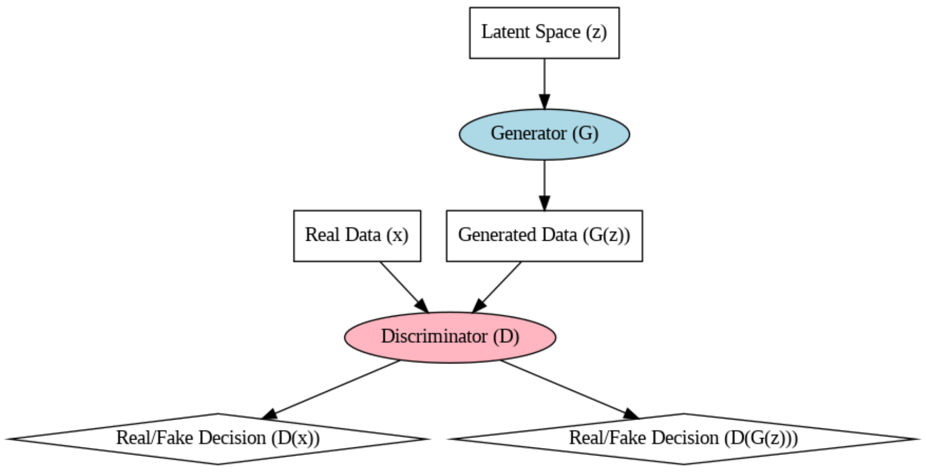
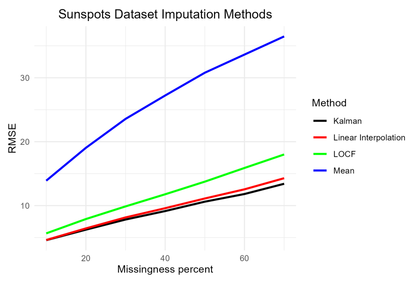


Figure 2.1: Structure of our GAN for TS imputation

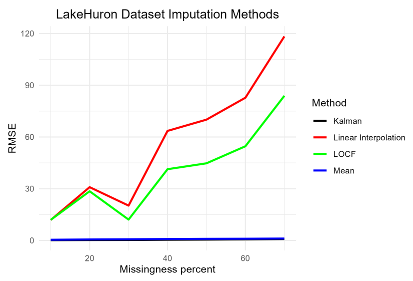
In our GAN, we use a standard Generator (G)and Discriminator (D) to handle missing value imputation. The generator receives a sampled vector from the latent space (Z) and uses that to estimate missing data points. The discriminator then evaluates whether the generated data follows the true observed data. Trained using a minimax game where the generator tries to keep fooling the discriminator.

Chapter 3

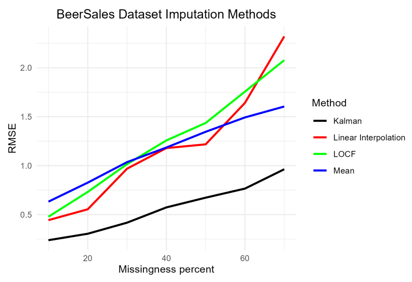
Illustration of time series imputation



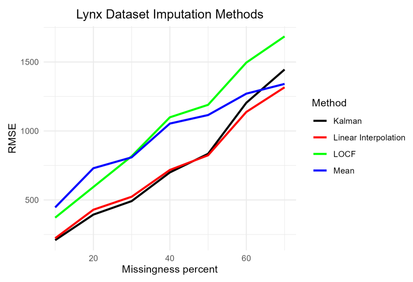
(a) Data imputation on Sunspots



(b) Data imputation on LakeHeuron



(c) Data imputation on BeerSales



(d) Data imputation on Lynx

Figure 3.1: Comparison of Locf, Mean, Linear, and Kalman data inputation on four datasets

Chapter 4

Discussion

Appendix

[1] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*, vol. 793. John Wiley & Sons, 2019.

[2] S. Moritz, A. Sardá, T. Bartz-Beielstein, M. Zaefferer, and J. Stork, "Comparison of different methods for univariate time series imputation in r," *arXiv preprint arXiv:1510.03924*, 2015.

[3] H. Ahn, K. Sun, K. P. Kim, *et al.*, "Comparison of missing data imputation methods in time series forecasting," *Computers, Materials & Continua*, vol. 70, no. 1, pp. 767–779, 2022.

[4] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf