Adapting Adversarial Estimator for Time Series Data

A Recurrent Neural Network Approach

José Luis Martínez Martínez

Introduction

- **Adversarial Estimator** is a Simulated Method[1] developed by Tetsuya Kaji, Elena Manresa and Guillaume Pouliot.

- **Why we should even try it?**

  Results from Tetsuya Kaji, Elena Manresa and Guillaume Pouliot (2021,2023):

  1. **Good Asymptotic Properties** $\rightarrow$ Consistency and Efficiency.

  2. **Good Finite Sample Performance** $\rightarrow$ Competitive Estimations against other popular simulated methods (in particular when using **Neural Networks** as Discriminator)

[1] Simulated methods depart from Analytical results and use computational power to derive Numerical results.

Motivation

- This New method is Amazing BUT still has something to be explore (Manresa et al. (2023)):

  "First, the theoretical results in this paper do **not cover time series data***.[...]. This is not to say that the adversarial framework cannot be extended thereto, but it would require a* **careful design of the discriminator** *to* **incorporate** *the structure of the* **serial correlation***."*

$\Downarrow$

Challenge:

[**Adapt** Adversarial Estimator (**Discriminator**) to **Time Series** Data]

Contribution

To Propose a Proper Adaptation of the Discriminator within the context of Time Series Data.

- **What do we know?**

  1. This Adversarial method works **better** when using **Neural Network**
  2. Time series data requires a **Discriminator** capable of **absorbing time dependencies**

Our proposed Adaptation will be driven by the above two facts.

Outline

Adversarial Estimator:

- GANs
- Neural Networks
- Adversarial Procedure

Adaptation to Time series:

- Why a Recurrent Approach should work?

Result

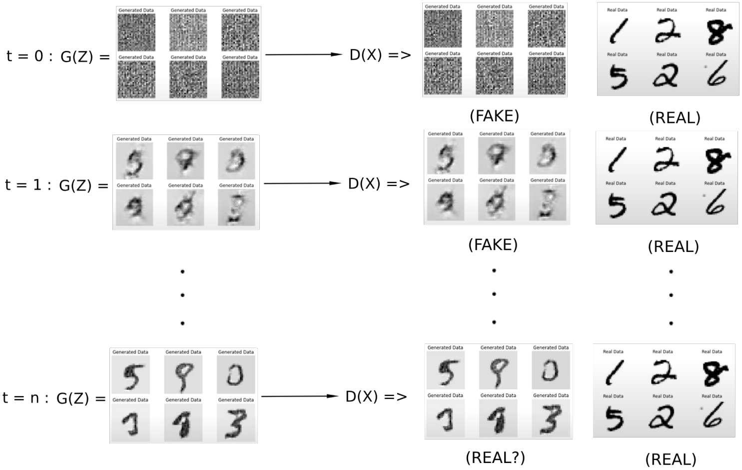
Generative Adversarial Networks (GANs)

GANs were first proposed by Goodfellow et al. 2014 as an image generator model. But the problem it addresses is broader than that:

- We have some **Data (Distribution)** that we want **to Mimic**
- So we create a contest where **two models** try to **Optimize** themselves **simultaneously**[2]:
  - Model 1 **(G)** : **Generator** $\rightarrow$ generates Fake Data
  - Model 2 **(D)** : **Discriminator** $\rightarrow$ labels Real Data vs. Fake Data
- **Results** $\rightarrow$ **(G)** gets really good on **reproducing Real Data**

[2] This end up being a min-max problem

Generative Adversarial Networks (GANs)



Neural Networks (NNs)

- **What are Neural Networks?:**

  NNs are Structures that **produce functions** relating some inputs to some outputs $\rightarrow$ mainly used for **Prediction** and **Classification** tasks

  They are formed by:

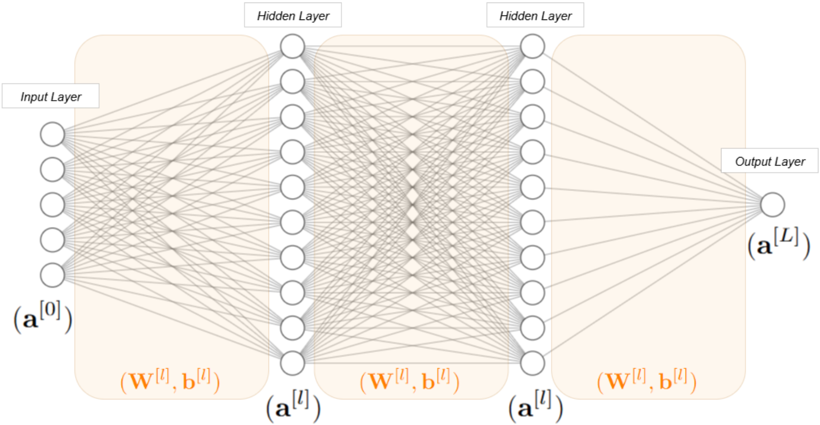
  - Layers $\equiv \mathbf{l}$
  - Neurons $\equiv \mathbf{a}$
  - Weights and Biases $\equiv (\mathbf{W}, \mathbf{b})$

  We can end up with something like this [3]:

$$D(\mathbf{x}) = \sigma\left(w_{11}^2 \cdot \sigma(w_{11}^1 \cdot \mathbf{x} + b_1^1) + w_{12}^2 \cdot \sigma(w_{21}^1 \cdot \mathbf{x} + b_2^1) + w_{13}^2 \cdot \sigma(w_{31}^1 \cdot \mathbf{x} + b_3^1) + b^2\right)$$

[3] Example for a NN with 1 single Input, 1 Hidden Layer with 3 Neurons and 1 single Output

Neural Networks (NNs)



$$\mathbf{a}^{[l]} = \sigma(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]})$$

Neural Networks (NNs)

- **How we compute those parameters?**

The set of parameters ($\theta = (\mathbf{W}, \mathbf{b})$) are computed **through** a Learning Process called **Training**.

This training consist of a **Loss Function Minimization** $\rightarrow$ **Gradient Decent**

We keep updating our parameters in the fastest direction, marked by the gradient:

$$\theta^* = \theta - \eta \left(\frac{\partial \mathcal{L}}{\partial \theta} \right)$$

Where:

$\mathcal{L} \equiv$ Loss function
$\eta \equiv$ Learning Rate (dictates the size of the descent step)

Adversarial Procedure (Parameter Estimation)

Adversarial Estimator introduce the figure of the **Discriminator** from the **GANs** to **select** the model's **parameters** that best **fit** the **real data points**.

For the context of **Parameter Estimation**, the **Generator (G)** is **represented by** our proposed **Model**.

Estimation gets the parameters that **best confound** fake data with real data
$\rightarrow D(G(\theta)) \approx 0.5$

The **Adversarial Procedure** to reach the estimated parameters can be **summarize as**:

1. Generator (G): generates fake data
2. Discriminator (D): is trained for classify real data (1) from fake data (0)
3. Update Parameters: we use trained (D) to update proposed parameters in (G)
4. Convergence: process repeats till convergence is reach (e.g Updating reach a flat state)

Outline

Adversarial Estimator:

- GANs
- Neural Networks
- Adversarial Procedure

Adaptation to Time series:

- Why a Recurrent Approach should work?

Result

Adaptation to Time series

In general the Discriminator can be any function $\mathcal{D} : D(x) \in (0, 1)$

But as Manresa et al. (2021,2023) shows, this elicitation is not trivial, efficiency and accurate estimation depend on it.

- Time series adds another layer of complexity:

  Discriminator must be able to **absorb intertemporal dependencies**.

  In order to **capture these correlation**s in data we propose to use a **Recurrent approach** of the Neural Network.

Why a Recurrent Approach should work?

NNs treat each data inputted as independent from the rest of the data set. On the other hand, **Recurrent Neural Networks (RNN)** where design to account for correlation in data $\rightarrow$ (**hidden state:** keeps track of previous output decisions)

$\Downarrow$

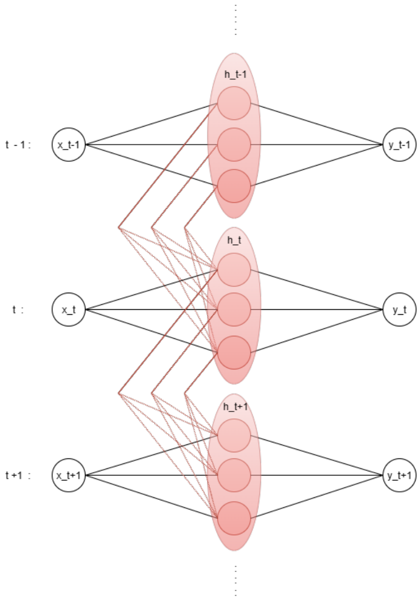
This sound like a **good candidate** for our purpose.

In particular we propose **Long Short Term Memory (LSTM) Networks**::

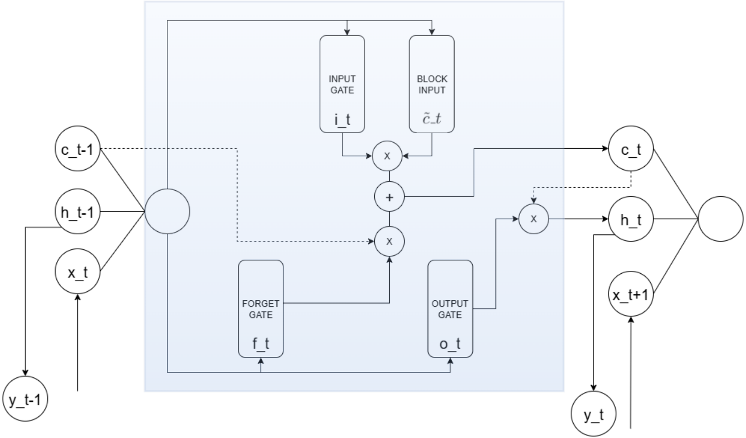
1. LSTM Networks self learn how much of previous learning use in present outputs $\rightarrow$ (memory management)
2. Is a commonly used RNN and also solve some problems of the classical RNN training.

(BUT recall, **Contribution** resides on applying this **Recurrent Structures as Discriminator**)

Recurrent Neural Network (RNN)



Long Short Term Memory (LSTM)



Outline

Adversarial Estimator:

- GANs
- Neural Networks
- Adversarial Procedure

Adaptation to Time series:

- Why a Recurrent Approach should work?

Result

Results(1)

To corroborate the effectiveness of our proposal LSTM Discriminator we start running some simulation incorporating it to the estimator.

BUT things did not went well:

- Estimations where too poor (faraway from true parameters)
- The convergency of the algorithm (GAN based) was not often reached and unstable

We have to concluded that our proposed estimator was a failure.

Results(2)

The **problem** was not on our proposed Discriminator (LSTM); the problem was in **how we were training it**:

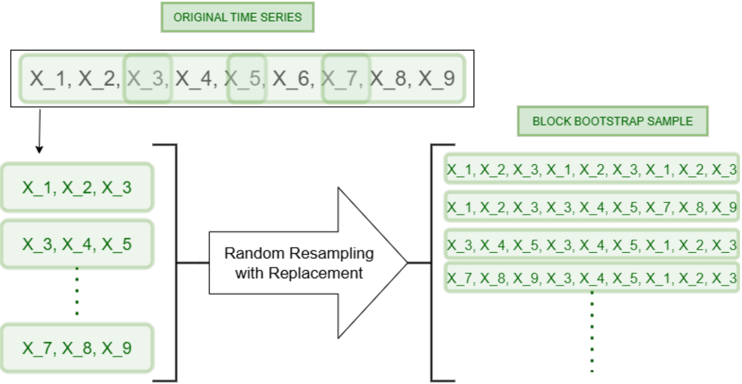
- NNs assume for their training process an iid training set.
- We are dealing with time series data $\rightarrow$ Long Series of Data (Correlated Training Data set by definition)
- Our training process consists simply on chopping subseries out of the Long series and labeling them as 'real data' $(= 0)$ $\rightarrow$ this correlation among the training set for the Discriminator was causing the lack of convergency and proper performance.

We need to find a way to **construct an iid training sample** BUT **keeping the correlation structure** of the Long **time series**.

Results(2). Improvement: Block Bootstrap

Solution: Train the Discriminator with a **Block Bootstrap** version of the **Real Data**.

Block Bootstrap $\rightarrow$ iid sample out of a correlated one while maintaining the underlying correlation.



Results(2)

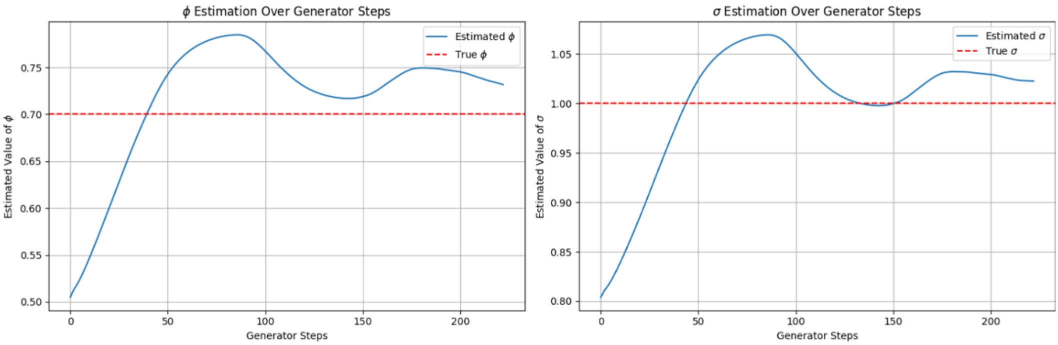


Figure: Estimation Updating for an *MA(1)*.

Estimated $\hat{\theta} = 0.7317$ (True $\theta = 0.7$)
Estimated $\hat{\sigma} = 1.024$ (True $\sigma = 1$)

Results(2)

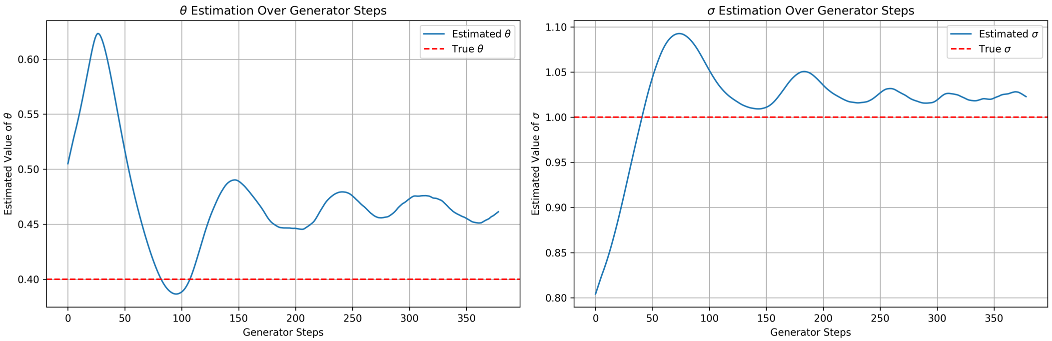


Figure: Estimation Updating for an *AR(1)*.

Estimated $\hat{\theta} = 0.4613$ (True $\theta = 0.4$)
Estimated $\hat{\sigma} = 1.0229$ (True $\sigma = 1$)

Results(2)

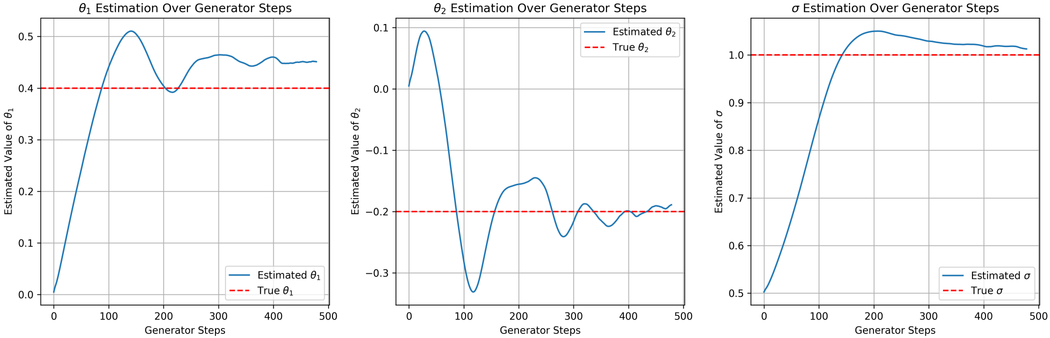


Figure: Estimation Updating for an *AR(2)*.

Estimated $\hat{\theta_1} = 0.4512$ (True $\theta_1 = 0.4$)
Estimated $\hat{\theta_2} = -0.1889$ (True $\theta_2 = -0.2$)
Estimated $\hat{\sigma} = 1.0127$ (True $\sigma = 1$)

Conclusion

- Findings:
  - LSTM based Discriminator captures the data correlation needed for a proper adaptation.
  - But it has to be followed by a transformation of the training data set (Real data) to fit the iid assumption (e.g using a Block Bootstraped version of the Real Data).
- Limitations (Above points have to be taken carefully):
  - The proposed estimator has not been properly tested (bigger and more robust simulation)
  - The proposed estimator has not been compared to other simulated methods.

Despite the short scope of our results, this seems to be at least a proper starting point for latter improvements and a vehicle to more complex models.

The End