BU.610.740: Forecasting Models for Business Intelligence
Chapter 4: Recent Advances

Ali Eshragh

Spring II, 2024

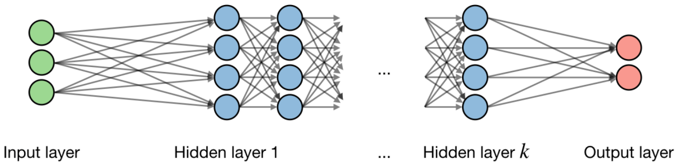
Topics Covered in These Slides

1. **Neural Forecasting**
   - Introduction
   - Recurrent Neural Network
2. **Efficient Algorithms for Big Time Series Data**
   - Autoregressive Model
   - Randomized Numerical Linear Algebra
   - Big Time Series Data and RandNLA
3. **Future Work**

Machine Learning for Prediction

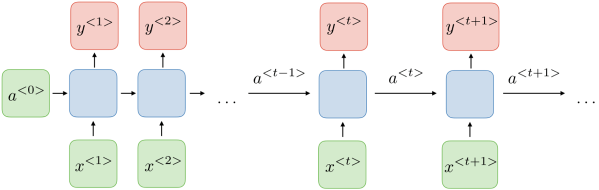
- **Machine learning** is a subset of artificial intelligence that involves **training** algorithms to make **predictions** or **decisions** without being explicitly programmed.
- **Neural networks** are a class of machine learning models inspired by the human **brain**, designed to **recognize** patterns and **solve** complex problems.
- Neural networks excel in **predicting** outcomes based on **large** and **complex** datasets, used widely in fields such as business, healthcare, and engineering.

Neural Networks: A General Overview



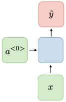
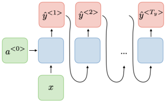
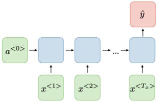
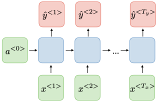
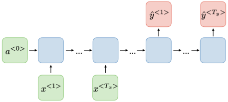
- Composed of **layers** of interconnected **nodes** or **neurons**, with each layer **transforming** the input data progressively to derive the output.
- Adjusts connection **weights** to minimize the **loss function**, which quantifies discrepancies between **predicted** values and **actual** targets, enhancing model **accuracy**.
- Various **types** including feedforward, recurrent, convolutional, each with specific **applications** and **capabilities**.

Recurrent Neural Networks



- **Recurrent neural networks** (**RNN**s) are a type of neural network designed to handle **sequential** data.
- They have **memory** due to the **recurrent** connections, allowing them to process **sequences** effectively.
- Popular **variants** include long short-term memory (LSTM) and gated recurrent unit (GRU).

Different RNN Types and Applications

Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network
One-to-many $T_x = 1, T_y > 1$		Music generation
Many-to-one $T_x > 1, T_y = 1$		Sentiment classification
Many-to-many $T_x = T_y$		Name entity recognition
Many-to-many $T_x \neq T_y$		Machine translation

Forecasting with Neural Networks

- **Neural forecasting** utilizes artificial neural networks to **predict** future outcomes by adapting to **complex** data patterns.
- These networks excel at recognizing **non-linear** patterns and relationships, essential for **accurate** predictions.
- Specifically, **RNNs** are ideal for time series forecasting as they retain context from **previous** data points, enhancing their predictive accuracy.
- Recent advances in **large language models** (LLMs) **enhance** RNN models for time series prediction **accuracy** through improved pattern recognition.

Statistical Models vs. RNNs for Time Series Analysis

- **Statistical models** like `SARIMA` are **simpler**, effectively handling **linear** trends and seasonality with superior **interpretability**, making them preferred for applications where model **understanding** and **transparency** are critical.
- **RNNs** excel in managing **complex**, **non-linear** patterns and **longer** sequences, providing greater **flexibility**, making them ideal for dynamic environments where **adaptability** to **new** patterns is crucial.
- **Empirical** studies show that although RNN models are **not** perfect, they are **competitive** alternatives to statistical models in many situations.
- The **choice** between statistical models and RNNs **depends** on the data **characteristics** and the specific **needs** of the forecasting task.

Effective RNN Models for Time Series Forecasting

- **Understanding Architecture:** Comprehensive knowledge of RNN architectures is crucial—not just selecting and using a model.
- **Training Process:** Effective training requires careful consideration of network parameters and training techniques.
- **Feature Integration:** Skillful combination of features can significantly enhance prediction accuracy.
- **Iterative Improvement:** Continuous testing and refinement are key to optimizing model performance.

Popular Auto-Predictor Models for Time Series Forecasting

- **Prophet by Facebook:** Tailored for forecasting with strong seasonal effects, using an additive model that handles daily, weekly, and yearly seasonality.
- **Amazon Forecast:** A managed service that uses machine learning to generate accurate forecasts, easy to use without ML expertise.
- **Google Cloud AutoML Tables:** Automates the building of ML models on structured data, ideal for both general predictive modeling and time series forecasting.

Further Reading

RNNs in Time Series Forecasting

- H. Hewamalage, C. Bergmeir, and K. Bandara, Recurrent neural networks for time series forecasting: Current status and future directions, *International Journal of Forecasting*, 37(1):388-427, 2021

DeepAR

- D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, DeepAR: Probabilistic forecasting with autoregressive recurrent networks, *International Journal of Forecasting*, 36(3):1181-1191, 2020

Deep Learning in Time Series Forecasting

- K. Benidis et al., Deep learning for time series forecasting: Tutorial and literature survey, *ACM Computing Surveys*, 55(6):1–36, 2022

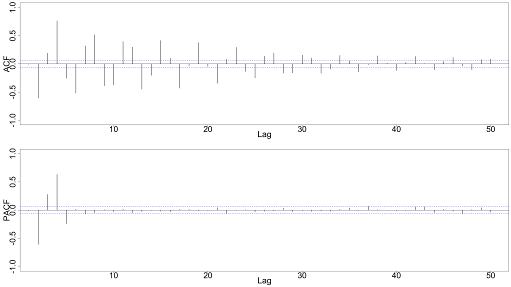
Outline

Autoregressive Model

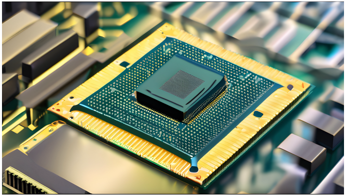
- In 1976, Box and Jenkins introduced the **autoregressive** (AR) model and its variations like SARIMA, simple yet **widely** applied methods for time series analysis across fields spanning business, health, and engineering.

- An **autoregressive** model of **order** p, denoted by $\mathtt{AR}(p)$, is described by the dynamic equation $X_t = \phi_0 + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + W_t$ where W_t represents **white noise** series with mean zero and variance σ_W^2

- ACF and PACF plots are used to **fit** an AR model with an appropriate **estimate** of the order of the model.



Fitting AR Models in Big Data Regime



- In problems involving **big time series data**, fitting an appropriate AR model requires solving **many** potentially large scale **ordinary least squares** (**OLS**) problems.

Question

Can a **randomized sub-sampling** algorithm be designed to greatly **speed-up** such model fitting for **big** time series data?

Large OLS Problems

- Many **statistical models** lead to OLS problems (cf. Lecture Slides Chapter 2)

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{x} - \boldsymbol{Z}\boldsymbol{\beta}\|^2 \;=\; \|\boldsymbol{x} - \boldsymbol{Z}\boldsymbol{\beta}^{\star}\|^2$$

involving $n \times 1$ **observation vector** $\boldsymbol{x}$ and $n \times d$ **data matrix** $\boldsymbol{Z}$ $(n > d)$

- When $\boldsymbol{Z}$ is dense, the **solution cost** is $\mathcal{O}(nd^2)$

- In **big data** regimes with $n \gg d$, $\mathcal{O}(nd^2)$ is **too much**.

- **Randomized Numerical Linear Algebra** (**RandNLA**) has successfully employed **random sub-sampling** schemes to compress $\boldsymbol{Z}$ into a smaller matrix, while approximately **retaining** many of its original properties.

RandNLA Sub-sampling Scheme

- RandNLA subroutines involve **construction** of appropriate **sub-sampling** methods, and **compressing** the data matrix $\boldsymbol{Z} \in \mathbb{R}^{n \times d}$ into a **smaller** version $\widehat{\boldsymbol{Z}} \in \mathbb{R}^{s \times d}$ for $d \leq s \ll n$

- In the classical **OLS** problem, RandNLA can readily be **applied** to the smaller scale problem

$$\min_{\widehat{\boldsymbol{\beta}}} \|\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{Z}}\widehat{\boldsymbol{\beta}}\|^2 = \|\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{Z}}\widehat{\boldsymbol{\beta}}^\star\|^2$$

at much **lower** computational costs.

Theorem (Drineas et al., 2011)

If s is **large** *enough, for an* **appropriate** *sub-sampling method, with* **high probability**, *we have*

$$\|\boldsymbol{x} - \boldsymbol{Z}\boldsymbol{\beta}^\star\|^2 \leq \|\boldsymbol{x} - \boldsymbol{Z}\widehat{\boldsymbol{\beta}}^\star\|^2 \leq (1 + \mathcal{O}(\varepsilon))\|\boldsymbol{x} - \boldsymbol{Z}\boldsymbol{\beta}^\star\|^2 .$$

Data-oblivious Sub-sampling Scheme

- The **simplest** way to construct the **compressed** data matrix $\widehat{Z}$ is using **uniform** sampling, where s rows of the **original** data matrix Z are uniformly **sampled**, each with an **equal** probability of $\frac{1}{n}$

- In the presence of **anomalies** in data, such **uniform** sampling schemes perform **poorly**, and it can be **shown** that one may **require** $s = \mathcal{O}(n)$ for effective sub-sampling.

Data-aware Sub-sampling Scheme

Leverage Score Sampling

Among **various** sub-sampling schemes, those based on **statistical leverage scores** enhance the **theoretical guarantees** for matrix algorithms in challenging scenarios and support **high-quality** numerical implementations.

Definition

For an $n \times d$ data matrix $\boldsymbol{Z}$, the **leverage scores** are denoted by $\ell(i)$ for $i = 1, 2, \ldots, n$ and defined as the i^{th} **diagonal** element of the **hat** matrix $\boldsymbol{H}$:

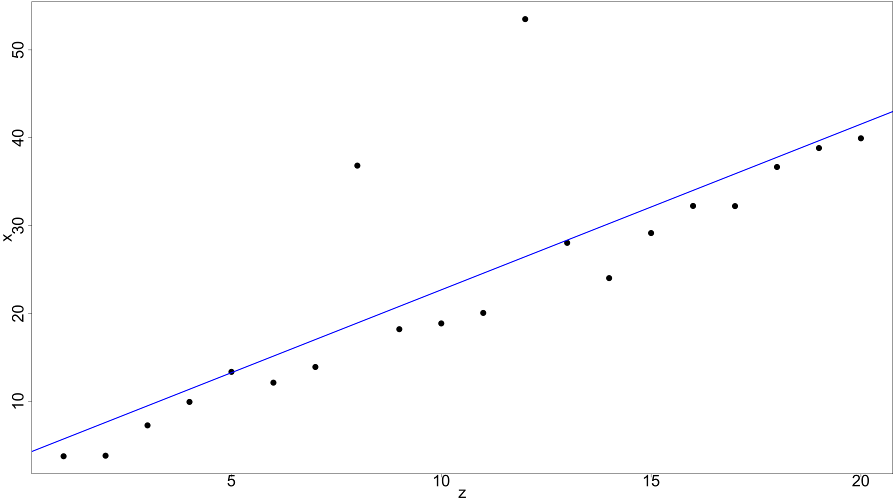
$$\boldsymbol{H} := \boldsymbol{Z}(\boldsymbol{Z}^\intercal \boldsymbol{Z})^{-1} \boldsymbol{Z}^\intercal$$

- It can be **shown** that $\ell(i) \geq 0$ for $i = 1, 2, \ldots, n$ and $\sum_{i=1}^{n} \ell(i) = d$, implying that

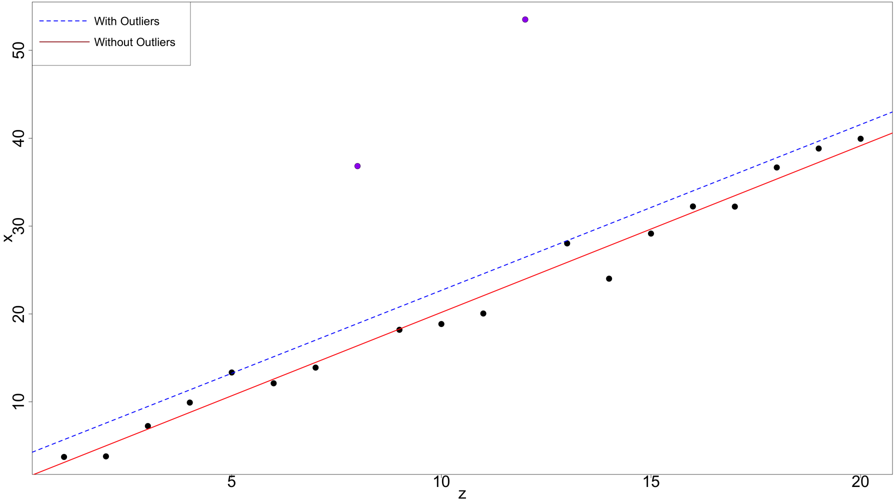
$$\pi(i) := \frac{\ell(i)}{d} \quad \text{for } i = 1, 2, \ldots, n$$

defines a **sampling distribution** over the rows of the data matrix $\boldsymbol{Z}$

Motivation: Outlier Detection in Linear Regression



Motivation: Outlier Detection in Linear Regression



Computational Complexity

- Clearly, **obtaining** the leverage scores is **as costly as** solving the original **OLS** problem, that is $\mathcal{O}(nd^2)$
- Some **randomized approximation** algorithms **estimate** the leverage scores in $\mathcal{O}(nd\log n + d^3)$ operations.

Question

Given the special **structure** of the data matrix Z in `AR` models, can we develop even **more efficient** algorithms to **approximate** the leverage scores?

Advancements in AR Model Fitting for Big Data

- We applied methods from **RandNL** to develop a new fast algorithm to estimate **leverage scores** in **big data** regimes, achieving accuracy within $(1 + \mathcal{O}(\varepsilon))$ of the true leverage scores with **high probability**.

- Utilizing these results, we created the **efficient** LSAR algorithm for AR models in **large-scale** time series data, ensuring theoretical **error bounds** and guaranteed **convergence**.

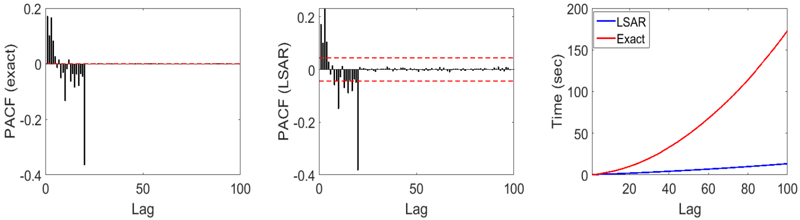
- The **exact** computation time for fitting an $\texttt{AR}(p)$ model is $\mathcal{O}(np^2)$, but our LSAR algorithm **reduces** complexity to $\mathcal{O}\left(np + \texttt{poly}(p/\varepsilon)\right)$, enhancing **efficiency**.

Further Reading

- A. Eshragh et al., LSAR: Efficient leverage score sampling algorithm for the analysis of big time series data, *Journal of Machine Learning Research*, 22:1-36, 2022

Synthetic Big Time Series Data: $\texttt{AR}(20)$

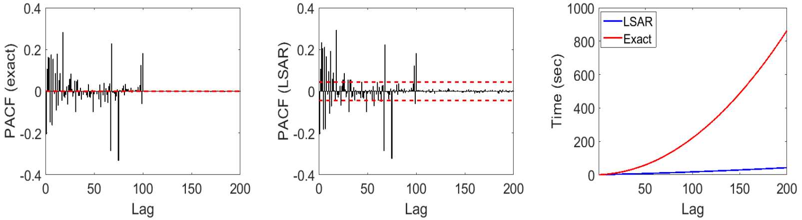
- An $\texttt{AR}(20)$ model with $n = 2,000,000$ and $s = 0.001 \times n = 2,000$



(a) Exact PACF (b) Estimated PACF (c) Computational time

Synthetic Big Time Series Data: $\mathtt{AR}(100)$

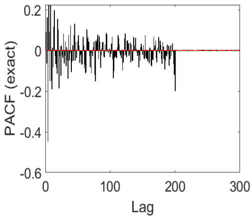
- An $\mathtt{AR}(100)$ model with $n = 2,000,000$ and $s = 2,000$



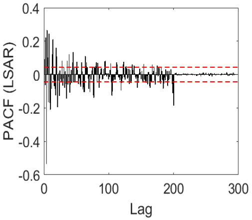
(a) Exact PACF (b) Estimated PACF (c) Computational time

Synthetic Big Time Series Data: $\mathtt{AR}(200)$

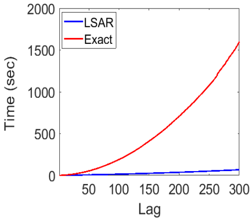
- An $\mathtt{AR}(200)$ model with $n = 2,000,000$ and $s = 2,000$



(a) Exact PACF



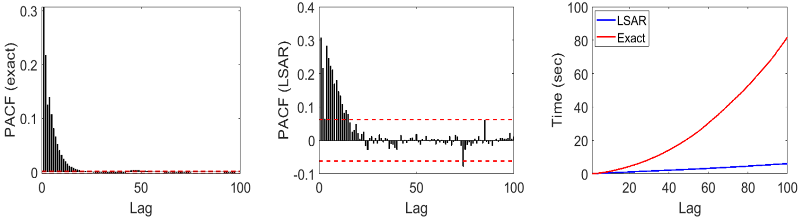
(b) Estimated PACF



(c) Computational time

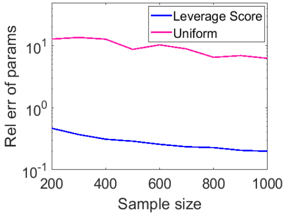
Real-world Big Time Series Data: Gas Sensors Data

- A real-world data with $n = 919,438$ and $s = 0.001 \times n$

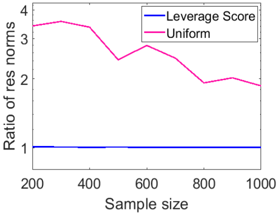


(a) Exact PACF (b) Estimated PACF (c) Computational time

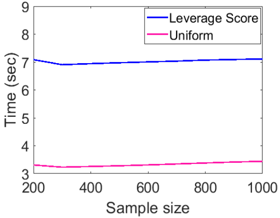
Leverage Score vs. Uniform Sub-sampling



(a) Relative error of estimates: $\frac{\|\hat{\phi}-\phi\|}{\|\phi\|} \times 100\%$



(b) Ratio of residuals norm: $\frac{\|\hat{r}\|}{\|r\|}$



(c) Computational time

Advancements in ARMA Model Fitting for Big Data

- Analogous to the LSAR algorithm, we have developed a **highly-efficient** algorithm, called LSARMA, for fitting ARMA models to big data with **provable guarantees**.

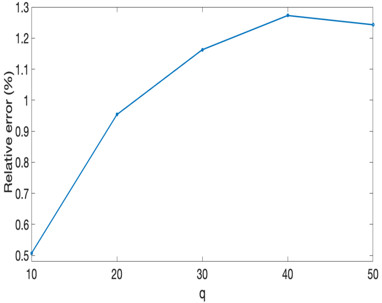
- The **exact** computation time for fitting an $\mathtt{ARMA}(p, q)$ model is $\mathcal{O}\left(np(p+q)^2\right)$, but our LSARMA algorithm **reduces** the complexity to $\mathcal{O}\left(n(p+q) + \mathtt{poly}((p+q)/\varepsilon)\right)$, enhancing **efficiency**.

Further Reading

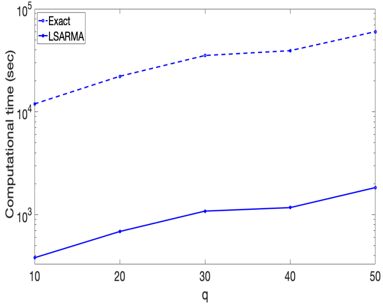
- A. Eshragh et al., SALSA: Sequential approximate leverage-score algorithm with application in analyzing big time series data, *arXiv preprint arXiv:2401.00122*, 2023

Synthetic Big Time Series Data: MA Models

- $\mathtt{MA}(q)$ models with $n = 5,000,000$ and $s = 0.002 \times n = 10,000$



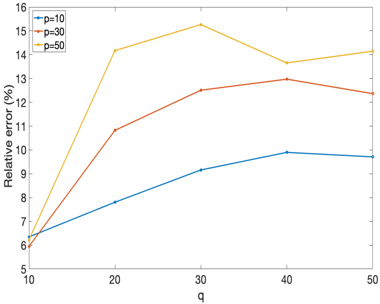
(a) Relative error of estimates: $\dfrac{\|\hat{\phi} - \phi\|}{\|\phi\|} \times 100\%$



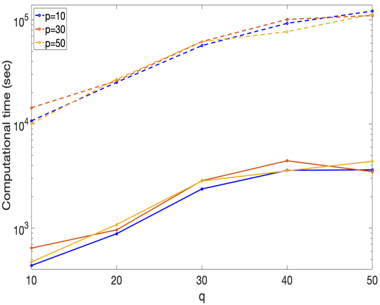
(b) Computational time

Synthetic Big Time Series Data: ARMA Models

- $\texttt{ARMA}(p, q)$ models with $n = 5,000,000$ and $s = 0.01 \times n = 50,000$



(a) Relative error of estimates: $\dfrac{\|\hat{\phi} - \phi\|}{\|\phi\|} \times 100\%$



(b) Computational time

Outline

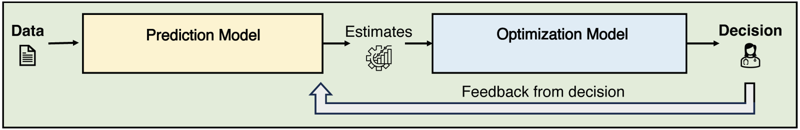
Multivariate Big Time Series Data

- Develop similar **efficient algorithms** for estimating the following models for large-scale **multivariate** time series data:
    - **Vector Autoregressive** (`VAR`) model: Forecasting weekly demand for items $\gamma_1, \ldots, \gamma_k$ at the retailer store η in location δ
    - **Matrix Autoregressive** (`MAR`) model: Forecasting weekly demand for items $\gamma_1, \ldots, \gamma_k$ at the retailer stores $\eta_1, \ldots, \eta_m$ in location δ
    - **Tensor Autoregressive** (`TAR`) model: Forecasting weekly demand for items $\gamma_1, \ldots, \gamma_k$ at the retailer stores $\eta_1, \ldots, \eta_m$ in locations $\delta_1, \ldots, \delta_l$

Ongoing Research Projects

- **Physics-informed Neural Forecasting Model:** Develop a hybrid physics-informed machine learning model where the `MAR` model captures the linear aspects of the multivariate time series data, assisting the machine learning model in more efficiently capturing the data's non-linearity.

- **Smart Predict-then-Optimize:**



Further Reading

- A.N. Elmachtoub and P. Grigas, Smart "Predict, then Optimize", *Management Science*, 68(1):9-26, 2021

References

1. R.H. Shumway and D.S. Stoffer, *Time Series Analysis and Its Applications With R Examples*, Springer, New York, 2017
2. S. Amidi, *Online Resources on Deep Learning*, Accessed on 28.04.2024