Machine Learning

Notes on Time Series

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

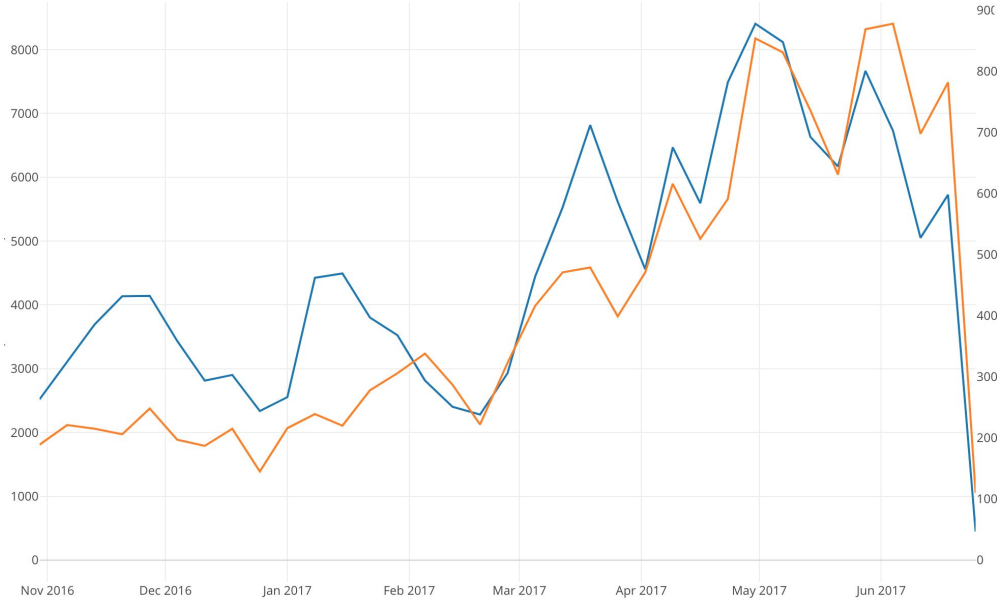
Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Time Series Analysis

- There are many time-based problems that are of interest
  - Weather data
  - Rainfall measurements
  - Temperature readings
  - Heart rate monitoring (EKG)
  - Brain monitoring (EEG)
  - Quarterly sales
  - Stock prices
  - Automated stock trading
  - Industry forecasts
  - Interest rates

Time Series Visualization

- The most basic and commonly used visualization for time series data.
- Time on x-axis. Connect line segments for observing trends, patterns, and outliers over time.
- There are many tools (see)
  - Seasonal Decompose
    - Library
      from statsmodels.tsa.seasonal
      import **seasonal_decompose**
  - Autocorrelation and Partial Autocorrelation Plots
  - Heatmaps
  - Time Series Decomposition Plots



Img src

Algorithms for time series

- Problems
  - Temporal problems can be video, audio, text or tabular data
    - Video has both spatial and temporal info (spatio-temporal)
- Algorithms
  - There are some statical (e.g. ARIMA) and ML algorithms for tabular data
  - Some algorithms process **step by step** such as LSTM
  - Other algorithms can process the **whole sequence** as one unit
    - Some video classification netowors process e.g. 64 frames at once
    - **Transformers** can process a whole sequence at once
  - Another direction is to **summarize** the sequence as a single feature vector
    - Then use normal ML algorithms such as tree-based, NN, Linear models, etc

Features Summarization

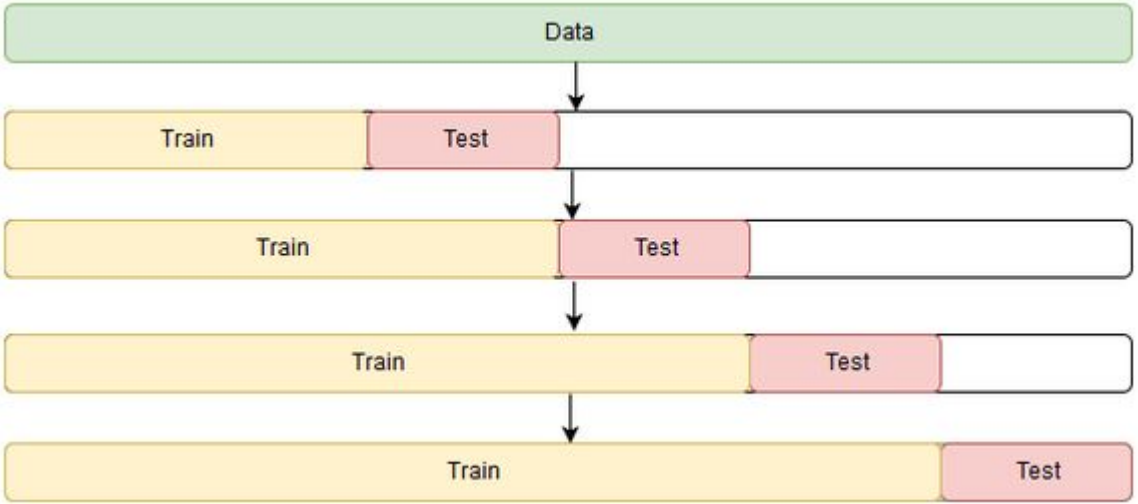
- Imagine tabular data for some daily 4 values (e.g. stock indicators)
- You would like to make a regression for the next day 4 values
- One can decide to use a **window** of 10 days to predict the next day
  - A *hyperparameter*: Short window focuses on narrow data.
    - Very large window might be affected with irrelevant data
- How can I extract features from the last D days?
  - You may extract statics such as min, max, avg, std, median
  - Extract Exponentially **Weighted** Mean (EWM) features on longer windows for **trend**
  - Divide your big window (e.g. 100 elements) into small subwindows (e.g 20)
    - Then we have 100/20=5 subwindows. Compute stats per window and concatenate
  - Seasonal based features (e.g. select specific days / weeks / weekend, etc)
  - In general: **Trend** (direction over time), **Seasonality** (**regular** intervals e.g. daily, monthly, or quarterly), **Cyclic Patterns** (e.g. long term economic conditions)

Data Split

- It is very important to carefully split the data into train/val/test in a way that doesn't cause **data leakage**
- You must divide the data based on time
  - **You can't have future samples in the train data**
  - For example: first 10 months for train, next for validation and next for test
    - In other words, make sure to sort the tabular data by date
    - Overall; test similar to the actual production flow
  - Production model probably is trained on ALL the recent data (no or limited val)
    - To be ready for tomorrow prediction

Data Split

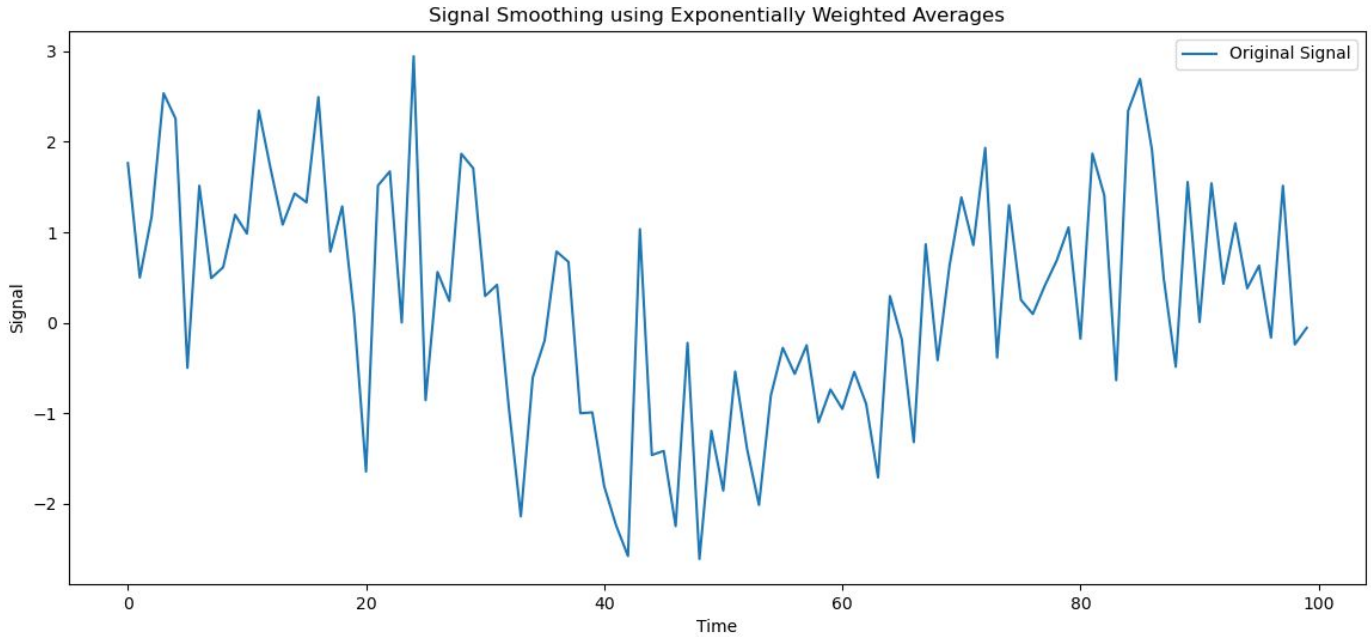
- Create one dataset at a time on historical data
- Make sure the model is good in different intervals



Img src

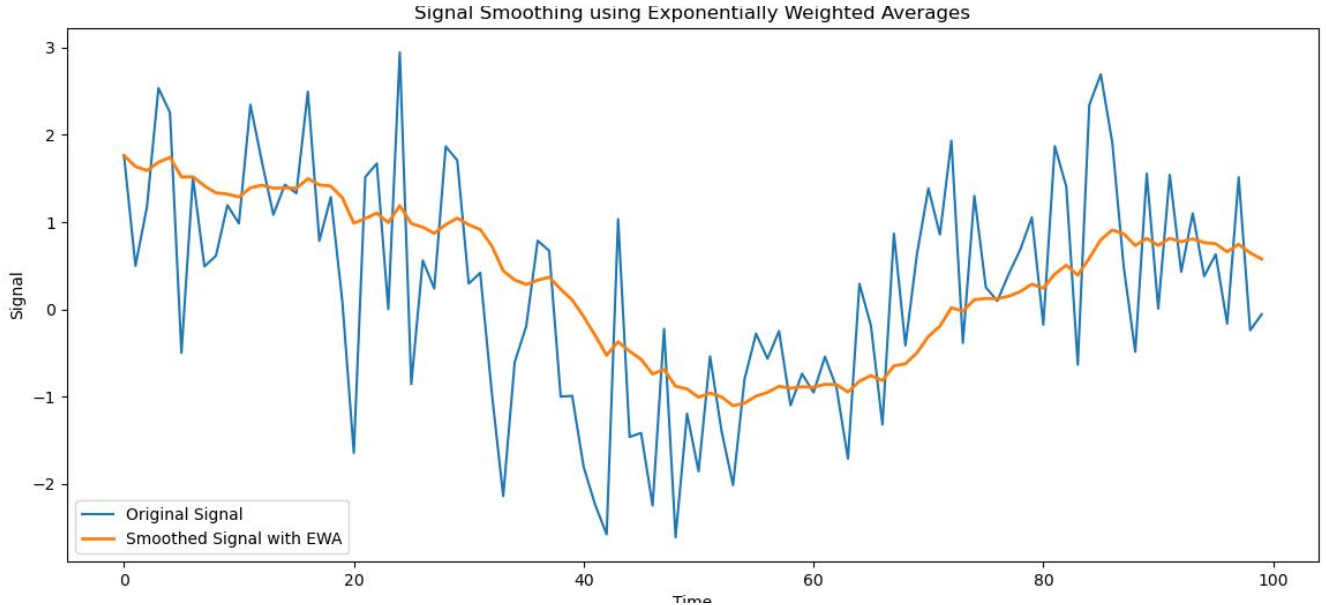
Data Smoothing

- Sometimes, one of your temporal features fluctuate a lot and it might be a better idea to **smooth the signal** in terms of the previous values!



Data Smoothing

- What if we can smooth with this way? This kind of smoothed data can make learning more robust and generalizable



Data Smoothing

- There are several techniques for that (e.g. a moving average window where you consider compute the average of the window)
- One popular technique that can use all the history (or subset of it) and let you control the effect of recent samples vs old samples is called Exponentially Weighted Moving Average (EWMA)
- Start with ewma[0] = data[0]
- Then at each step compute this formula

$$\mathrm{EWMA}_t = \alpha * r_t + (1 - \alpha) * \mathrm{EWMA}_{t-1}$$

Exponentially Weighted Moving Average

- Alpha (smoothing factor) parameter controls how important the current observation is in the calculation of the EWMA.
  - Alpha = 1 ⇒ Just use the current signal (no history)
  - Alpha = 0.9 ⇒ Most of the weight from the current step
  - Alpha = 0.2: The history has great effect
- To get insights, you can keep expanding the equation

$$\mathrm{EWMA}_t = \alpha * r_t + (1-\alpha) * \left(\alpha * r_{t-1} + (1-\alpha) * (\alpha * r_{t-2} + (1-\alpha) * \mathrm{EWMA}_{t-3})\right)$$

- Observe the ewma[t-k] has weight: $\alpha * (1-a)^k$

Exponentially Weighted Moving Average

- We can simply code this idea as follows

```
# Exponentially Weighted Averages
def ewa(data, alpha=0.9):
    smoothed_data = np.zeros(data.shape)
    smoothed_data[0] = data[0]
    for i in range(1, len(data)):
        smoothed_data[i] = alpha * smoothed_data[i - 1] + (1 - alpha) * data[i]
    return smoothed_data

alpha = 0.9
smoothed_signal = ewa(signal, alpha)
```

N-Day EWMA

- If you want a sliding window of only N values, and want to compute EWMA, there is a suggested formula for the alpha: 2 / (N+1)
  - N = 1 ⇒ alpha = 2/2 = 1
  - N = 2 ⇒ 2 / 3
  - N = 20 (days) = 2 / 21 = 0.09

Pandas

- Pandas allows you to compute ewm
  - Using alpha or span (ndays)
  - I don’t know why their alpha doesn’t generate outputs as I expect!
  - adjust=true, use some normalized equation. Check the API

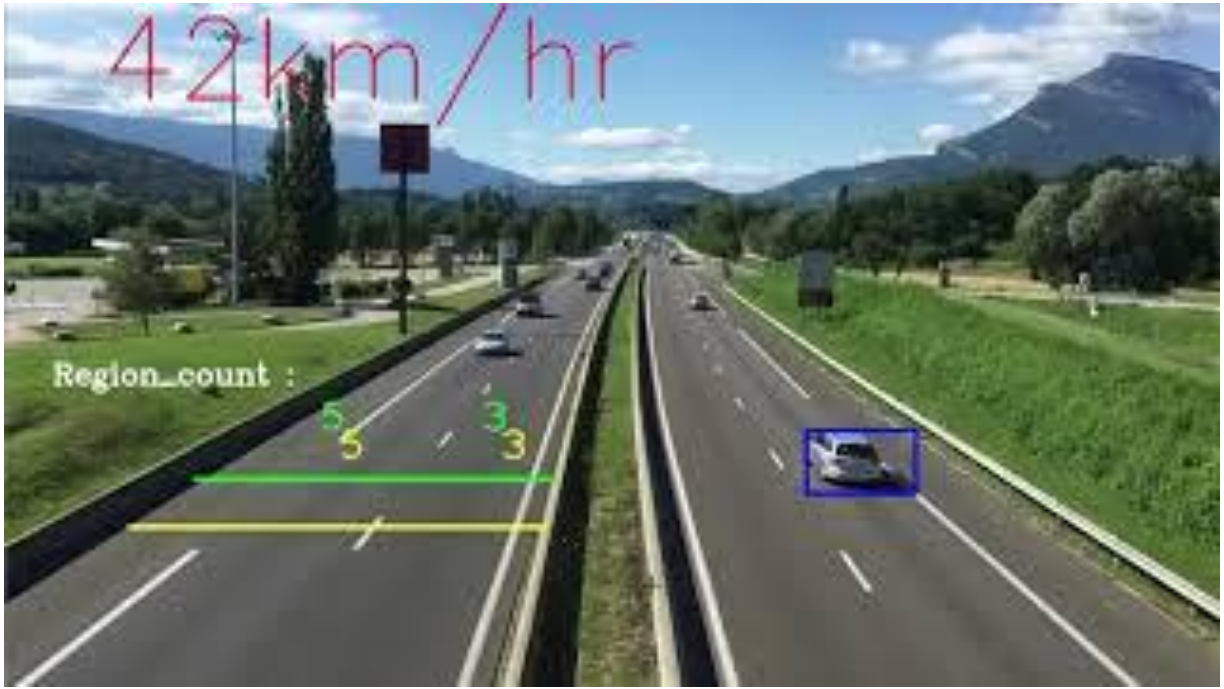
```
smoothed_signal = signal_series.ewm(alpha=alpha, adjust=False).mean()
smoothed_signal = signal_series.ewm(span=10, adjust=False).mean()
```

Off-Topic: Temporal Outputs

- In tasks with temporal outputs, we also suffer from fluctuations
- Imagine a task where, we would like to monitor the car driver and issue an alert if he uses a phone during the drive
  - Input: series of frames
  - Output: series of binary classifications
  - Any classifier will make a mistake, so imagine output like: 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
    - Probably these two ones are just classifier mistake as corresponding frames are 0
- Another problem: car bbox detection in videos

Off-Topic: Temporal Outputs - Car Detection

- Problem 1: Detector may fail to detect in some frames (but we have history)
- Problem 2:
  - Fluctuation



Off-Topic: Temporal Outputs - Solutions

- **Majority voting** for classification: select a window (e.g. last 10 frames) and make voting among them
- **State machine** for classification: develop a state machine that decides when to flip between different statuses
  - Debounce Signal: use a timer mechanics: don't switch until specific duration is passed
- To **stabilize** the bounding boxes of detected cars:
  - For every corner in the box do Temporal Smoothing (moving average or EWMA)
  - Use **Kalman Filter** to predict the missing bboxes or interpolate between available ones
    - **Optical Flow** may help
  - Use **tracking** instead of heavy tracking
    - It is common to do matching by solving the sum **assignment problem** (scipy)

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."