NUS Fintech Society

Machine Learning

Time Series

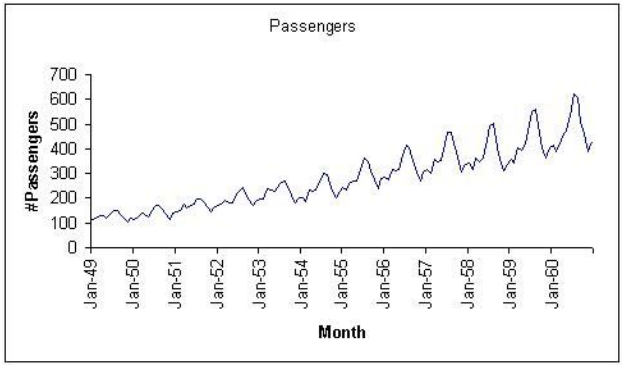
Content

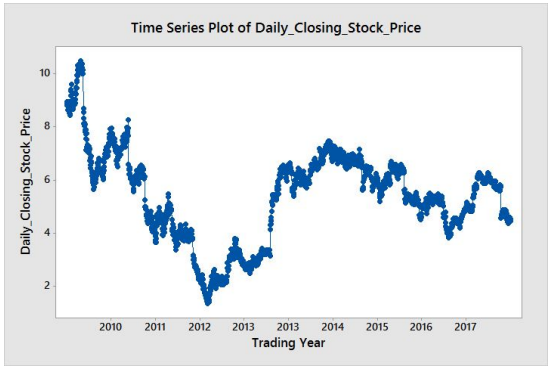
- Intro to Time-Series
- Specific models for time-series data
  - Autoregressive Model (AR)
  - Moving Average Model (MA)
  - ARMA and ARIMA
- “Exotic” models for time-series data
  - Search
  - Filtering
  - Neural Networks

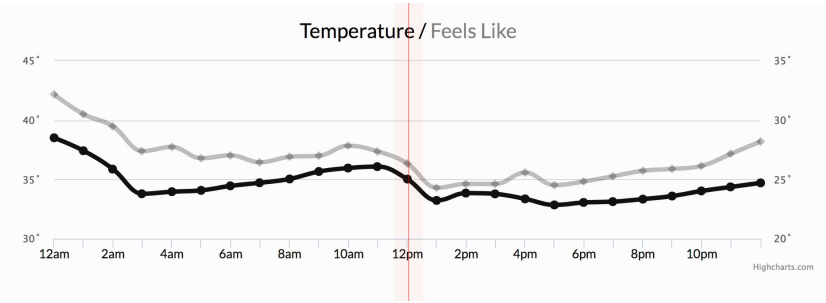
Intro to Time Series

Closing Price over Time

- A time series is a sequence of numerical data points in successive order

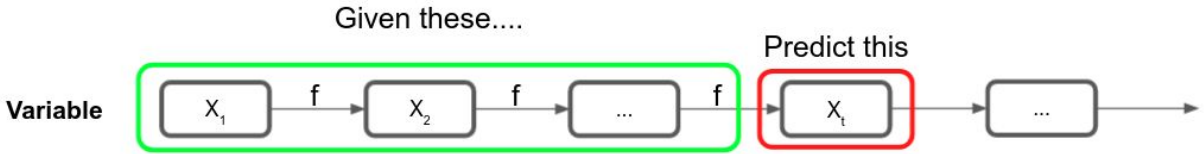






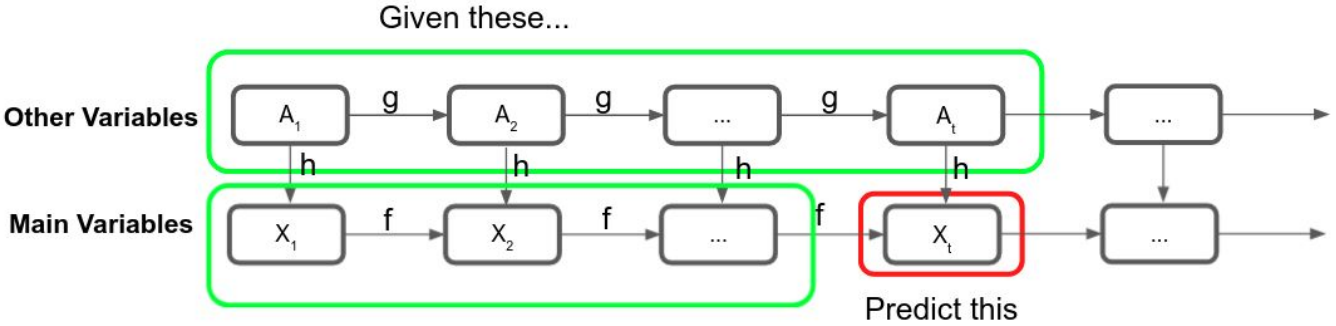
What Are We Solving?

- Problem Statement 1:



Example: Stock Prediction

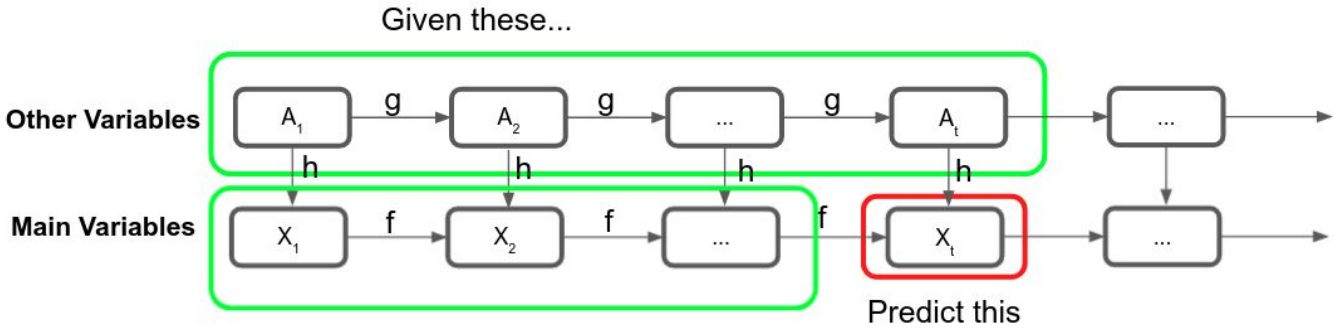
- Problem Statement 2:



Example: Stock Prediction + Consumer Sentiment Information

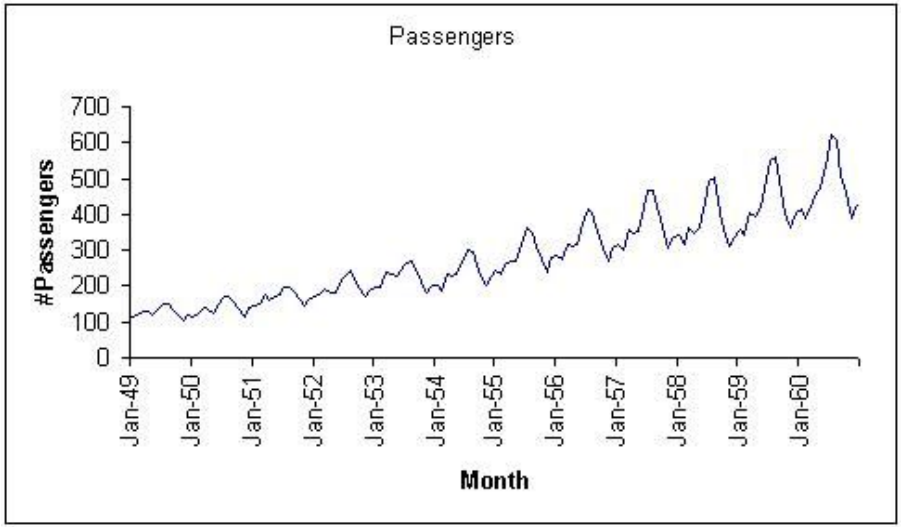
Why is it Difficult?

- Problem: Both problem statements are ill-posed!
  - Problem Statement 1:
    - How many “yesterdays” actually affect today?
    - What if the function f itself changes over time?*
    - What if past values don’t predict future values?
    - And more...
  - Problem Statement 2:
    - What if A is not observable? E.g. Consumer Confidence



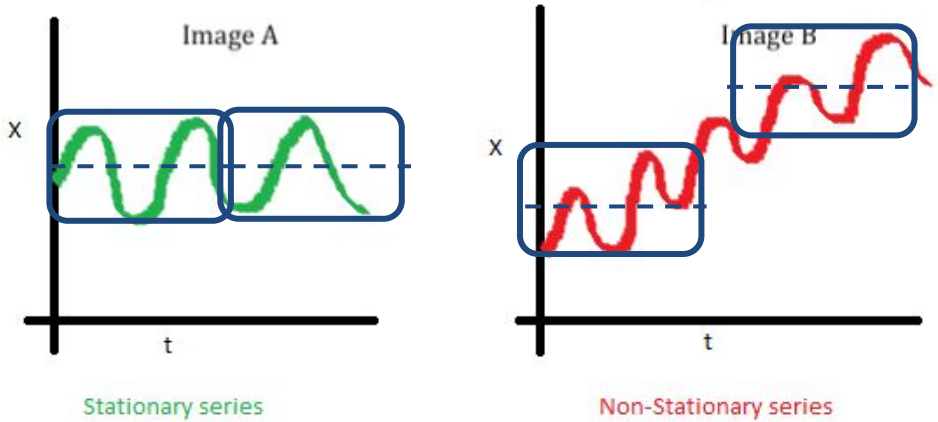
Characteristics of Time Series

- Characteristics to consider:
  - trend
  - seasonality: any periodic traits
  - outliers → This might make fitting the general trend harder



Required Assumptions

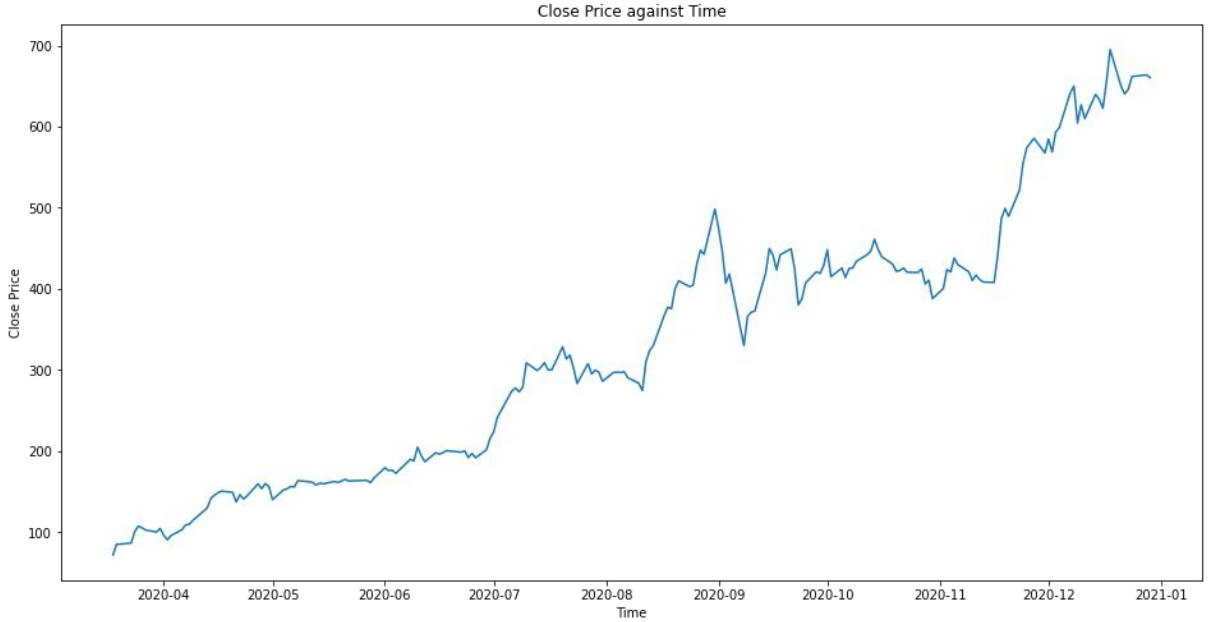
- Assumption 1: The past affects the present (and thus future)
- Assumption 2: Only finite k past values affect the present
- Assumption 3: No function changes over time
- Assumption 4: Stationarity
  - Needed for the theory to work
  - Achievable by *differencing*



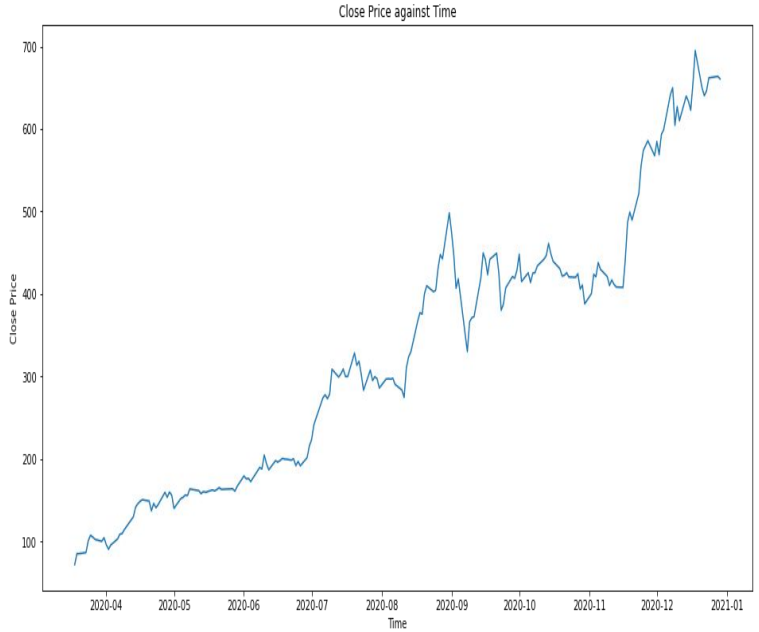
AR, MA, and ARIMA

Closing Price over Time

- Let’s look at Tesla’s closing price over time, from Apr 2020 to Dec 2020



Autoregressive Model



- The trend looks suspiciously like a linear trend.
- What is the independent variable? Can we use date?
- What if we use *past stock price* to predict *future stock price* instead?
- This approach is called **Autoregressive (AR)** model.

Autoregressive

- Key assumptions: *Relationship is linear*
- Essentially linear regression, with past data as independent variable/s and future data as dependent variable
  - Linear Regression: $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$
  - AR(3) Model: $X_t = c + \theta_1 X_{t-1} + \theta_2 X_{t-2} + \theta_3 X_{t-3}$
- There is one parameter to be set manually in an AR model: the *order* of the AR model. Denote this parameter as *p*
- The value of *p* DOESN'T mean only *p* past values affect the present
- It says that knowing *p* past values is sufficient in predicting the present
- Simple extension => change linear to some Neural Network

Dangers of AR Model

- Markov Property: today depends only on yesterday

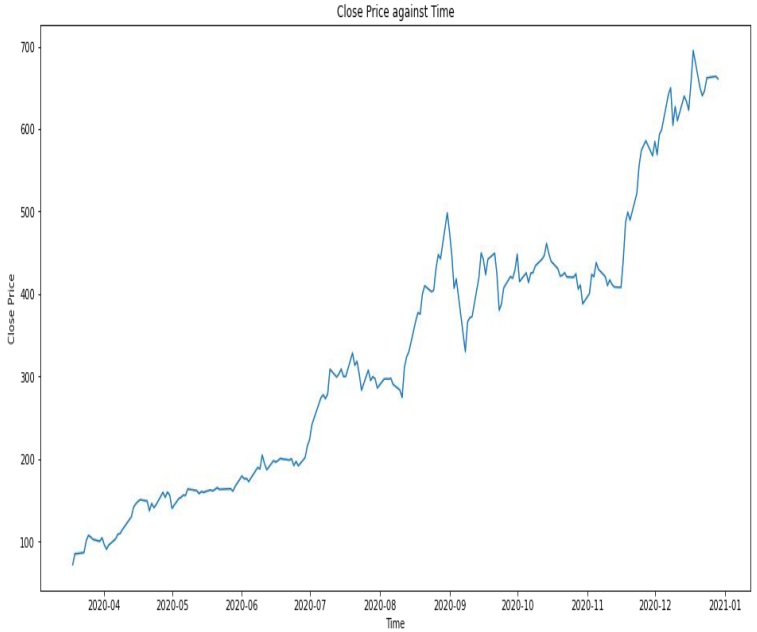
$$P(X_n = x_n \mid X_{n-1} = x_{n-1}, \ldots, X_0 = x_0) = P(X_n = x_n \mid X_{n-1} = x_{n-1}).$$

- Martingale Property: Best guess for today is just yesterday's value

$$\mathbf{E}(X_{n+1} \mid X_1, \ldots, X_n) = X_n.$$

Disclaimer: the above descriptions do injustice to the real mathematics and statistics behind them. Please read more if you are interested

Moving Average Modelling



- We used *past stock price* to predict *future stock price* in AR
- What if we use *past errors* to predict *future stock price* instead?
- This approach is called **Moving Average (MA)** modelling.

Intuition behind MA

"Algorithm":

1. Day 1:
   a. Start with today's stock price
   b. Predict tomorrow's stock price
2. Day 2:
   a. Messed up. Calculate error ϵ in prediction
   b. Predict tomorrow's stock price based on today's actual stock price p_2, plus $k\epsilon$

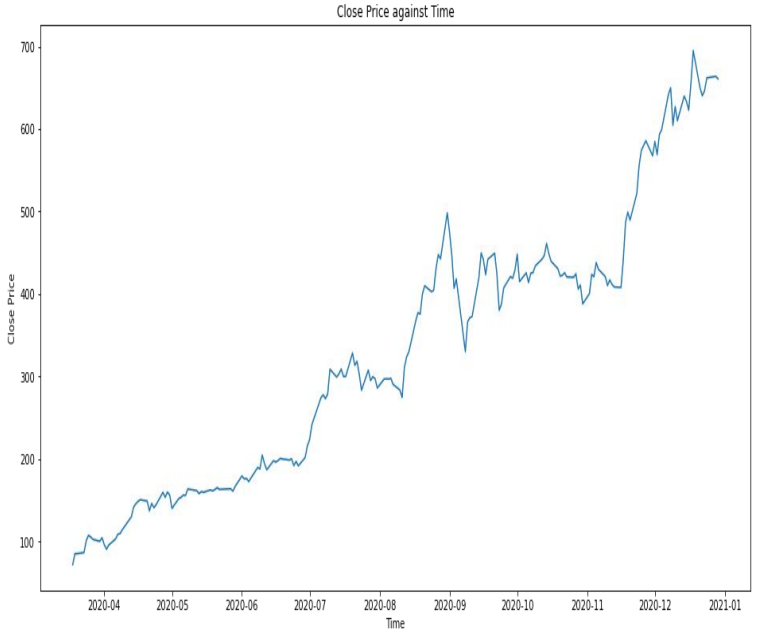
1. Day 3:
   b. Messed up. Calculate error ϵ in prediction
   c. Predict tomorrow's stock price based on today's actual stock price p_3, plus $k\epsilon$

1. Repeat for all days. Then calculate average error made for the given k value.
2. Find optimum k such that average error is minimum

Intuition behind MA

- *Order* of the MA model. To differentiate it from the order of AR, we denote the order of MA as *q* instead
- How to find this parameter *q*?
- Benefits:
  - the intuition is relatively simple
  - it is conceptually a linear regression:
    - Linear Regression: $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$
    - MA(3) Model: $X_t = \mu + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \theta_3 \epsilon_{t-3}$
- Problem:
  - Need error to make prediction
  - But need prediction to compute error!
  - In other words: error term is not observable

Closing Price over Time



Looking at this data, do we use AR or MA model?

By this point you should know the answer to this question:

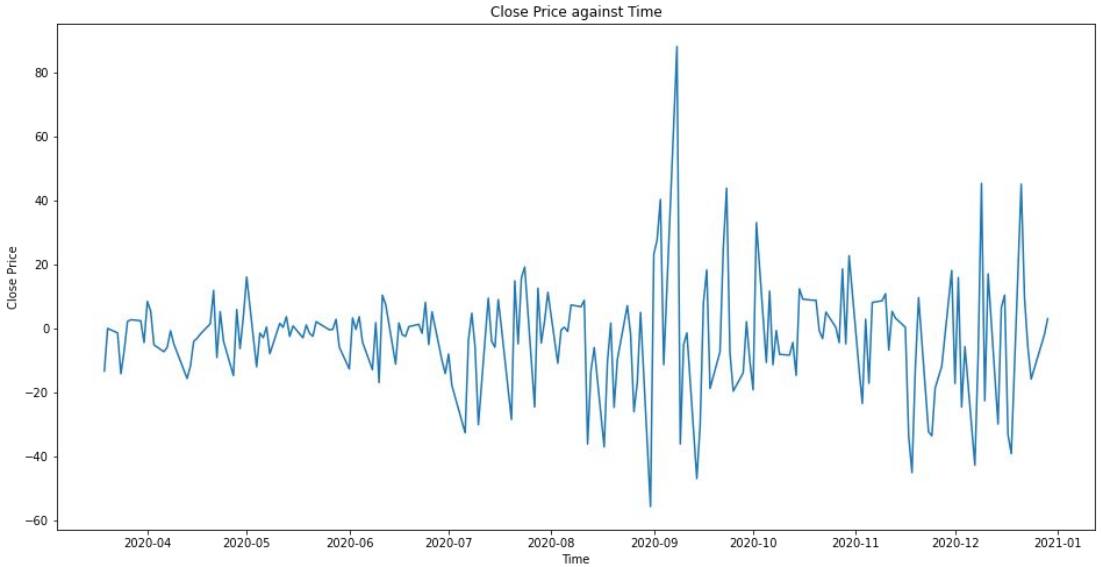
Try both!

ARMA and ARIMA

- There's nothing wrong with trying to fit both AR and MA model to the data at the same time
- This is called the ARMA model (no points for guessing why)
- There are two parameters to be set manually in ARMA:
  - p: the order of the AR model
  - q: the order of the MA model
- Recall that the theory behind fitting time-series data requires *stationarity* (constant mean)
- We can do *differencing* in order to get a time series whose mean does not change. We then apply ARMA to the *differenced time series*

B	0	1	1	2	3	5	8	13	21	…
A	NaN	0	1	1	2	3	5	8	13	…
B - A	NaN	1	0	1	1	2	3	5	8	…

ARMA and ARIMA



After differencing, with *d = 1*. Mean looks more or less constant.
We can do statistical analysis to check this

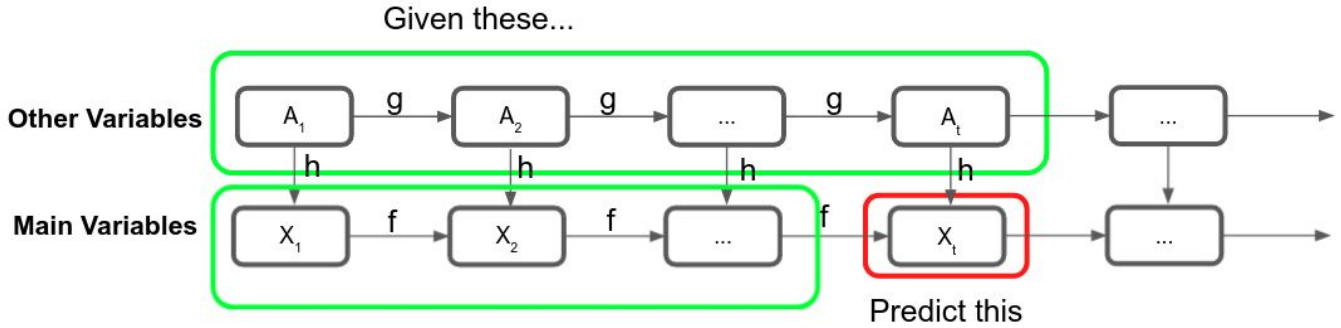
ARMA and ARIMA

- The process of differencing, followed by fitting the ARMA model is called the ARIMA model, where I stands for “integrated”
  - Integrated because we technically fit the differentiated data, and thus must integrate it *d* times to get back to the original time series data
  - ARIMA model has 3 parameters to be set: [*p, d, q*]
    - p for Autoregressive model
    - d for differencing and summing to get predictions
    - q for Moving Average model
  - How do we set *p, d, q*? for loops…

Exotic Models

Recall Our Problem

- Problem Statement 2:

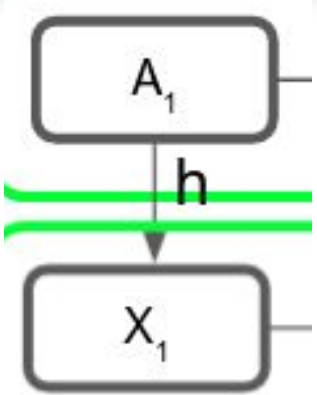


Example: Stock Prediction + Consumer Sentiment Information

- But sometimes variable A is not observable! E.g. Consumer Sentiment, etc.
- Possible Solution:
  - Ignore: Frame the problem as problem statement 1 instead
  - Indices: Find some index that proxies Consumer Sentiment

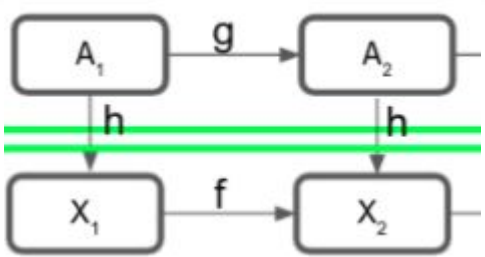
Math / CS Way of Solving

- A is not observable! But we know A can produce X when using function h
- Reduce to Search Problem: Search A such that h(A) = X
  - Discrete A: Binary Search
  - Continuous A:
    - Gradient Descent
    - L-BFGS
    - Etc…



Math / CS Way of Solving

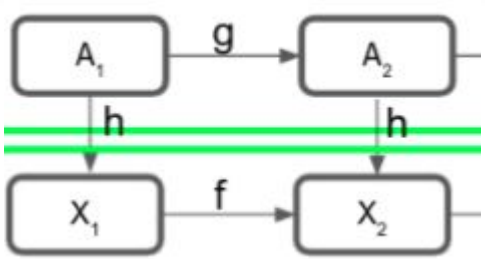
- Reduce to Search Problem:
  - Hard Constrained Search: search A s.t:
    - h(A2) = X2, AND
    - g(A1) = A2
  - Soft Constrained Search: search A s.t.:
    - ||h(A2) - X2|| < e AND
    - ||g(A1) - A2 || < k
  - Loss function: search A s.t.
    alpha * ||h(A2) - X2|| + beta * ||g(A1) - A2||
    is minimized



- Possible approaches: Binary Search, Gradient Descent, etc.

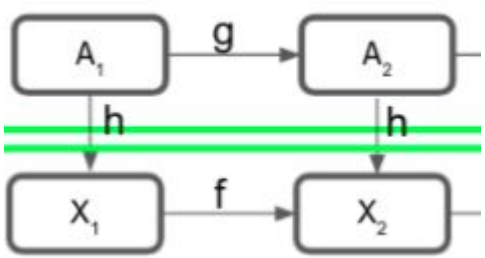
Engin. Way of Solving

- A is not observable! But we know A can produce X when using function h
- Solution: Particle Filter
- Time Step 1:
  1. A can be anything, so make K guesses on what A is
  2. Observe value of X1
  3. For each K guesses we make on A, pass it through h to obtain h(A_i)
  4. Weigh each guess according to how close h(A_i) is to X1 => closer means higher weight
  5. Choose with replacement which of the K guesses can continue. Probability of choice proportional to weight.



Engin. Way of Solving

- A is not observable! But we know A can produce X when using function h
- Solution: Particle Filter
- Time Step 2 and so on:
  1. We have a set of K (better) guesses on A1
  2. Set of K guesses of A2 by passing each guess through g
  3. Repeat the same procedure as before to eliminate unlikely candidates of A2
  4. Continue doing this for all time steps

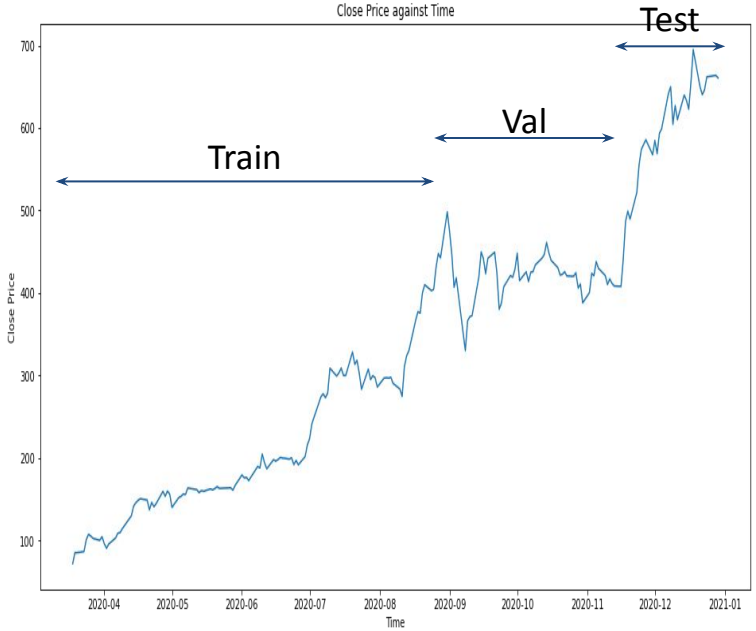


Result: Terrible guess for A1, A2, … Eventually all K guesses are similar => probably correct

Deep Learning Way of Solving

- Roughly: NN == universal function approximator
- See a function => Probably can be replaced with NN (given training data)
  Input-output pair can be (X1, X2), (X2, X3), etc.
  Many Creative ways to do this..

- How to split train-validation-test?
  - Random => WRONG
  - Stratified by time



Summary

Summary

- Basics of time-series data:
  - Formalisation of the problem to be solved
  - Characteristics of time-series data
  - Possible approaches to model time-series data
- Specific approaches for time-series data:
  - Autoregressive Model (AR) → basically Lin.Reg
  - Moving Average Model (MA) → basically Lin.Reg
  - ARMA and ARIMA
- ‘Exotic’ approaches:
  - Search
  - Particle Filter
  - Neural networks

What Now?

- While we briefly covered the importance of *analysing and preparing data*, there are still much that one can learn in this aspect:
  - *Decomposition of Time-Series data*:
    - *Trend*
    - *Seasonality*
  - *Feature Selection, Feature Engineering, Feature Processing*
  - *Markov property vs Martingale property*
  - And more…
- Knowing them reduces time spent on project and may improve performance e.g. using big brain *ACF* and *PACF* instead of for loops: $O(n^k) \rightarrow O(\log n)$
- We briefly discussed Gradient Descent for optimising differentiable models. This brings us to *neural networks*. They also use GD to optimise their parameters, but the differentiation is done by the *backpropagation* algorithm, usually abstracted out in libraries like *TensorFlow, Pytorch, JAX, etc*.

Resources

- ACF and PACF in Python: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/
- ARIMA in Python: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
- Time Series Forecasting as Supervised Learning: https://machinelearningmastery.com/time-series-forecasting-supervised-learning/

Thank You!