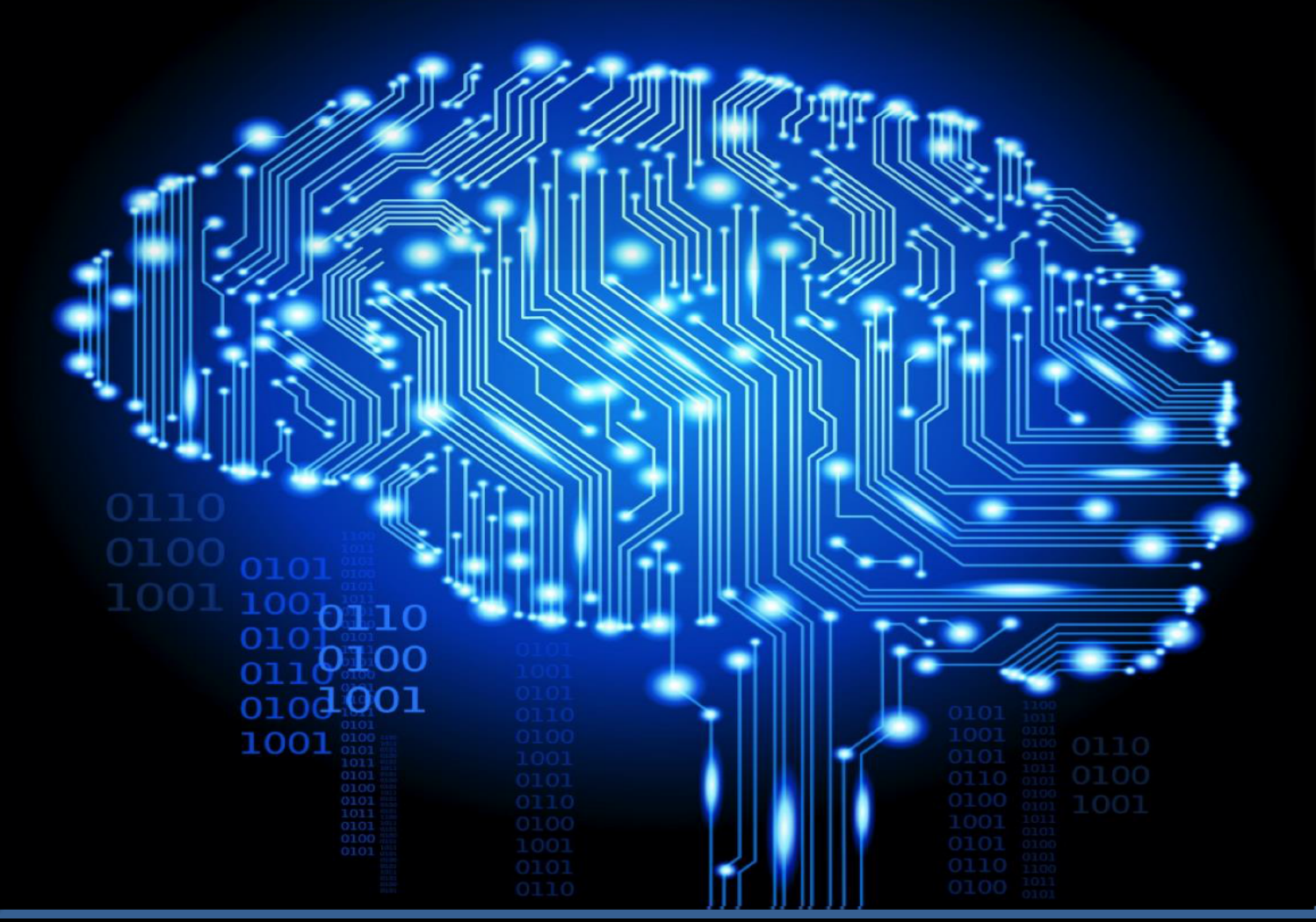


Statistical Learning and Computational Finance Lab.
Department of Industrial Engineering
http://slcf.snu.ac.kr

Day 12: Time Series Models & DL with Financial Data

Agenda

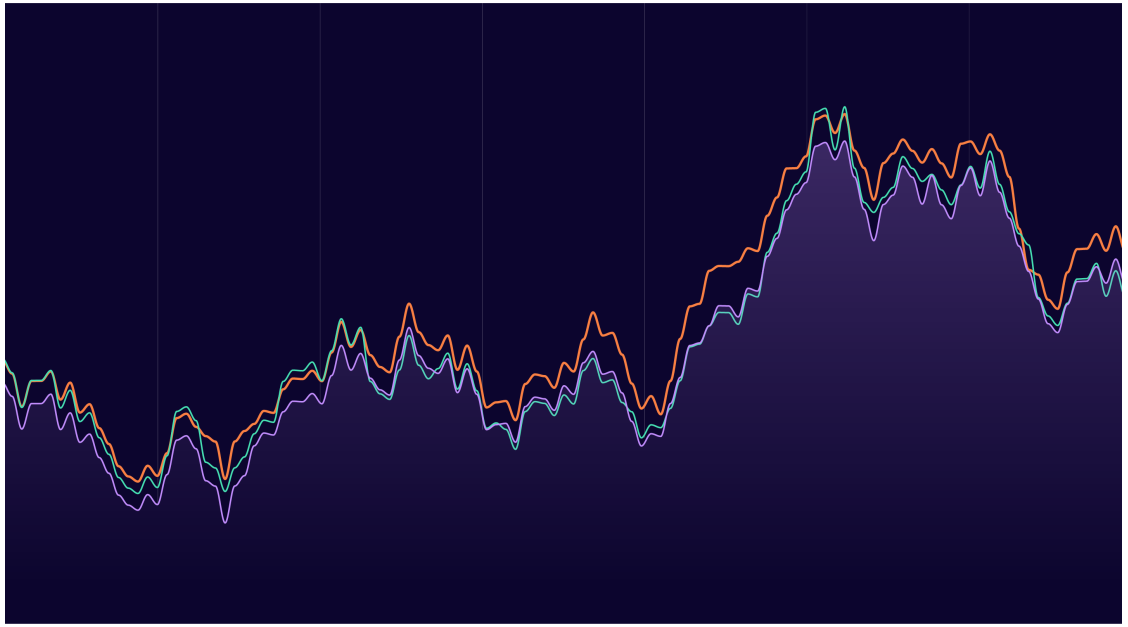
- Time Series Data and its Properties
- Traditional Time Series Models
- ML based Time Series Models
- Deep Learning in Finance

Time Series Data and its Properties

Image Source: https://www.space.com/time-how-it-works

What makes time series data special?

- There are some special components
  - Trend component
  - Seasonal component
  - Cyclical component
  - Irregular component



Source: https://www.dummies.com/article/technology/information-technology/data-science/big-data/key-properties-of-a-time-series-in-data-analysis-141265/

Why it's difficult to handle?

- Additional dimension!
  - We are living in a 4D space
- Information and trend might change as time goes
  - Stock price of Tesla
  - Sudden increase in COVID-19 cases in South Korea
- Infinite and vague range of data
  - Let's take stock price prediction for example
    - How do you decide the valid window length for the analysis?
    - Is there a rule of thumb?

Classical Approach for Time Series Analysis

Time domain analysis	Width, step, height of signal
Frequency domain analysis	Fourier analysis or wavelets
Nearest neighbors analysis	Dynamic time warping
Probabilistic model	Language modeling
(S)AR(I)MA(X) models	Autocorrelation inside of time series
Decomposition	Time series=trend + seasonal part + residuals
Non-linear Dynamics	Differential Equation
Machine Learning	Use ML model with hand-made features

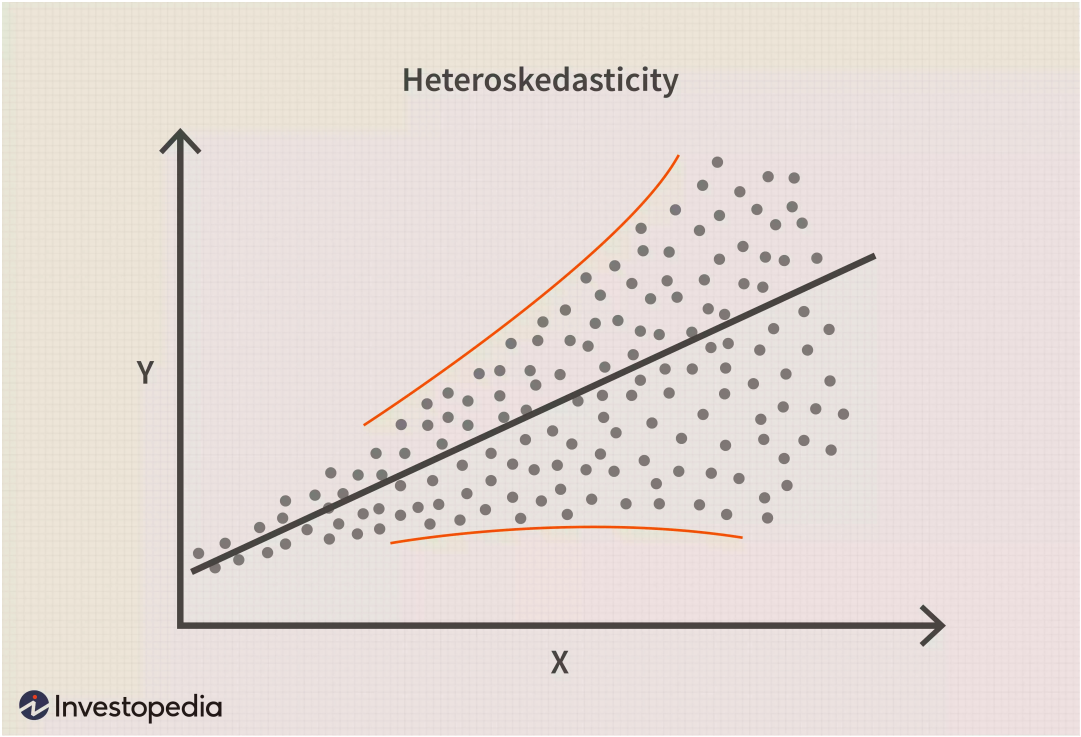
Source: https://alexrachnog.medium.com/deep-learning-the-final-frontier-for-signal-processing-and-time-series-analysis-734307167ad6

Autocorrelation

- Definition (Source: Investopedia, Wikipedia)
  - Autocorrelation represents the degree of similarity between a given time series and a lagged version of itself over successive time intervals.
  - the correlation of a signal with a delayed copy of itself as a function of delay
- In other words,
  - Yesterday's data might affect today's data
- This is a very special and powerful characteristic for many time series data
  - Autocorrelation based strategies tend to show competitive performances in finance domain

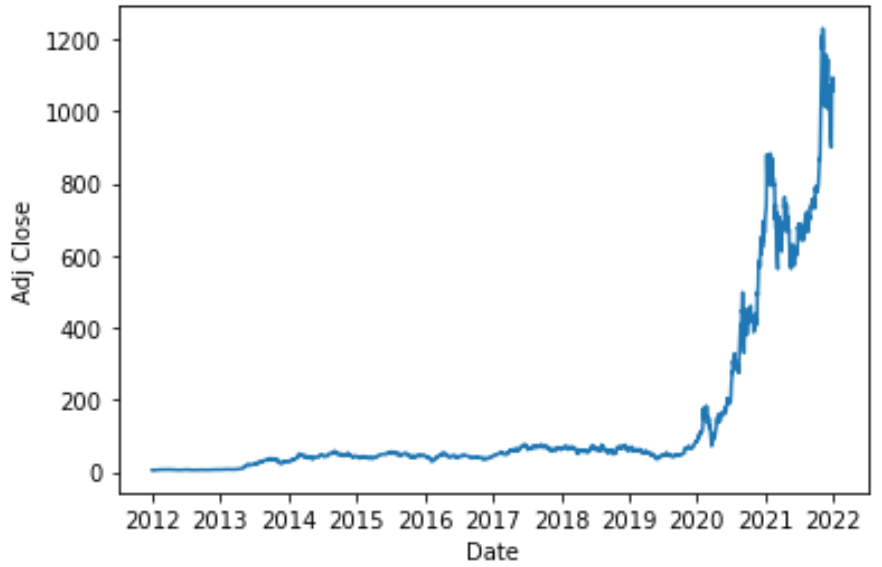
Heteroskedasticity

- Definition (Source: Investopedia, Wikipedia)
  - In statistics, heteroskedasticity (or heteroscedasticity) happens when the standard deviations of a predicted variable, monitored over different values of an independent variable or as related to prior time periods, are non-constant.



Heteroskedasticity

- Why should we consider this in time series data?
  - A time-series model can have heteroscedasticity if the dependent variable changes significantly from the beginning to the end of the series
  - What if we don't consider heteroskedasticity in our model to analyze Tesla's stock price
    - Cannot trust the result
- Key property for regression-based analysis



Stationarity

- A stationary time series is one whose properties do not depend on the time at which the series is observed.
  - If our data is stationary, it's easy to interpret
  - If not, we need to find a trend, seasonal, cyclical, irregular component of data
    - Or just feed it into deep learning models
- In finance, we frequently assume that returns are stationary
  - Similar to mean reverting
  - Many analysis are based on this assumption

Statistical tests for time series properties

- Autocorrelation: Durbin Watson test
  - Test with residuals from regression
- Heteroskedasticity: Breusch Pagan Test
  - Test variance of errors in regression
- Stationarity: Augmented Dickey-Fuller test
  - We simply call it ADF test
- Many statistical tests assume that error terms are normally distributed
  - To check normality: Jarque Bera test
- You can implement all of them in python using libraries like statsmodels

Traditional Time Series Models

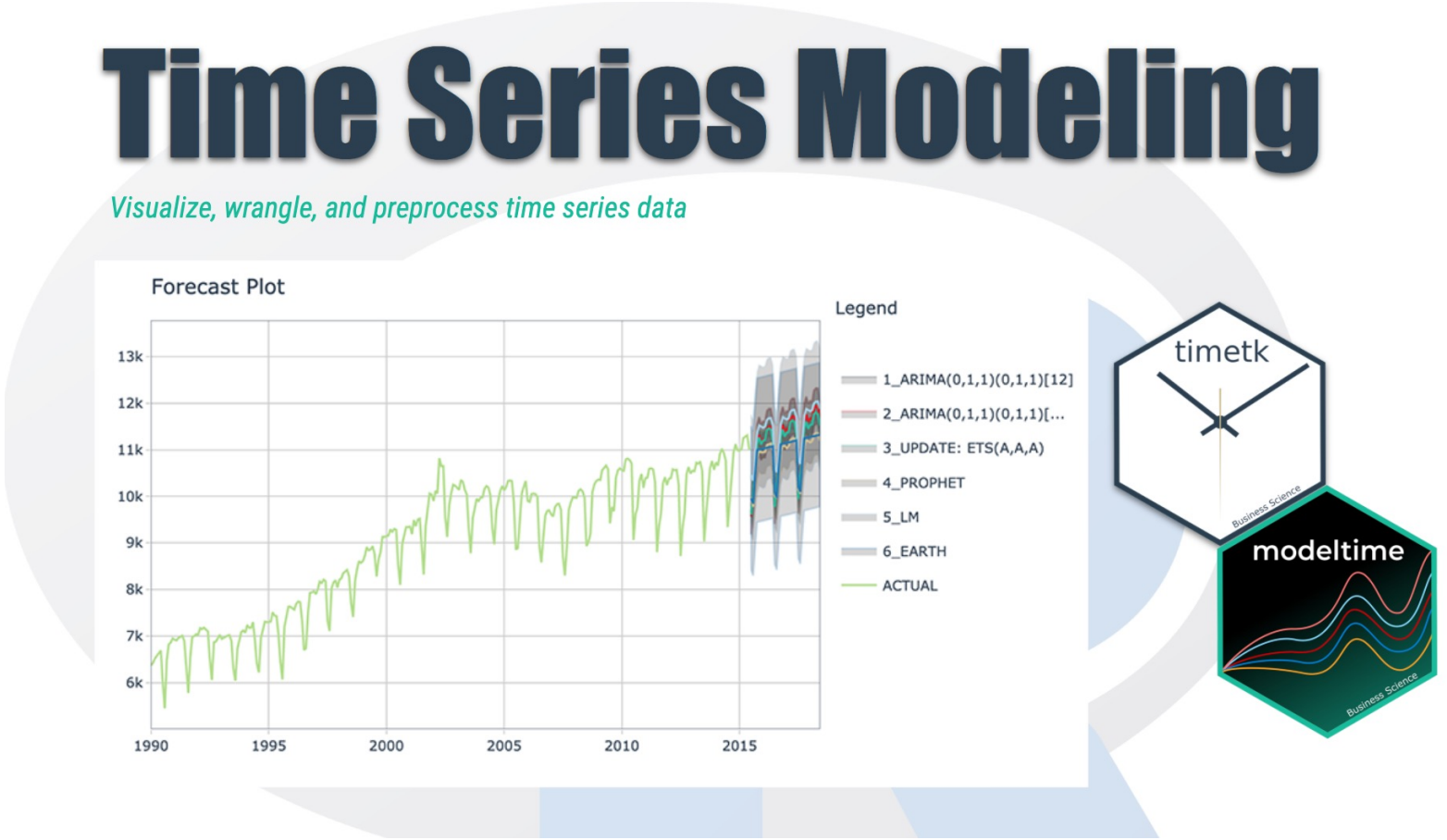


Image Source: https://www.business-science.io/code-tools/2020/09/09/five-minute-time-series-modeling-data.html

AR (AutoRegressive Model)

- Autoregressive models use previous observations to predict future values
- These models use regression technique and focus on autocorrelation in time series data
  - You should decide lag value, which can be defined as a fixed amount of passing time
  - For example, if you use lag of 3, you will use $data_{t-3}$ to predict $data_t$
  - HAR models are also widely used in finance to estimate volatility
    - H here stands for heterogeneous
    - You use daily, weekly, monthly value as independent variables in regression
- In python, use AutoReg from statsmodels.tsa.ar_model

MA (Moving Average Model)

- Similar to AR but focus on residuals of previous forecasts!
- Definition (Source: Wikipedia)
  - The moving-average model specifies that the output variable depends linearly on the current and various past values of a stochastic (imperfectly predictable) term.
  - Different from moving average

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t,$$

- In python, you may use ARMA from statsmodels.tsa.arima_model
  - You may notice that we use ARMA which is a combined version of AR, MA model
  - The integrated version, ARIMA has been a go-to model for many years

ARIMA (AutoRegressive Integrated Moving Average Model)

- Key takeaways (Source: Investopedia)
  - ARIMA models predict future values based on past values.
  - ARIMA makes use of lagged moving averages to smooth time series data.
  - They are widely used in technical analysis to forecast future security prices.
  - Autoregressive models implicitly assume that the future will resemble the past.
  - Therefore, they can prove inaccurate under certain market conditions, such as financial crises or periods of rapid technological change.
- Parameters
  - p: the number of lag observations in the model (lag order)
  - d: the number of times that the raw observations are differenced (degree of differencing)
  - q: the size of the moving average window (order of the moving average)

ARIMA (AutoRegressive Integrated Moving Average Model)

- Tip from TA
  - ARIMA captures trends very well and if you want to capture seasonal component of time series data, please consider using SARIMA
  - To decide which lag value to use in this model, I recommend you making use of ADF test
  - From previous slides, you learned that ADF test is used for testing the stationarity of time series data
  - So, you can choose lag value which doesn't reject the null hypothesis of ADF test!
- In python,
  - ADF test: from statstmodels.tsa.stattools import adfuller
  - ARIMA: from statsmodels.tsa.arima_model import ARIMA

ARCH (AutoRegressive Conditional Heteroskedasticity Model)

- What is ARCH? (Source: Investopedia)
  - Autoregressive conditional heteroskedasticity (ARCH) is a statistical model used to analyze volatility in time series in order to forecast future volatility
  - In the financial world, ARCH modeling is used to estimate risk by providing a model of volatility that more closely resembles real markets.
  - ARCH modeling shows that periods of high volatility are followed by more high volatility and periods of low volatility are followed by more low volatility
  - ARCH should be applied to time series data that does not show trends or seasonal components

GARCH (Generalized AutoRegressive Conditional Heteroskedasticity Model)

- ## What is GARCH?
  - GARCH is generalized version of ARCH
  - GARCH incorporates a moving average component together with the autogregressive component
  - This allows us to model the conditional change in variance over time and capture changes in the time-dependent variance
  - So, you can say that GARCH is the "ARMA equivalent" of ARCH, which only has an autoregressive component.
- ## In python, you can pip install arch package and use arch_model
  - For this part of the lecture, I refer to the blog post below. Feel free to check out!
    - https://medium.com/@ranjithkumar.rocking/time-series-model-s-arch-and-garch-2781a982b448

ML based Time Series Models

Recurrent Neural Networks

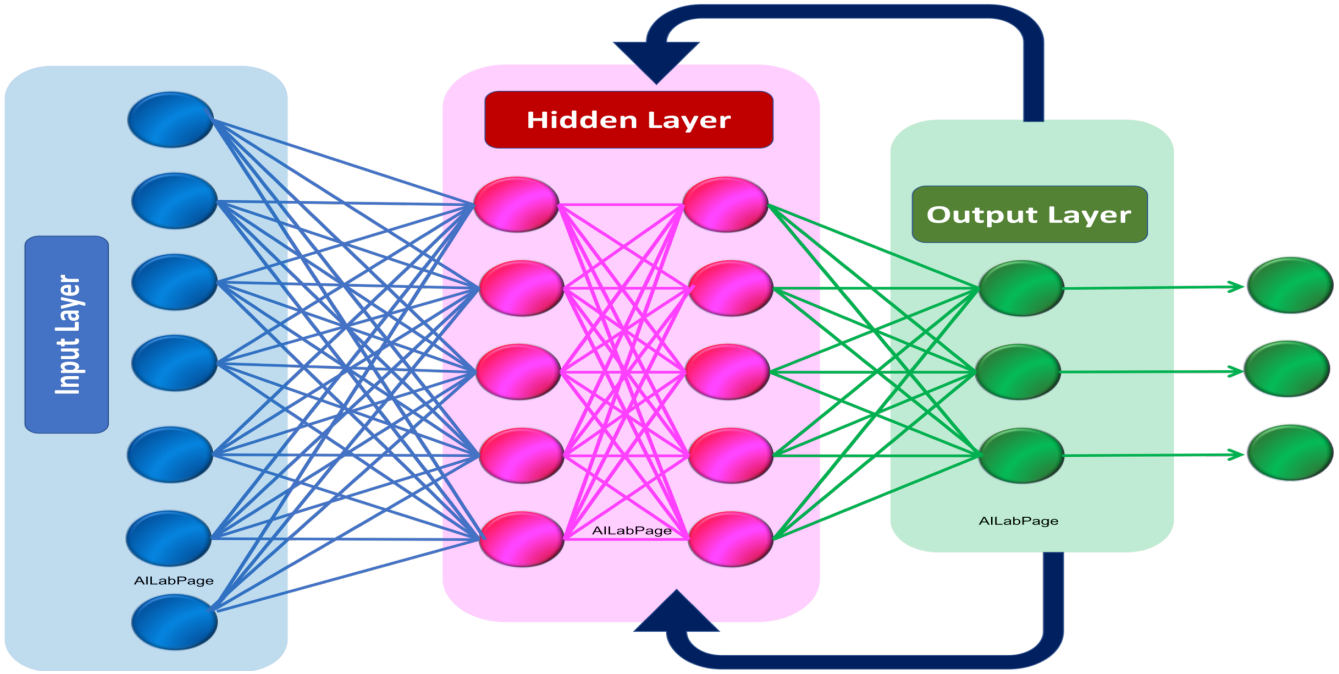
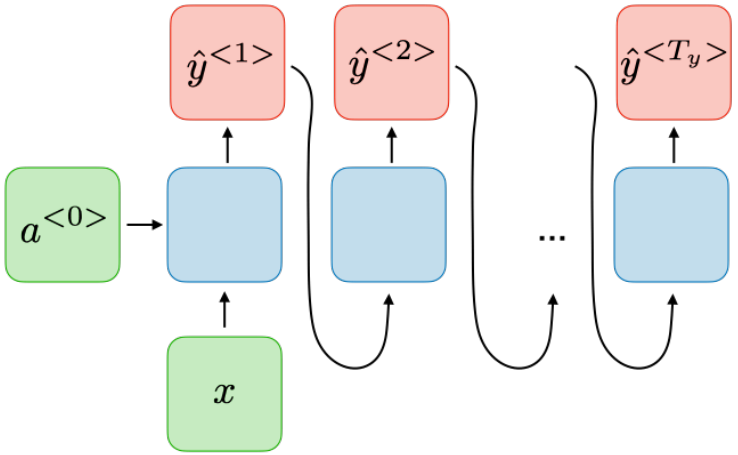


Image Source: https://morioh.com/p/1bc305d7dbdf

RNN (Recurrent Neural Network)

- Definition
  - Recurrent neural networks, also known as RNNs, are a class of neural networks that allow previous outputs to be used as inputs while having hidden states.
- What makes it special?
  - RNNs can use their internal state (memory) to process variable length sequences of inputs



Reference: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

RNN in PyTorch

- Simply use RNN module in PyTorch
  - Same for LSTM, GRU, etc…

RNN

CLASS `torch.nn.RNN(*args, **kwargs)` [SOURCE]

Applies a multi-layer Elman RNN with $\tanh$ or ReLU non-linearity to an input sequence.

For each element in the input sequence, each layer computes the following function:

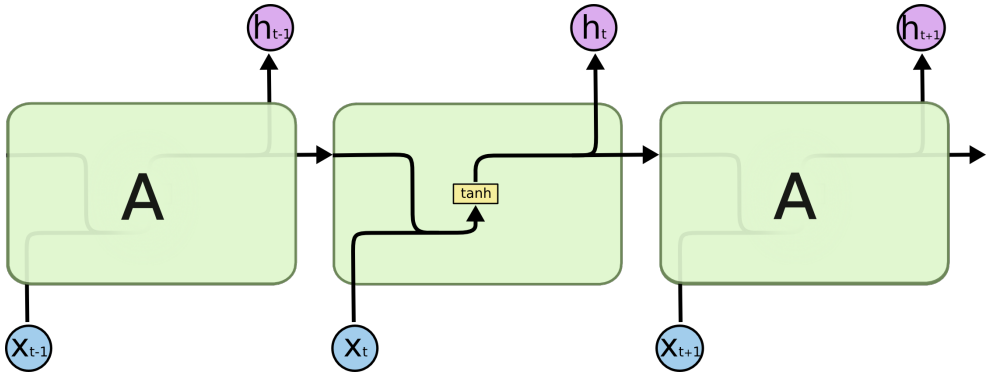
$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh})$$

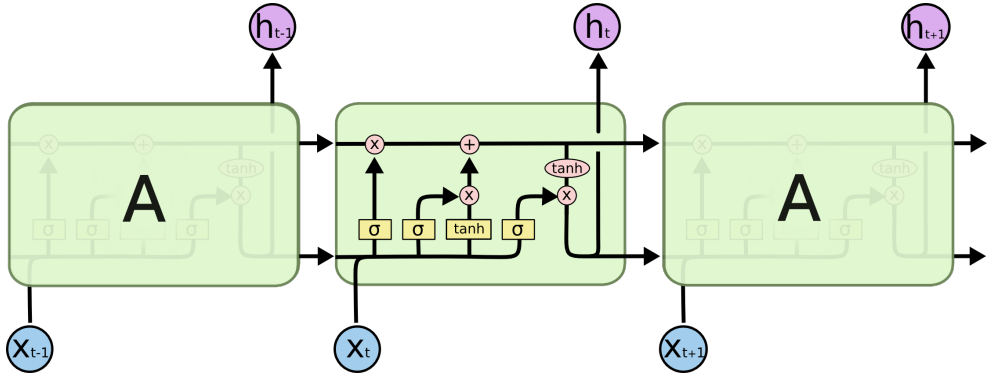
where h_t is the hidden state at time *t*, x_t is the input at time *t*, and $h_{(t-1)}$ is the hidden state of the previous layer at time *t-1* or the initial hidden state at time *0*. If `nonlinearity` is `'relu'`, then ReLU is used instead of $\tanh$.

LSTM (Long Short Term Memory)

- Vanishing gradient problem
  - In vanilla RNN, gradient vanishes during back propagation
  - This might prevent weight from updating its value
  - Sometimes, early stopping might occur
- LSTM (Long Short Term Memory)
  - Introduced in 1997
  - LSTMs are explicitly designed to avoid the long-term dependency problem (vanishing gradient problem).
  - Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

LSTM (Long Short Term Memory)



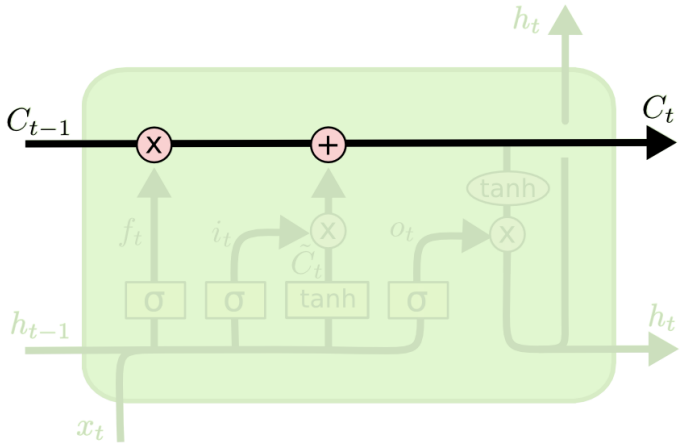


- RNN vs. LSTM
  - The repeating module in a standard RNN contains a single layer.
  - The repeating module in an LSTM contains four interacting layers.

Reference: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM (Long Short Term Memory)

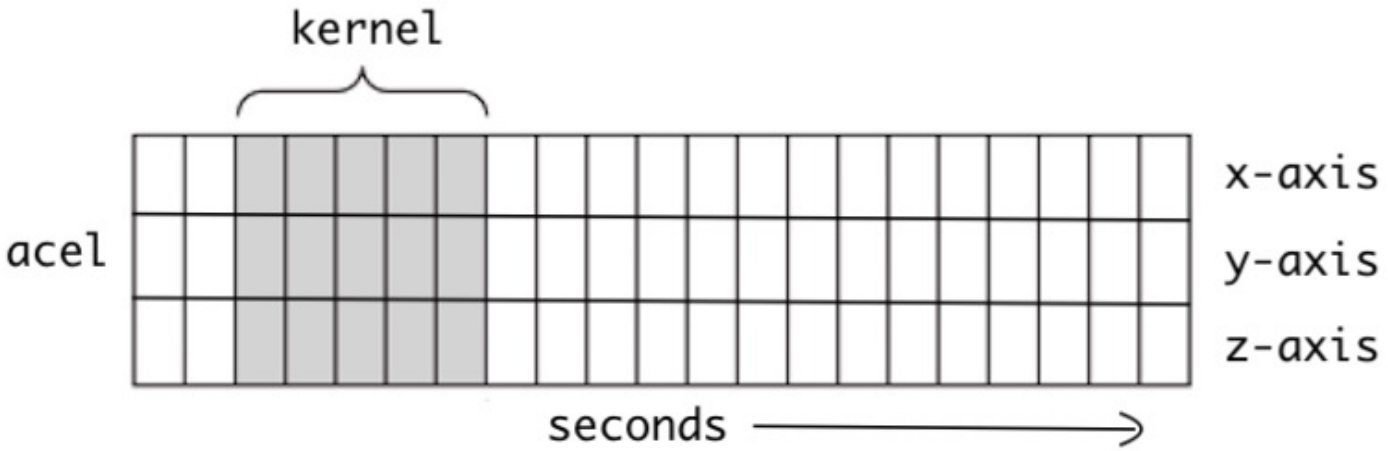
- Core idea behind LSTMs
  - The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.
  - The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.



Reference: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

1D CNN

- 1D Convolution
  - Widely used in time series analysis
  - Go over entire length with same filter
  - Able to extract some minor properties inside our data
  - Sometimes performs much better than RNN-based models



Deep Learning in Finance

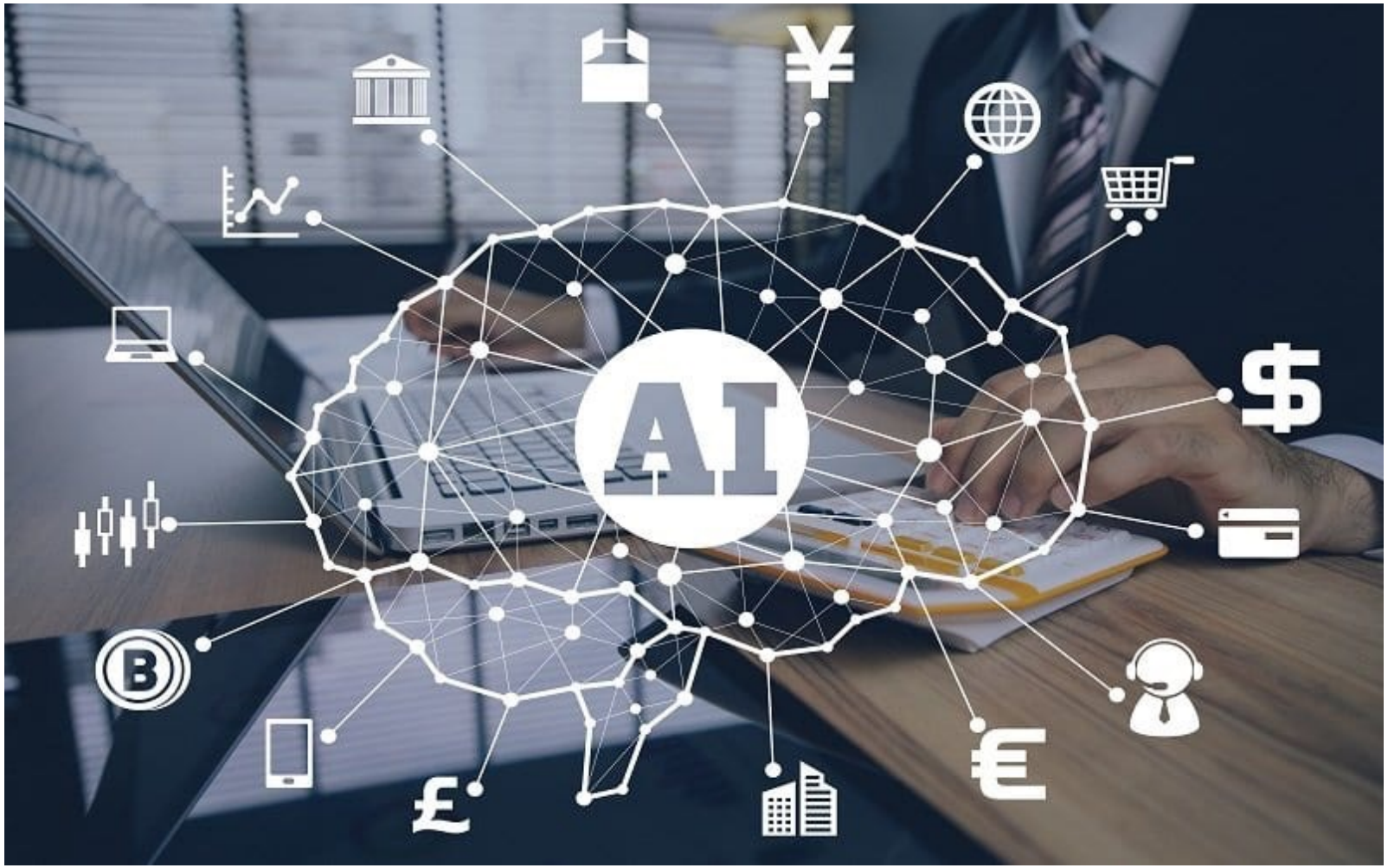


Image Source: https://riskgroupllc.com/how-will-deep-learning-disrupt-financial-sector/

For this part of lecture, I refer to this blog post: https://research.aimultiple.com/deep-learning-in-finance/

Why is deep learning relevant in finance?

- Finance deals with both structured and unstructured data such as documents and text. Deep learning allows financial firms
  - to convert unstructured data into structured, machine-readable data. For example, this allows banks to get financial information on companies from their annual reports published in regulatory platforms like the Companies House in the UK
  - to make predictions & classifications on structured data. Data such as stock market information is highly structured and can be used to automate trading activities

Deep Learning Use Cases

- Customer Service
  - Financial services companies use finance-specific chatbots with deep learning models to improve user experience. Deep learning-based solutions bring personalized services to customers. According to customer's financial activities, virtual assistants can
    - automate frequently completed actions
    - suggest products that are not used by the customers but can be a good fit for them
    - answer questions
  - Deep learning algorithms can identify potential churn by analyzing interactions. This capability helps insurance companies and banks to offer discounts and new plans and protect their customer base.

Deep Learning Use Cases

- Financial Security and Compliance
  - Deep learning algorithms are effective for
    - revealing suspicious transactions with high precision in real time
    - Using unstructured data (e.g. satellite and street view images) to check the existence of a business or to perform other compliance controls
  - This provides advantages such as
    - reduction of operational cost
    - improvement of regulatory compliance

Deep Learning Use Cases

Insurance Underwriting

- Insurance companies use historical consumer data to train deep learning algorithms. This consumer data includes health records, information gathered from wearable devices, potential health issues, age, income, profession, loan payment history, etc. Deep learning based solutions help sector to
  - predict and reduce risks
  - set suitable premiums
  - improve speed and accuracy of underwriting processes.

Insurance Claims

- Thanks to computer vision and document processing capabilities, deep learning models allow insurance companies to asses damages for car accident claims and risks for home insurance. Also, these models can identify fraudulent claims more accurately.

Deep Learning Use Cases

- Lending
  - Deep learning models use learned patterns and results of document processing to assess credit risks and loan requests. This data covers income, occupation, age, current financial assets, current credit scores, overdrafts, outstanding balance, foreclosures, loan payments. Then, they can make a decision about the qualification of the client for lending.
- Algorithmic Trading
  - By analyzing historical data & current price movements and extraction information from the news simultaneously, deep learning algorithms can predict stock values more accurately. These predictions are used for fast trading decisions. Due to lack of emotions, predictions and decisions deep learning models deliver are more neutral/objective and data- driven.

Challenges

- Are deep learning-based solutions perfect?
- Are they explainable?
- Are there any privacy issues?
- Who will be in charge when there's a problem?
- etc…

Time to code!

- For this session, you will practice 2 machine learning (deep learning) based approaches using financial data
  - Stock price prediction
  - Bankruptcy prediction
- There are lots of projects on GitHub so please google them and get some ideas for your projects 😄