Time Series Forecasting using ML

BITS Pilani-Hyderabad

Dr. Tanujit Chakraborty

Sorbonne University, Abu Dhabi, UAE

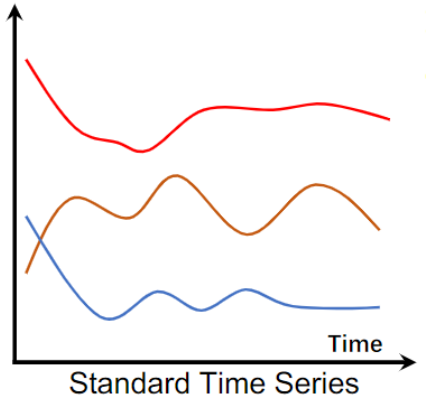
Sorbonne Centre for Artificial Intelligence, Paris, France

Research Areas: Time Series Forecasting, Machine Learning, Econometrics, Health Data Science

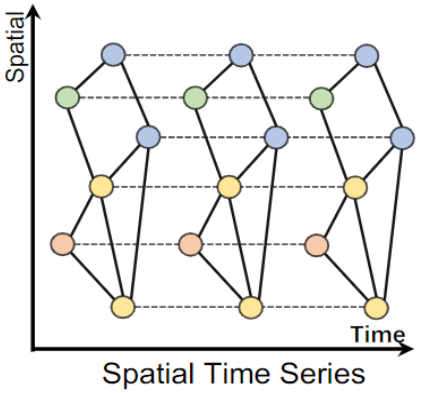
January 29, 2026

Space, Time, and Forecasting

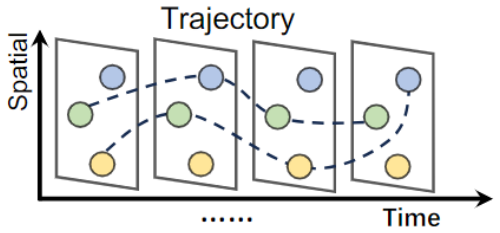
Time series is a set of observations, each one being recorded at a specific time, e.g., Annual GDP of UAE, iPhone Sales figure, etc.



A spatial time series is a type of data that combines spatial (cross-sectional) and temporal dimensions, capturing how measurements or observations vary across different locations over time, e.g., Air quality of Abu Dhabi.



Forecasting is estimating how the sequence of observations will continue into the future.



Time Series is omnipresent

- Time series data is a specialized form of data that plays a vital role in various fields, including economics, finance, climate science, healthcare, and many others.

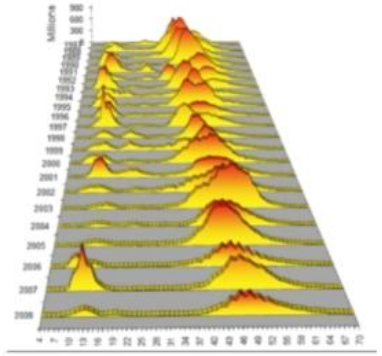
Shape of Time Series Data

- We usually think that the data is one-dimensional.
- It only consists of the time and the data associated with it (Temp.).

	New York City
1990-01-01	1
1990-01-02	5
1990-01-03	9
1990-01-04	13

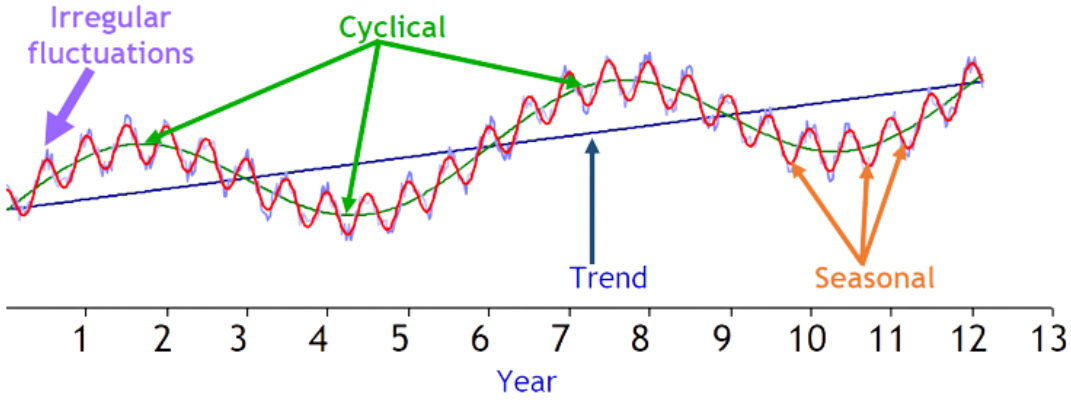
- But it can be Multidimensional.

	New York City	London	Tokyo	Paris
1990-01-01	1	2	3	4
1990-01-02	5	6	7	8
1990-01-03	9	10	11	12
1990-01-04	13	14	15	16



Time Series Components

- Trend (T_t): long-term increase or decrease in the time series.
- Seasonal (S_t): regular, repeating patterns at fixed intervals.
- Cyclic (C_t): Long-term fluctuation with no fixed periodicity.
- Irregular (I_t): Random, unpredictable fluctuations with no discernible pattern.



- Decomposition : $Y_t = f(T_t; S_t; C_t; I_t)$, where Y_t is data at period t.
- Additive decomposition: $Y_t = T_t + S_t + C_t + I_t$
- Multiplicative decomposition: $Y_t = T_t * S_t * C_t * I_t$

SORBONNE UNIVERSITÉ

Forecasting

What people think I forecast?

SORBONNE UNIVERSITÉ

When I go to any university, and I tell people that my job is time series forecasting and machine learning, usually one of two things happens:

- ...like, weather forecasting?
  - Lots of domain knowledge and specialized models exist
  - We leave it to meteorologists
- ...so, can you predict the stock market and we all get rich?
  - I'll tell you how, and we're all going to be rich!
  - Try it on your own risk!

What I forecast?

- Epidemic time series (e.g., dengue, malaria, hepatitis, etc.)
- Sales forecasting in the supply chain, retail at pharmacy companies
- Forecasting in climate
  - Air quality
  - El Nino
  - Seismic events
- Key macroeconomic variables (inflation, unemployment, exchange rate, etc.)
- ...

Can these be forecasted?

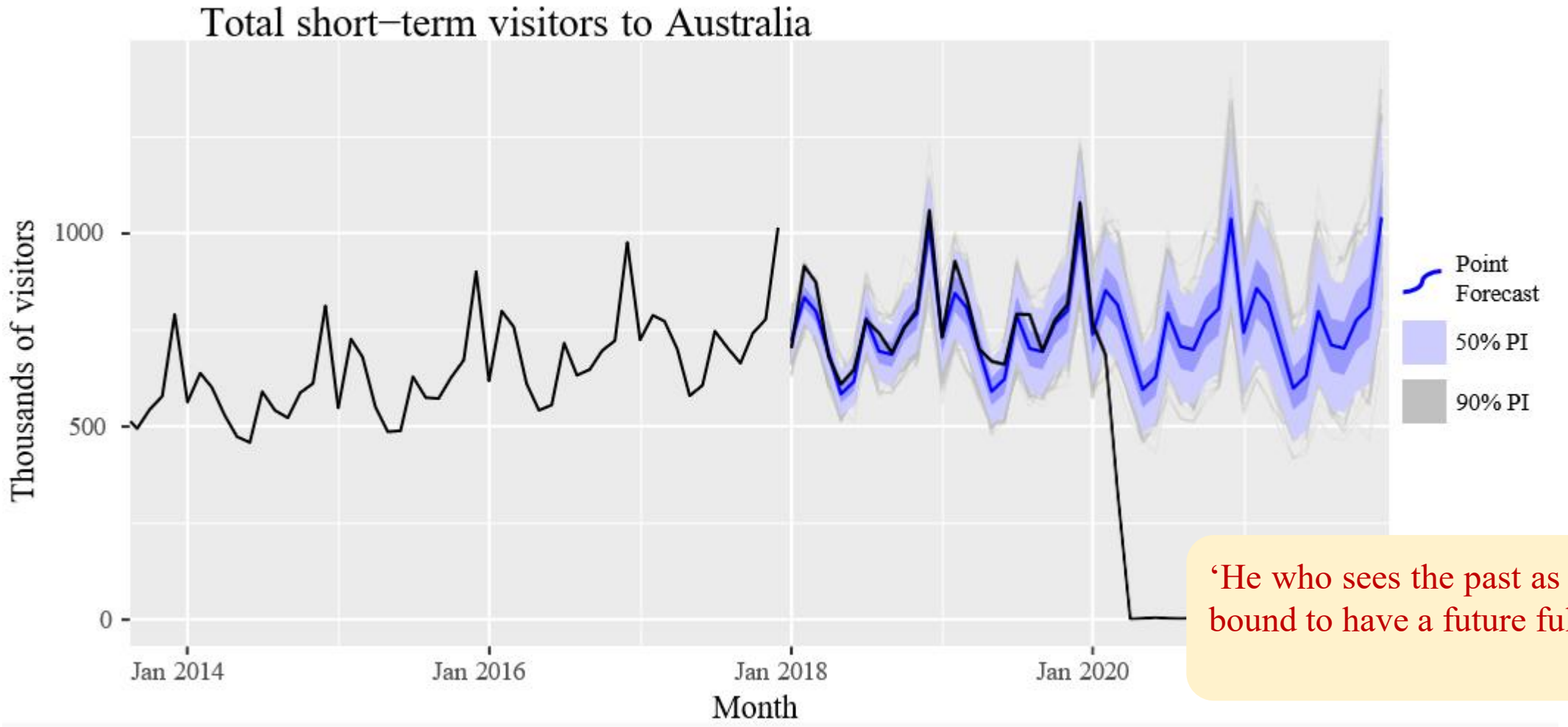
1. daily electricity demand in 3 days' time
2. Google stock price tomorrow
3. Google stock price in 6 months' time
4. maximum temperature tomorrow
5. total sales of drugs in pharmacies next month

Something is easy to forecast if:

1. we have a good understanding of the factors that contribute to it
2. there is a lot of data available
3. the future is somewhat similar to the past
    - ID assumption: samples are identically distributed
4. the forecasts cannot affect the thing we are trying to forecast.
    - self-fulfilling prophecies (election polls)
    - controlled systems
    - Big bull effect in stock markets / bitcoin prices

Uncertainty and Forecasting

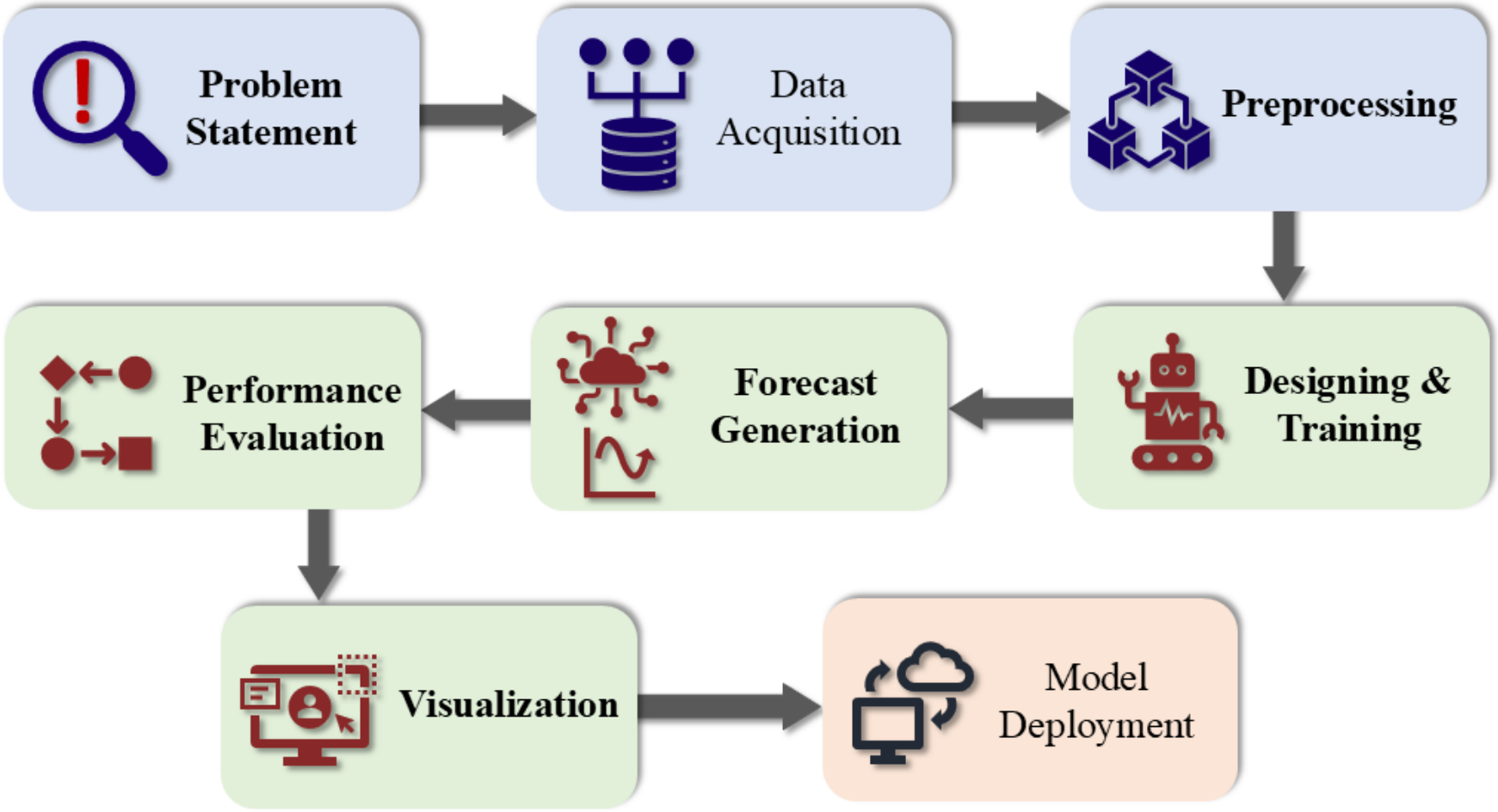
Forecasting is estimating how the sequence of observations will continue into the future.



'He who sees the past as surprise-free is bound to have a future full of surprise.'

- Amos Tversky

Process of Forecasting



Forecasting workflow with main strata: data (blue rectangles), analytics (green rectangles), and deployment (orange rectangles)

Time Series is omnipresent

SORBONNE UNIVERSITÉ

- Time series data is a specialized form of data that plays a vital role in various fields, including economics, finance, climate science, healthcare, and many others.

- A forecast is a scientifically justified assertion about the possible states of an object in the future

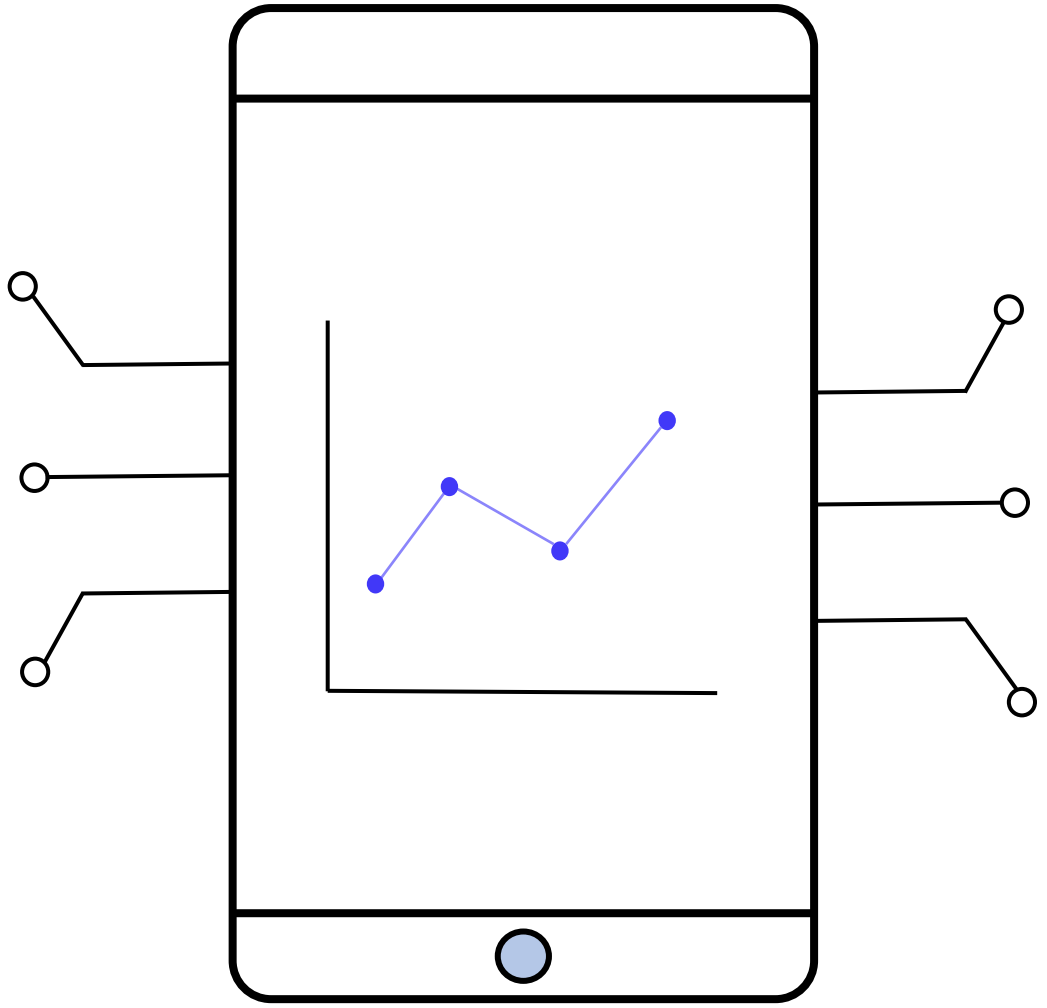
Time Series is everywhere

- Epidemiology: Epidemic/Flu/Covid-19 cases observed over some time period.
- Economics: Stock prices, unemployment rates, inflation rates, etc.
- Earth and Environmental Sciences: Daily Sea Surface Temperature, Southern Oscillation Index, Seismic Waves, Air Quality Index, Global Warming, etc.
- Astronomy: Sunspot numbers, Luminosity of stars, etc.
- Demography: Population series, Birthrates, Mortality rates, etc.
- Medical Science: Blood pressure, Blood oxygen level, Sugar level, etc.
- Business: Product demand, Sales, Market share, etc.

What makes for a good forecasting model?

SORBONNE UNIVERSITÉ

- Forecast accuracy
- Resource requirements
- Speed



- Data leakage
- Understandability & explainability
- Maintainability & debuggability

Forecast Performance Evaluation

Performance metrics such as mean absolute error (MAE), root mean square error (RMSE), and mean absolute percent error (MAPE) are used to evaluate the performances of different forecasting models for the unemployment rate data sets:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2};$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|;$$

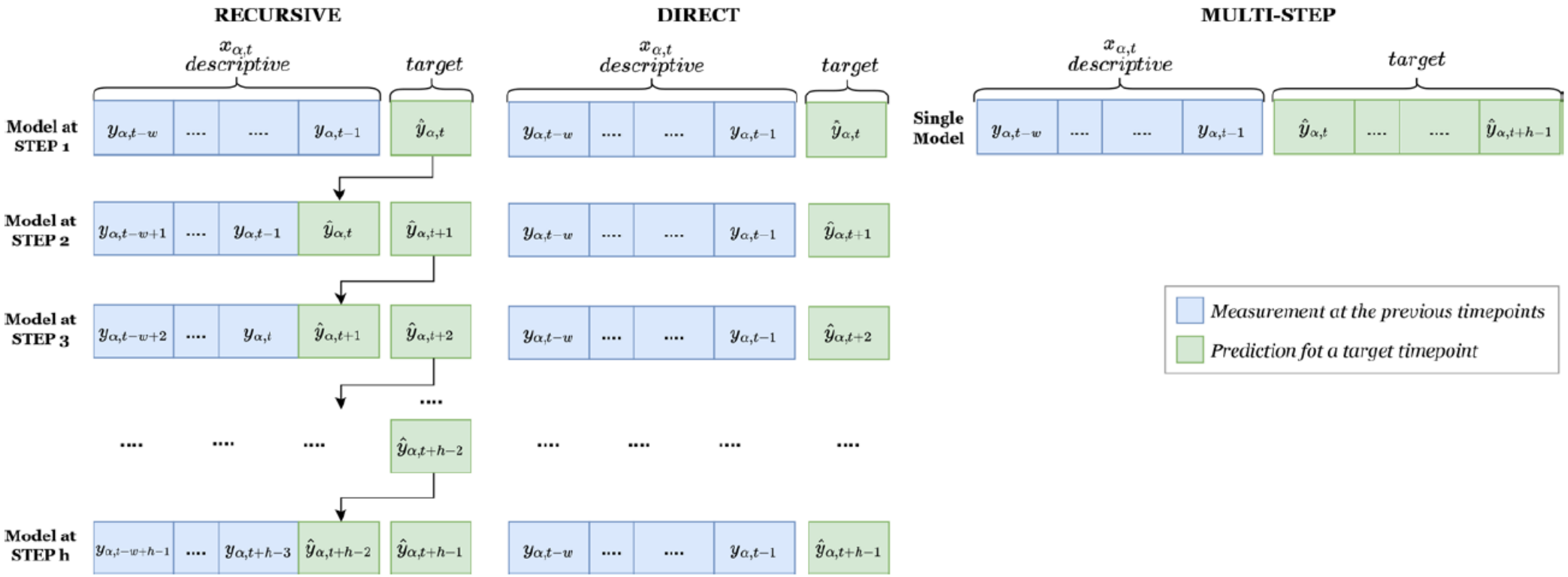
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|,$$

Where y_i is the actual output, $\hat{y}_i$ is the predicted output, and n denotes the number of data points.

By definition, the lower the value of these performance metrics, the better is the performance of the concerned forecasting model.

Types of Forecasting Tasks

1. **One-Step Forecasting** **2. Multi-Step Forecasting** (Incremental Multi-step and Multi-output Multi step)



SORBONNE UNIVERSITÉ

Classical Forecasting Methods

Average Method

- Here, the forecasts of all future values are equal to the average (or "mean") of the historical data. If we let the historical data be denoted by $y_1, \ldots, y_T$, then we can write the forecasts as

$$\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T.$$

- The notation $\hat{y}_{T+h|T}$ is a short-hand for the estimate of y_{T+h} based on the data $y_1, \ldots, y_T$.

Naïve & Seasonal Naïve Method

- Naïve Method: All forecasts are set to be the value of the last observation. That is,

$$\hat{y}_{T+h|T} = y_T.$$

- This method works remarkably well for many economic and financial time series.
- Seasonal Naïve Method: For or highly seasonal data, forecasts are set equal to the most recent observation from the same season (e.g., the same month in the previous year):

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)},$$

where $m =$ the seasonal period, and k is the integer part of $(h-1)/m$ (i.e., the number of complete years in the forecast period prior to time $T+h$).

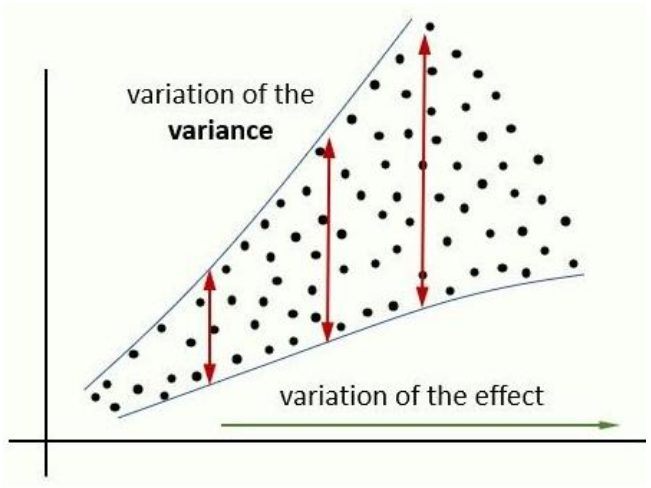
SORBONNE UNIVERSITÉ

Auto-correlation Analysis

Auto Regression Analysis

- Regression analysis for time-ordered data is known as Auto-Regression Analysis
- Time series data are data collected on the same observational unit at multiple time period.

Example: Indian rate of price inflation

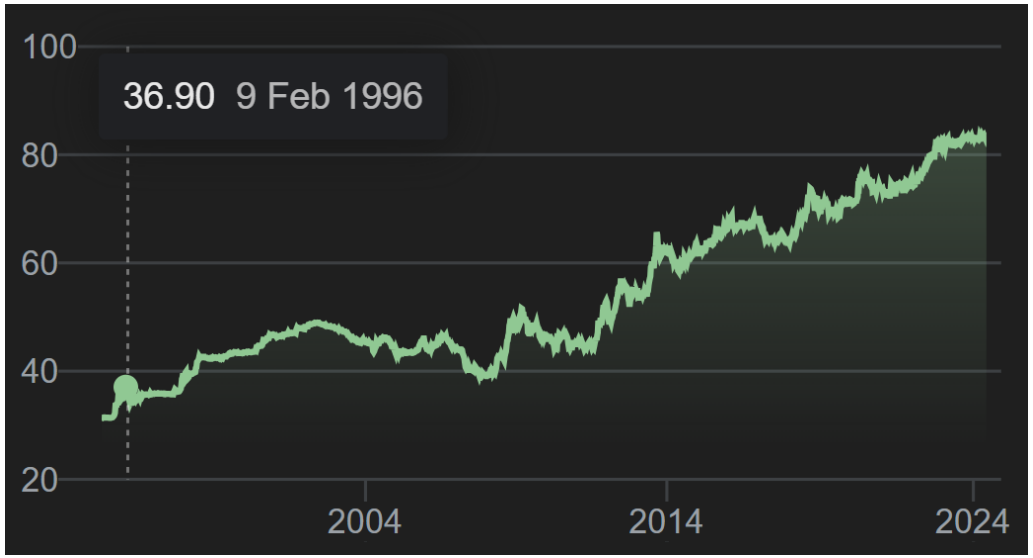


Heteroscedasticity

Modeling with Time Series Data

SORBONNE UNIVERSITÉ

- Correlation over time
  - Serial correlation, also called autocorrelation
  - Calculating standard error
- To estimate dynamic causal effects
  - Under which dynamic effects can be estimated?
  - How to estimate?
- Forecasting model
  - Forecasting model build on regression model



Can we predict the trend (USD vs. INR) at a time, say 2025?

Auto-correlation coefficient

Just as correlation measures the extent of a linear relationship between two variables, autocorrelation measures the linear relationship between lagged values of a time series.

- There are several autocorrelation coefficients, corresponding to each panel in the lag plot.
- For example, let $[y_1, y_2, \ldots, y_{T-1}, y_T]$ denote T observations on the time series random variable Y_t, then r_1 measures the relationship between y_t and y_{t-1}, r_2 measures the relationship between y_t and y_{t-2}, and so on.

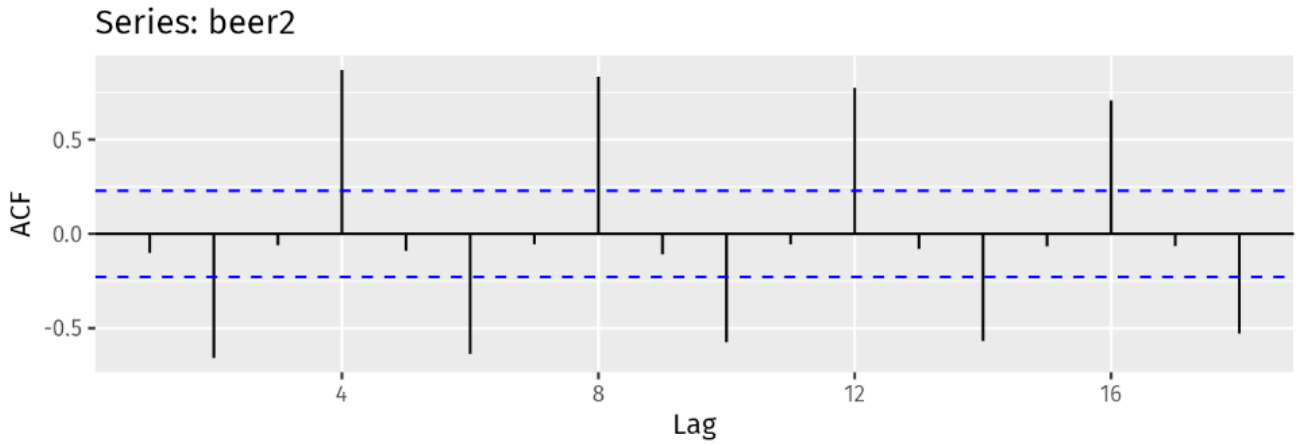
The value of r_k can be written as

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2}$$

where T is the length of the time series. The autocorrelation coefficients make up the autocorrelation function or ACF.

ACF

The autocorrelation coefficients are plotted to show the autocorrelation function or ACF of quarterly beer production in Australia. The plot is also known as a correlogram.



In this graph:

- r_4 is higher than for the other lags. This is due to the seasonal pattern in the data: the peaks tend to be four quarters apart and the troughs tend to be four quarters apart.
- r_2 is more negative than for the other lags because troughs tend to be two quarters behind peaks.
- The dashed blue lines indicate whether the correlations are significantly different from zero.

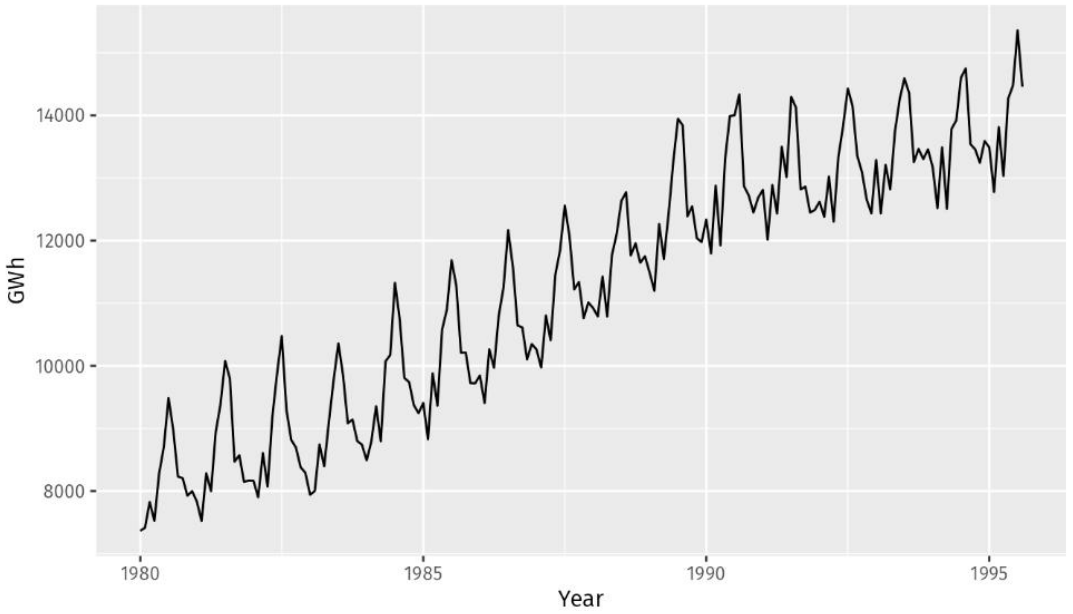
Example: Autocorrelation

- When data have a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. So the ACF of trended time series tend to have positive values that slowly decrease as the lags increase.

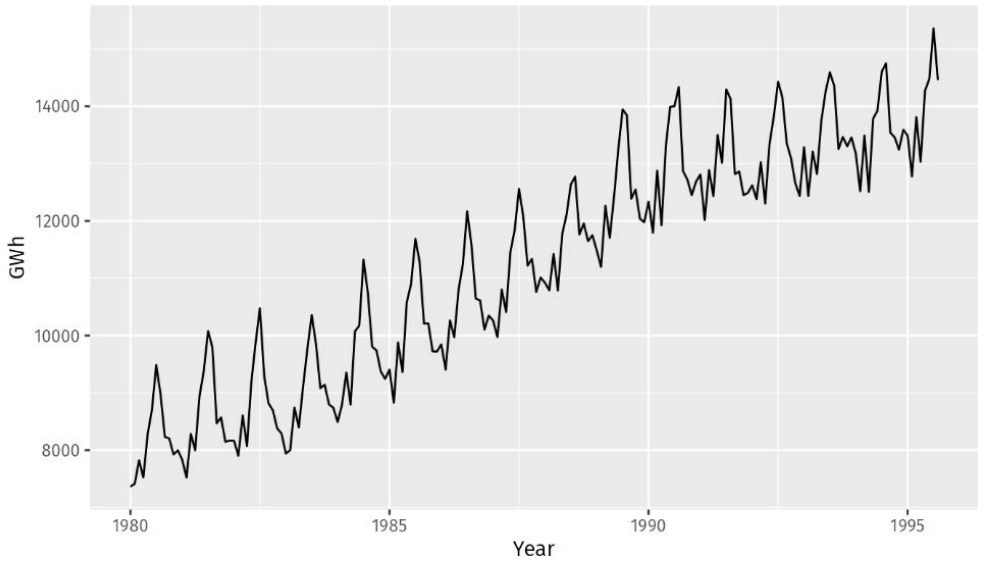
- When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal frequency) than for other lags.

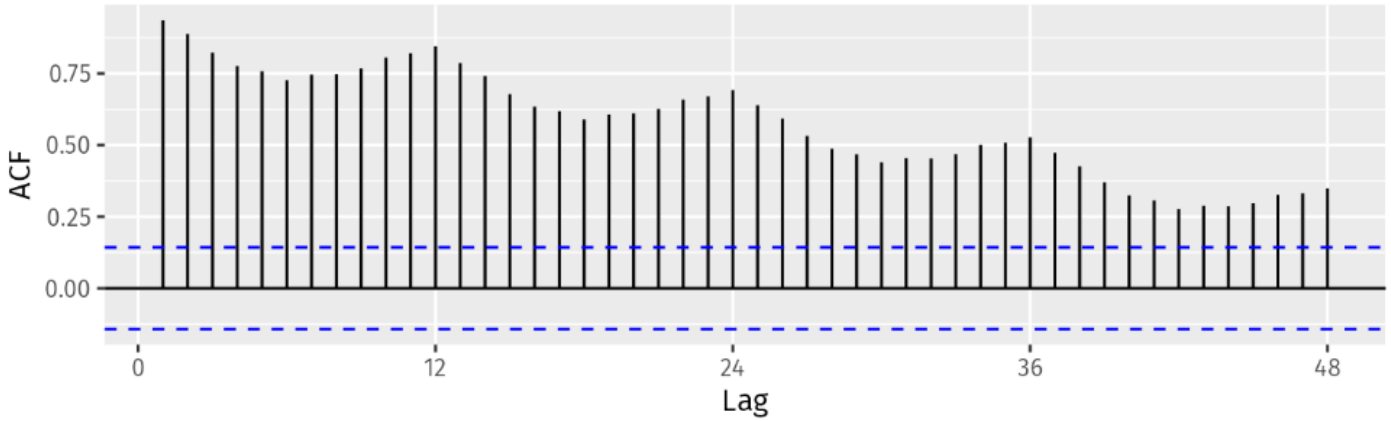
- When data are both trended and seasonal, you see a combination of these effects.

- The monthly Australian electricity demand series plotted in Figure shows both trend and seasonality.



Example: Autocorrelation





The slow decrease in the ACF as the lags increase is due to the trend, while the "scalloped" shape is due to the seasonality.

SORBONNE UNIVERSITÉ

Auto-Regression Model

Auto-Regression Model

SORBONNE UNIVERSITÉ

An autoregressive model (also called AR model) is used to model a future behavior for a time-ordered data, using data from past behaviors.

- Essentially, it is a linear regression analysis of a dependent variable using one or more variables(s) in a given time-series data: $Y_t = f(Y_{t-1}, Y_{t-2}, \dots, Y_{t-p})$
- A natural starting point for forecasting model is to use past values of Y, that is, Y_{t-1}, Y_{t-2}, ... to predict Y_t.
- An auto-regression is a regression model in which Y_t is regressed against its own lagged values.
- The number of lags used as regressors is called the **order of auto-regression.**
  - In first order auto-regression (denoted as AR(1)), Y_t is regressed against Y_{t-1}
  - In p^{th} order auto-regression (denoted as AR(p)), Y_t is regressed against, $Y_{t-1}, Y_{t-2}, \dots, Y_{t-p}$

Auto-Regression Model for Forecasting

SORBONNE UNIVERSITÉ

An **AutoRegressive model of order 1** (AR(1)) says: Today's value = ϕ×Yesterday's value + shock

More formally:

$$Y_t = \phi Y_{t-1} + \epsilon_t$$

Where:

- Y_t: value of the variable at time t (e.g., stock return, inflation, sales).
- ϕ: a parameter that tells us how much of yesterday's value spills into today.
- ϵ_t: a random shock (news, surprises, etc.) $\epsilon_t \sim \text{iid } N(0, \sigma^2)$ is white noise.

Example:

Suppose a stock's daily return depends slightly on the previous day:

$$R_t = 0.4 R_{t-1} + \epsilon_t$$

If yesterday's return was **2%**, then **today's expected return** is:

$$\mathbb{E}[R_t] = 0.4 \times 2\% = 0.8\%$$

p^{th} order Auto-regression Model

Formula: p^{th} Order Auto-regression Model

In general, the p^{th} order auto-regression model is defined as

$$Y_t = \beta_0 + \sum_{i=1}^{p} \beta_i Y_{t-i} + \varepsilon_t$$

where, $\beta_0, \beta_1, \ldots, \beta_p$ is called auto-regression coefficients and ε_t is the noise term or residue and in practice it is assumed to Gaussian white noise

Computing AR Coefficients

Method 1: Ordinary Least Squares (OLS)

$$Y_t = \phi Y_{t-1} + \epsilon_t, \quad t = 2, \ldots, T$$

Step-by-step derivation

1. Define the sum of squared residuals (SSR):

$$S(\phi) = \sum_{t=2}^{T} (Y_t - \phi Y_{t-1})^2$$

2. Differentiate $S(\phi)$ with respect to ϕ:

$$\frac{dS(\phi)}{d\phi} = \sum_{t=2}^{T} 2(Y_t - \phi Y_{t-1})(-Y_{t-1}) = -2 \sum_{t=2}^{T} Y_{t-1}(Y_t - \phi Y_{t-1})$$

Computing AR Coefficients

Method 1: Ordinary Least Squares (OLS)

3. Set the derivative to zero for minimization:

$$\sum_{t=2}^{T} Y_{t-1}(Y_t - \phi Y_{t-1}) = 0$$

4. Solve for ϕ:

$$\sum_{t=2}^{T} Y_{t-1}Y_t = \phi \sum_{t=2}^{T} Y_{t-1}^2 \quad \Rightarrow \quad \hat{\phi} = \frac{\sum_{t=2}^{T} Y_{t-1}Y_t}{\sum_{t=2}^{T} Y_{t-1}^2}$$

5. Estimate residual variance:

$$\hat{\sigma}^2 = \frac{1}{T-1} \sum_{t=2}^{T} (Y_t - \hat{\phi} Y_{t-1})^2$$

AR Model by Hand

Here is a small dataset of 5 time points used for manually fitting the AR(1) model.

	Time	Y
1	1	1.2
2	2	1.8
3	3	2.4
4	4	2.1
5	5	1.9

AR Model by Hand

SORBONNE UNIVERSITÉ

1. **"Step-by-step AR(1) Estimation (Small Data)"** – This table shows:
   - Y_{t-1} (lagged values)
   - Y_t (current values)
   - Predicted Y_t using $\hat{\phi} Y_{t-1}$
   - Residuals $= Y_t - \hat{\phi} Y_{t-1}$
2. **"Estimated Parameters (Small Data)"** – This table displays:
   - Estimated $\hat{\phi} \approx 1.044$
   - Estimated $\hat{\sigma} \approx 0.453$

These values are derived manually (without libraries) using the formulas:

- $\hat{\phi} = \frac{\sum Y_{t-1} Y_t}{\sum Y_{t-1}^2}$
- $\hat{\sigma}^2 = \frac{1}{T-1} \sum (Y_t - \hat{\phi} Y_{t-1})^2$

Y_{t-1}	Y_t	Predicted_	Residual
1.2	1.8	1.25333	0.54667
1.8	2.4	1.88	0.52
2.4	2.1	2.50667	-0.40667
2.1	1.9	2.19333	-0.29333

Applications in Business and Finance

SORBONNE UNIVERSITÉ

In Finance:

- **Interest rates** tend to follow an AR(1) pattern — central banks move gradually.
- **Volatility forecasting** uses AR-type models in GARCH models.
- **Stock returns** may have slight autocorrelation due to investor herding.

In Business:

- **Monthly sales**, if not seasonally adjusted, often follow AR(1)-like behavior.
- **Customer demand** today may depend on yesterday's trend.

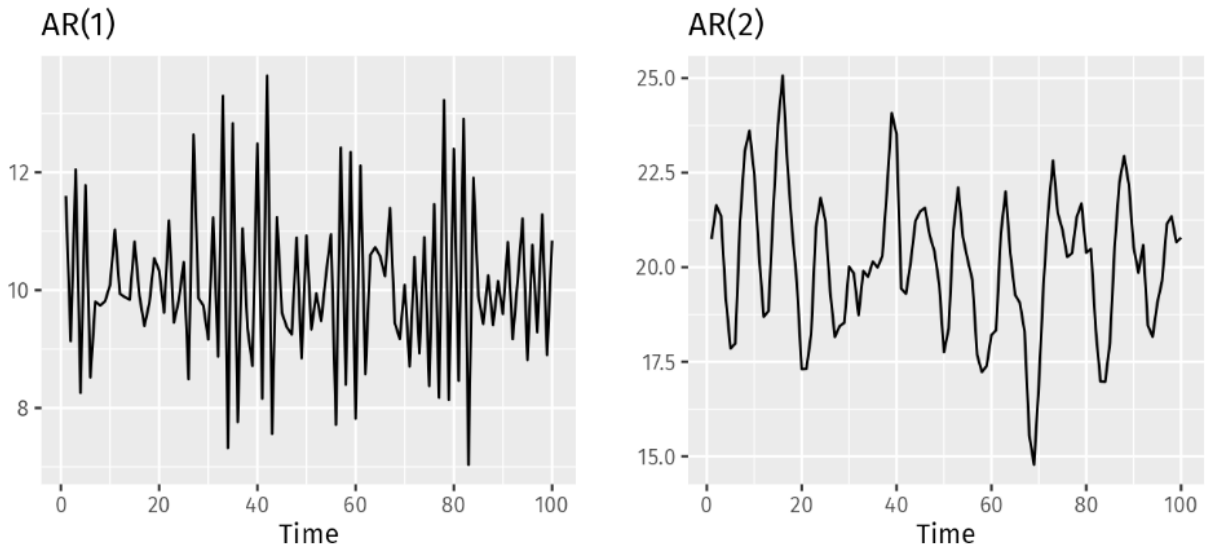


Figure 8.5: Two examples of data from autoregressive models with different parameters. Left: AR(1) with $y_t = 18 - 0.8y_{t-1} + \varepsilon_t$. Right: AR(2) with $y_t = 8 + 1.3y_{t-1} - 0.7y_{t-2} + \varepsilon_t$. In both cases, ε_t is normally distributed white noise with mean zero and variance one.

Applications in Business and Finance

SORBONNE UNIVERSITÉ

Application	What AR Model Helps With
Inventory management	Forecast next week's demand
Revenue forecasting	Predict next quarter's sales
Risk management	Model interest rate paths
Trading strategies	Test autocorrelation of returns

Key Takeaways:

- AR models use past information — simple yet powerful.
- AR(1) means "today is a shadow of yesterday, plus some new noise."
- Widely used in finance (returns, rates) and business (sales, demand).
- Easy to implement, interpret, and extend.

SORBONNE UNIVERSITÉ

ARIMA Model

ARIMA

SORBONNE UNIVERSITÉ

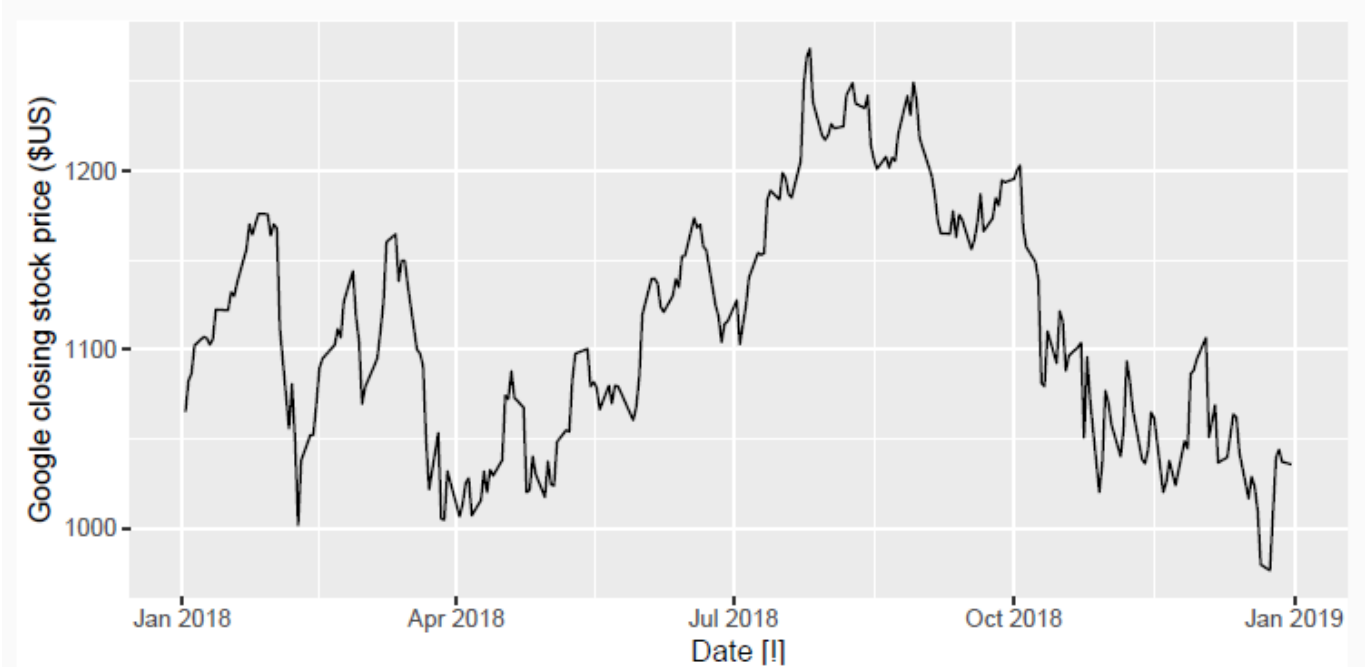
- **AR:** autoregressive (lagged observations as inputs)
- **I:** integrated (differencing to make series stationary)
- **MA:** moving average (lagged errors as inputs)

An ARIMA model is rarely interpretable in terms of visible data structures like trend and seasonality. But it can capture a huge range of time series patterns.

Stationary

Definition: If $\{y_t\}$ is a stationary time series, then for all s, the distribution of $(y_t, \dots, y_{t+s})$ does not depend on t.

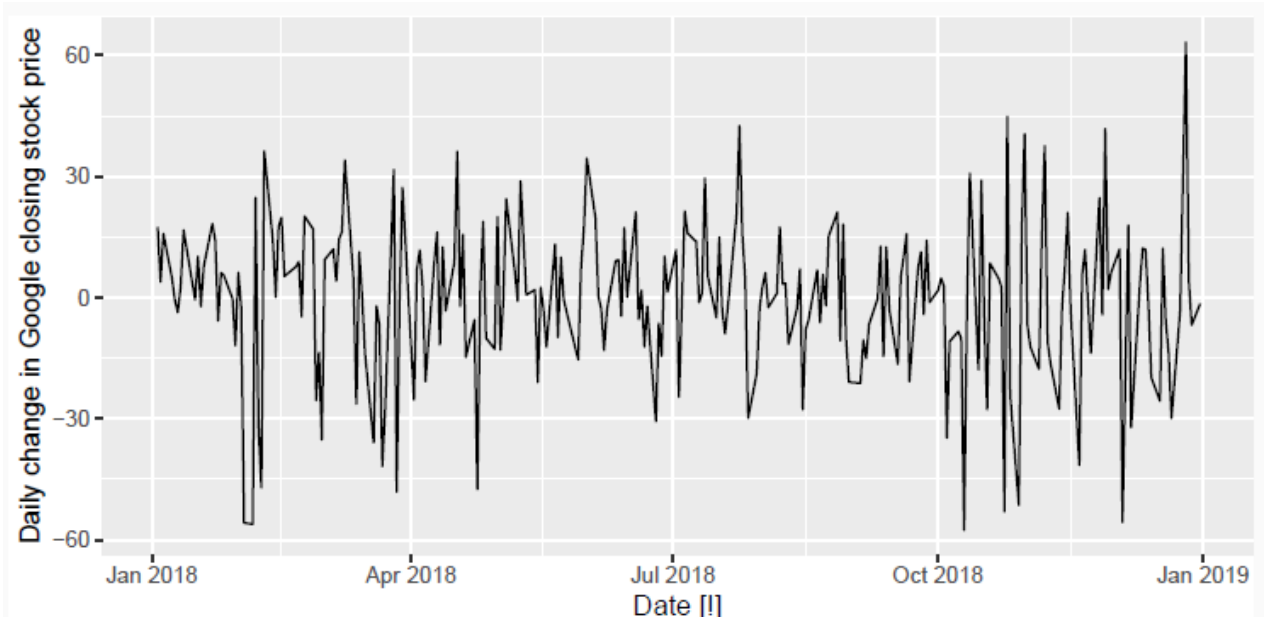
- A stationary series is:
- roughly horizontal
- constant variance
- no patterns predictable in the long-term



Daily close price of Google Stock during Jan – Dec 2018.

Differencing

- Differencing helps to stabilize the mean.
- The differenced series is the change between each observation in the original series.
- Occasionally the differenced data will not appear stationary and it may be necessary to difference the data a second time.
- In practice, it is almost never necessary to go beyond second-order differences.



Daily change in close price of Google Stock during Jan – Dec 2018.

ARIMA models

SORBONNE UNIVERSITÉ

Autoregressive Moving Average models:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

- Predictors include both lagged values of y_t and lagged errors.
- Combine ARMA model with differencing.
- d-differenced series follows an ARMA model.
- Need to choose p, d, q and whether or not to include c.

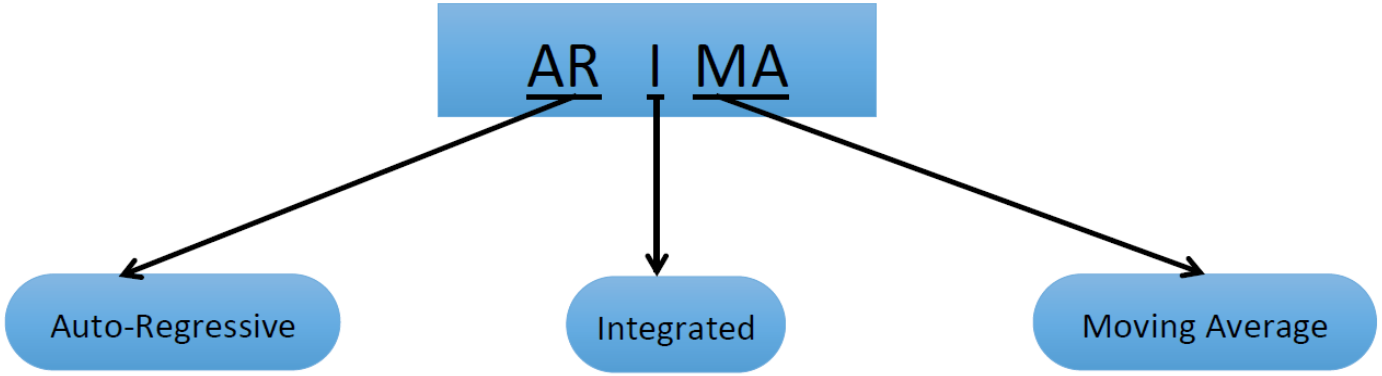
ARIMA models

ARIMA(p, d, q) model

AR: p = order of the autoregressive part

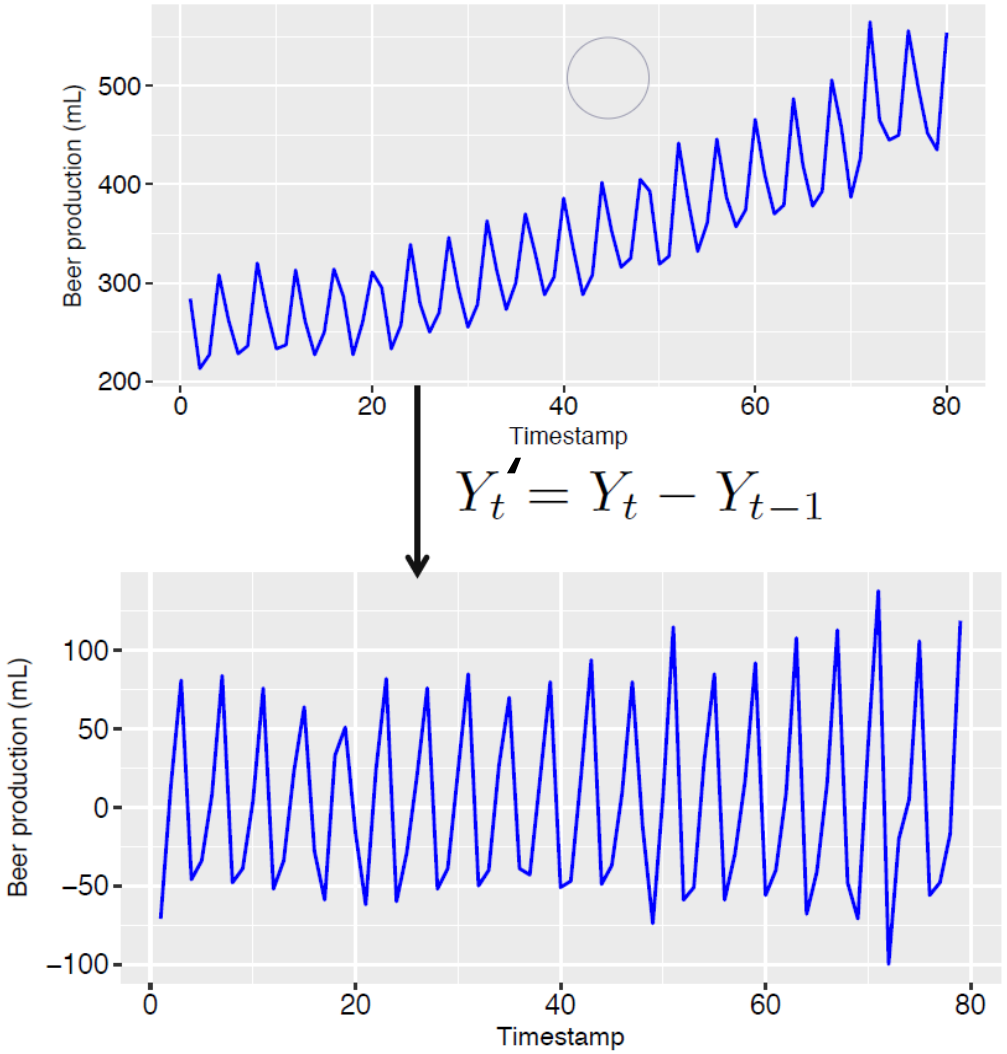
I: d = degree of first differencing involved

MA: q = order of the moving average part.



- White noise model: ARIMA(0,0,0)
- Random walk: ARIMA(0,1,0) with no constant
- Random walk with drift: ARIMA(0,1,0) with const.
- AR(p): ARIMA(p,0,0)
- MA(q): ARIMA(0,0, q)

How to select d?



Differencing order (*d*):
Number of times differencing is done

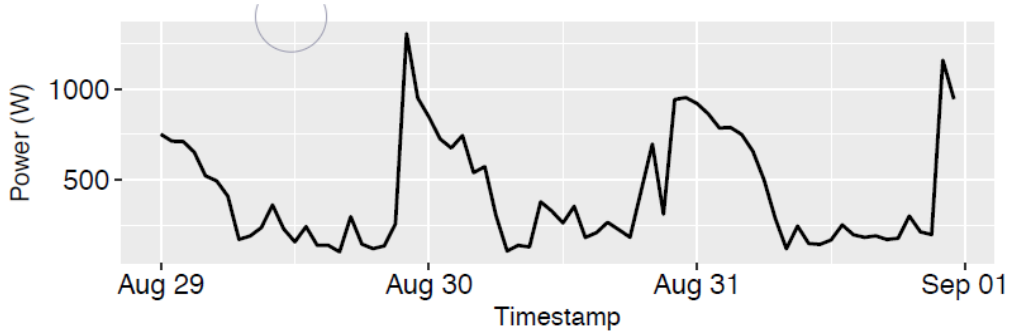
Selecting p and q manually

SORBONNE UNIVERSITÉ

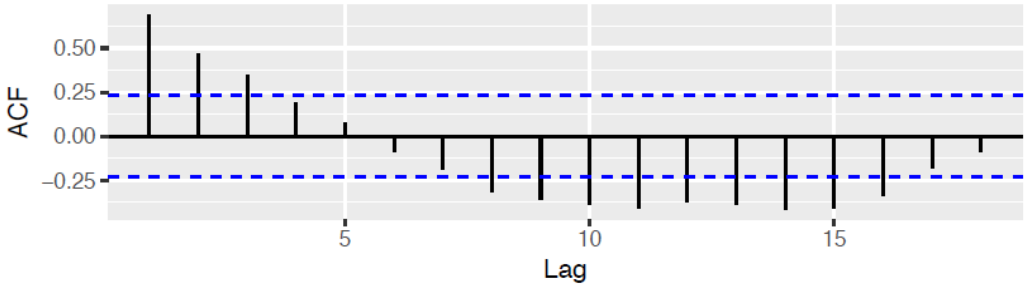
1. Auto-Correlation Function (ACF) Plot:
    - Correlation coefficients of time-series at different lags
    - Defines q order of MA model

2. Partial Auto-correlation Function (PACF) Plot:
    - Partial correlation coefficients of time series at different lags
    - Defines p order of AR model

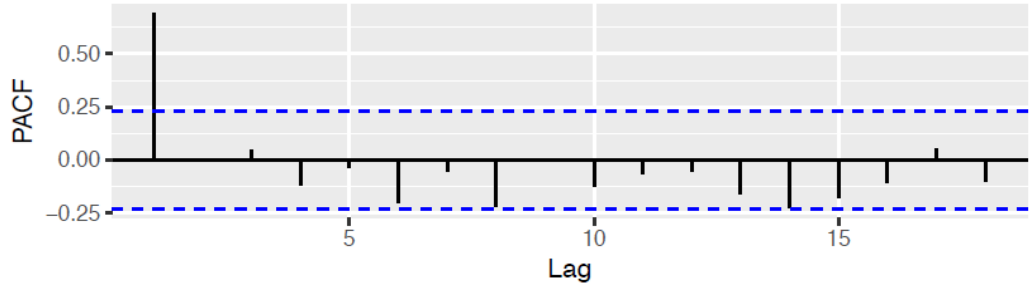
ACF / PACF Plots : Example



Data



ACF Plot



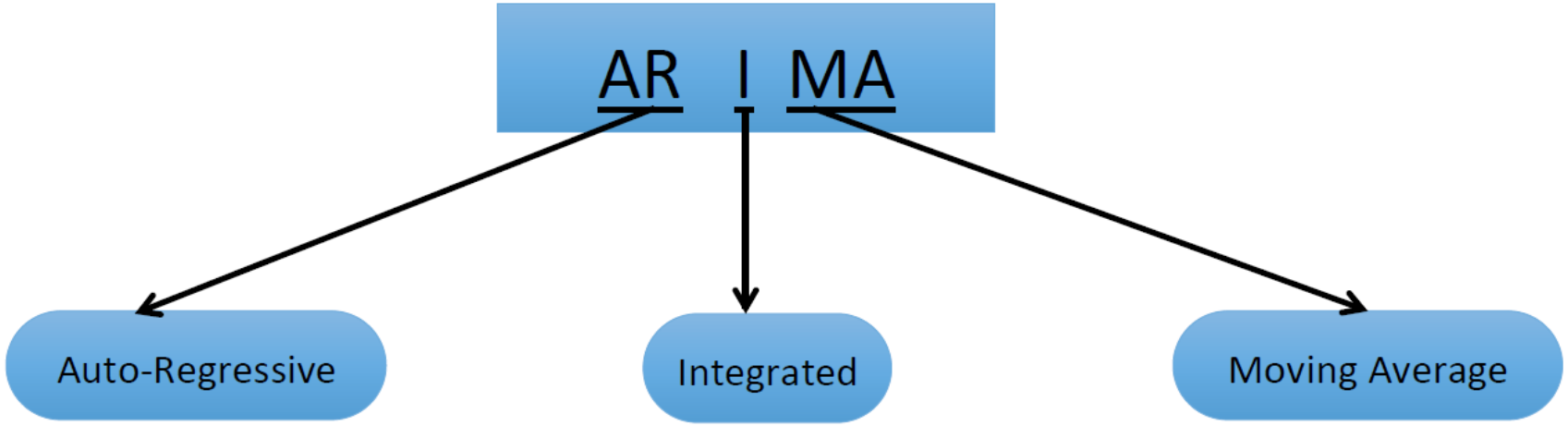
PACF Plot

Understanding ARIMA models

SORBONNE UNIVERSITÉ

- If $c = 0$ and $d = 0$, the long-term forecasts will go to zero.
- If $c = 0$ and $d = 1$, the long-term forecasts will go to a non-zero constant.
- If $c = 0$ and $d = 2$, the long-term forecasts will follow a straight line.
- If $c = 0$ and $d = 0$, the long-term forecasts will go to the mean of the data.
- If $c = 0$ and $d = 1$, the long-term forecasts will follow a straight line.
- If $c = 0$ and $d = 2$, the long-term forecasts will follow a quadratic trend.

ARIMA models



ARIMA(p, d, q) model

AR: p = order of the autoregressive part

I: d = degree of first differencing involved

MA: q = order of the moving average part.

How does ARIMA select parameters?

- Select no. differences d via differencing tests, such as Augmented Dickey Fuller (ADF) test.
- Select p, q and inclusion of c by minimizing Corrected Akaike's Information Criterion (AICc).
- Use stepwise search to traverse model space.

$$\text{AICc} = -2\log(L) + 2(p+q+k+1)\left[1 + \frac{(p+q+k+2)}{T-p-q-k-2}\right]$$

where L is the maximised likelihood fitted to the differenced data, $k = 1$ if $c \neq 0$ and $k = 0$ otherwise.

Note: Can't compare AICc for different values of d.

How does ARIMA work?

SORBONNE UNIVERSITÉ

Step1: Select current model (with smallest AICc) from:

$\text{ARIMA}(2, d, 2)$

$\text{ARIMA}(0, d, 0)$

$\text{ARIMA}(1, d, 0)$

$\text{ARIMA}(0, d, 1)$

Step 2: Consider variations of current model:

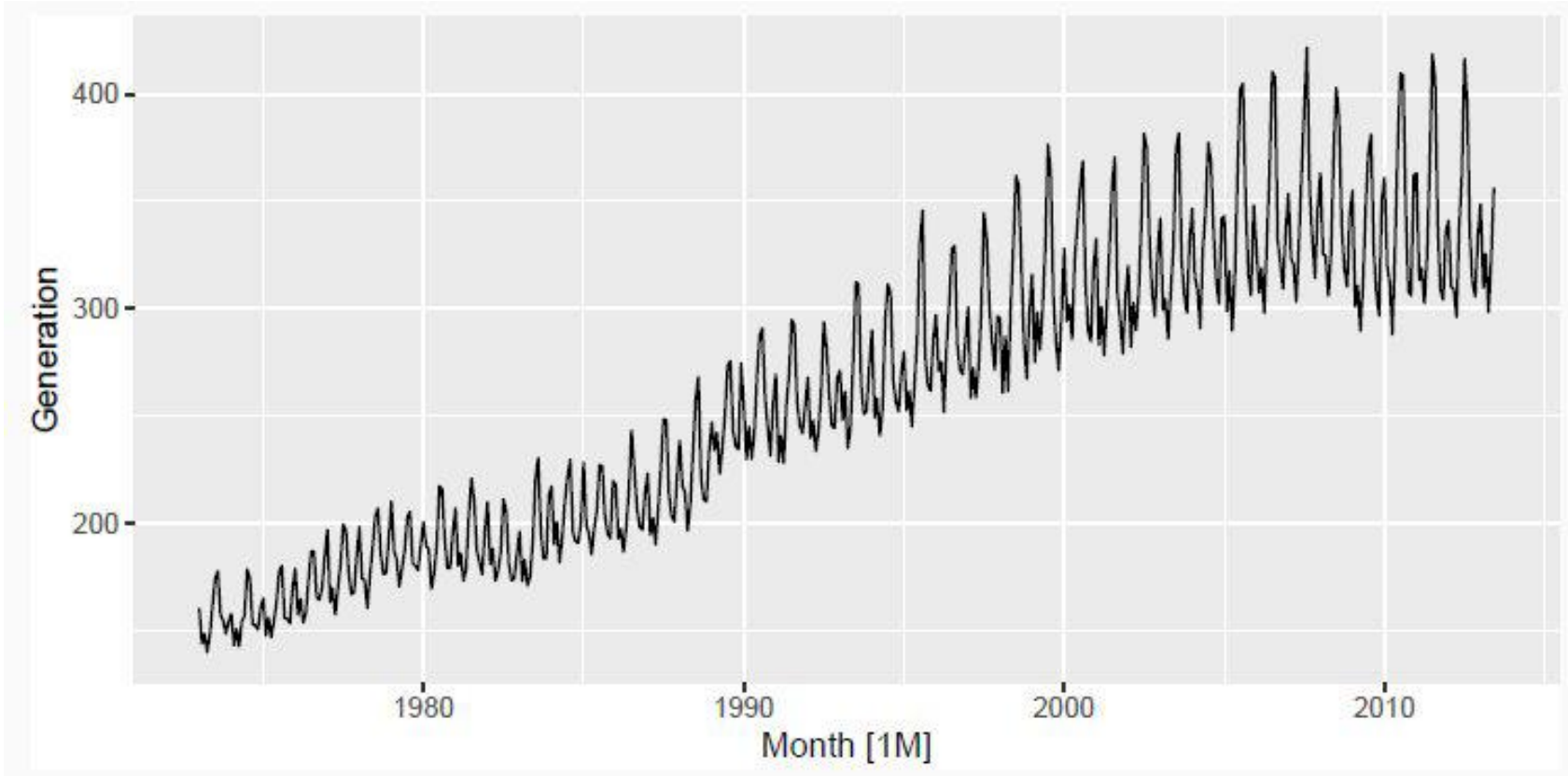
- vary one of p, q, from current model by ± 1;
- p, q both vary from current model by ± 1;
- Include/exclude c from current model.

Model with lowest AICc becomes current model.

Repeat Step 2 until no lower AICc can be found.

Seasonal ARIMA

- Many real-world time series, like temperature, sales, rainfall, electricity consumption, exhibit seasonal patterns that repeat at regular intervals. ARIMA alone cannot handle such seasonal effects effectively.



Monthly US electricity generation data from 1972 to 2013

Seasonal ARIMA

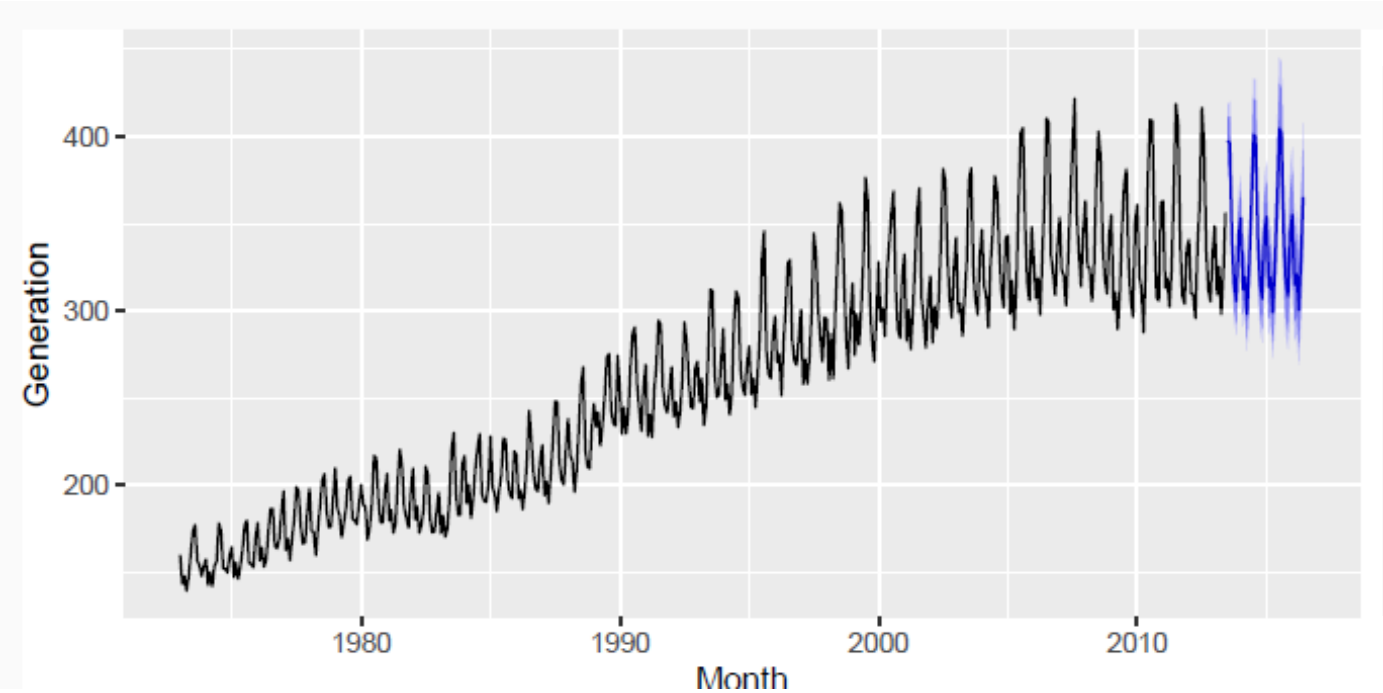
SORBONNE UNIVERSITÉ

ARIMA	(p, d, q)	$(P, D, Q)_m$
	↑	↑
	Non-seasonal part of the model	Seasonal part of of the model

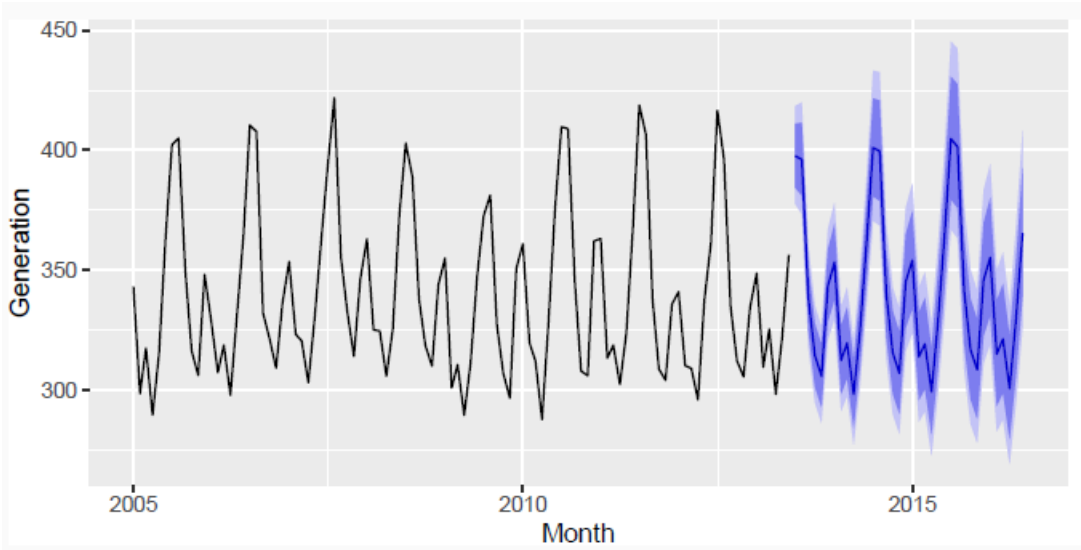
- m = number of observations per year.
- d first differences, D seasonal differences
- p AR lags, q MA lags
- P seasonal AR lags, Q seasonal MA lags

Seasonal and non-seasonal terms combine multiplicatively

Seasonal ARIMA Model Forecast



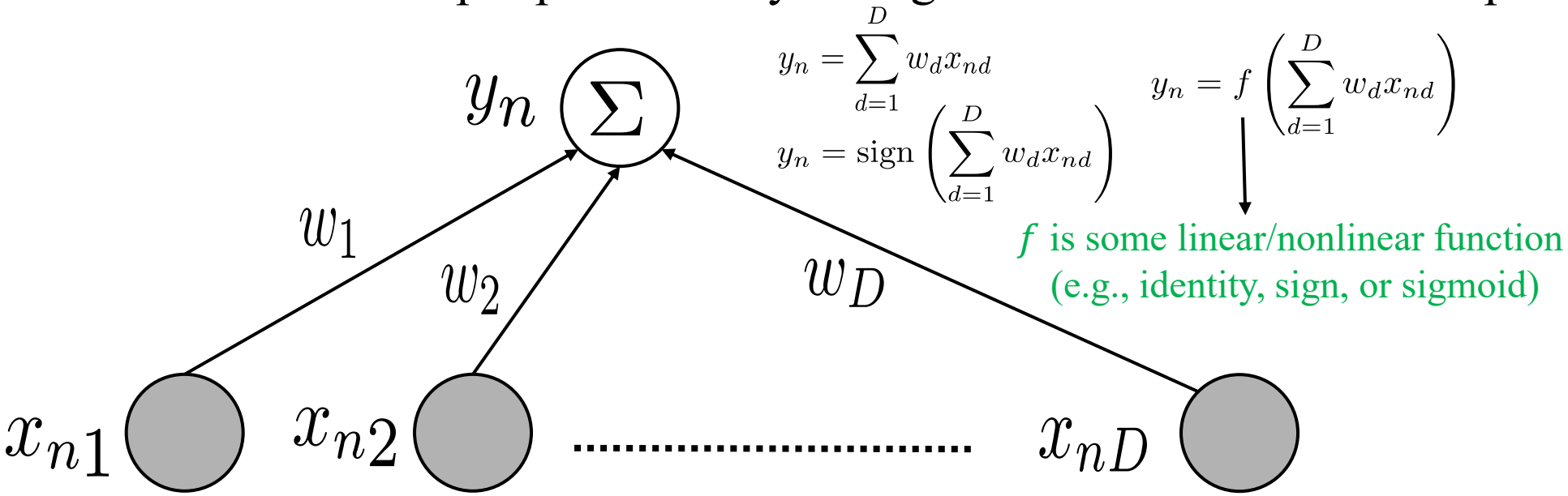
Monthly US electricity generation forecast for 2014 - 2016



Forecast horizon zoomed in

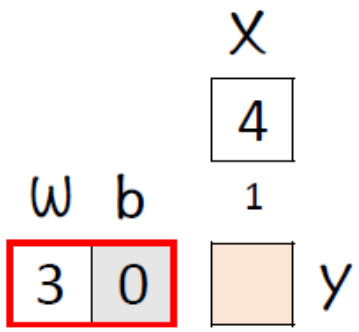
Limitation of Linear Models

- Linear models: Output produced by taking a linear combination of input features

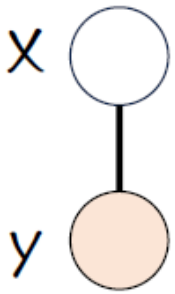


- A basic unit of the form $y = f(\boldsymbol{w}^\top \boldsymbol{x})$ is known as the "Perceptron" (not to be confused with the Perceptron "algorithm", which learns a linear classification model)
- This can't however learn nonlinear functions or nonlinear decision boundaries

Exercise: Linear Layer

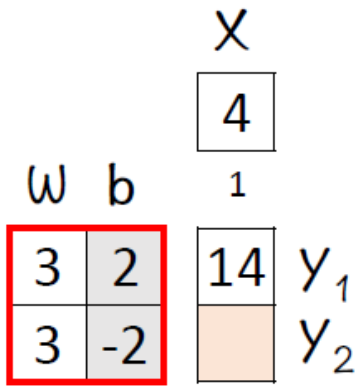


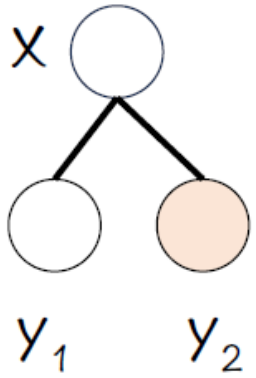
$y = [W|b] \cdot [X|1] = WX+b$



Exercise 1

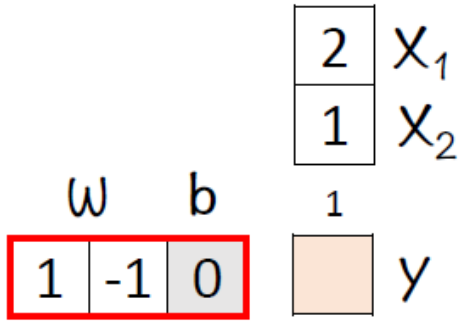
12

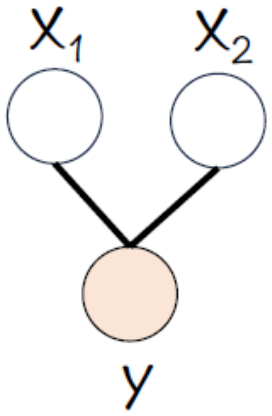




Exercise 2

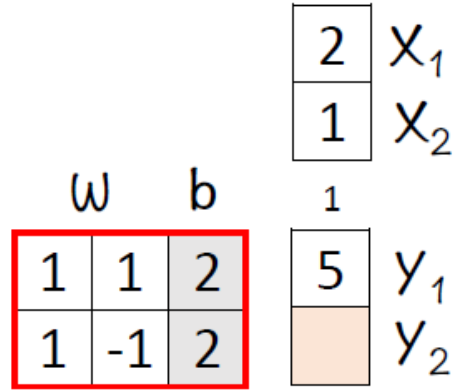
10

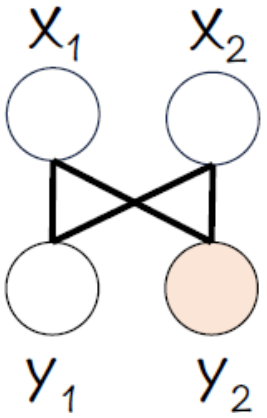




Exercise 3

1





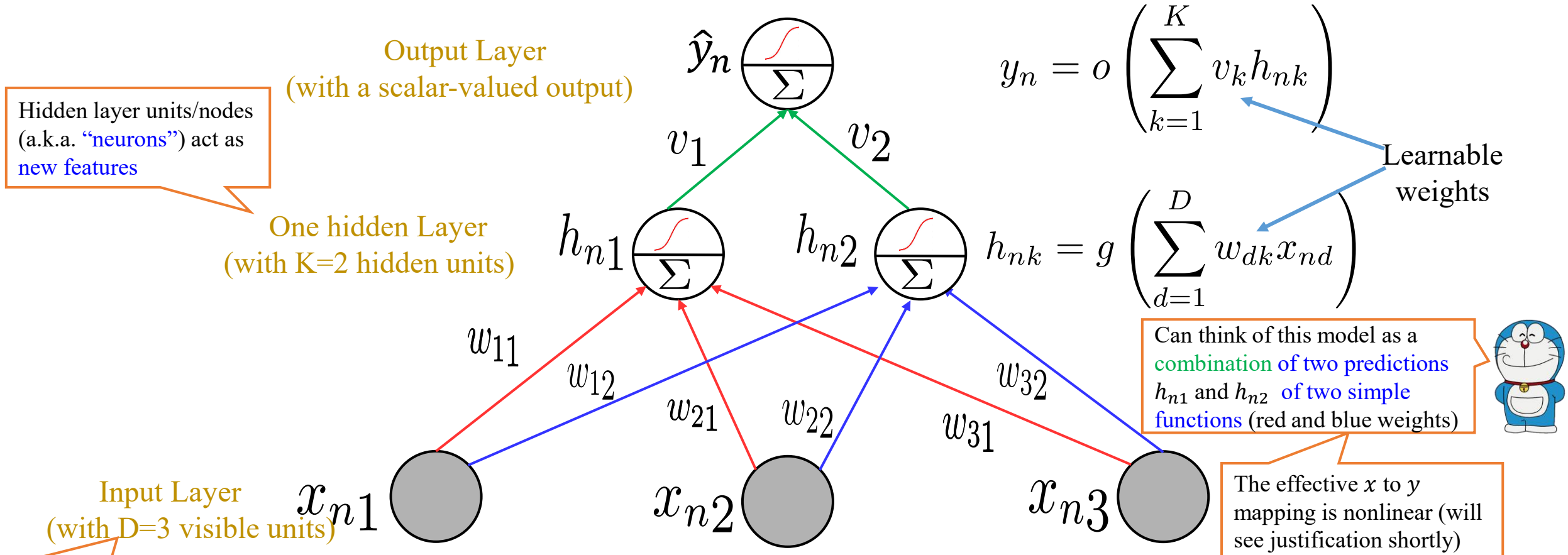
Exercise 4

3

Neural Networks: Multi-layer Perceptron (MLP)

SORBONNE UNIVERSITÉ

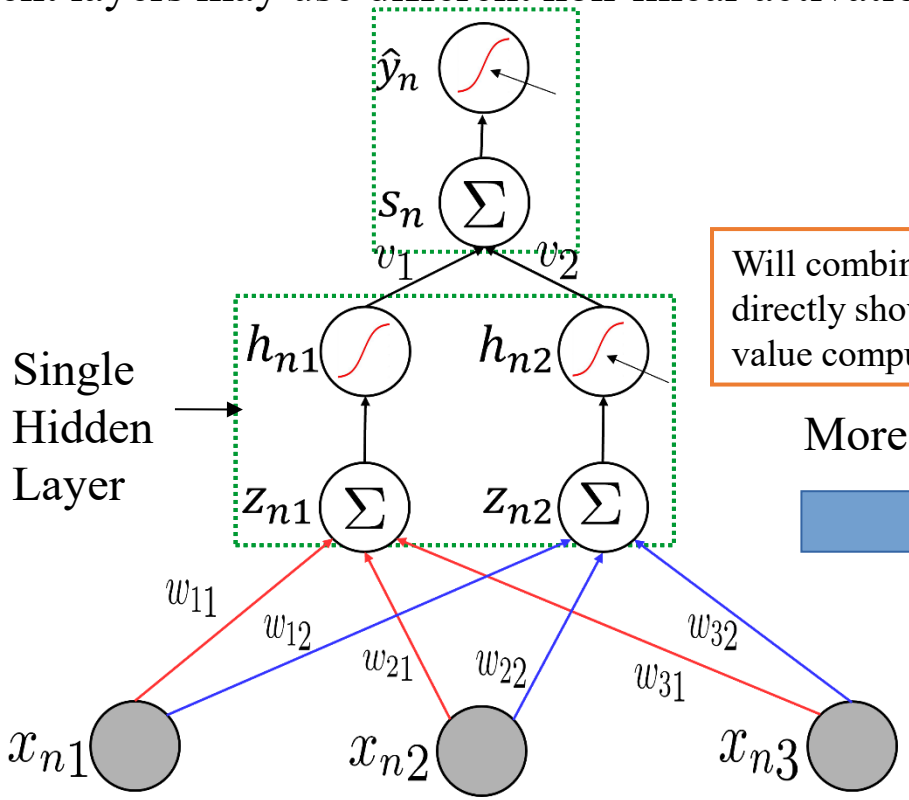
- An MLP consists of an input layer, an output layer, and one or more hidden layers



MLP is also called feedforward fully-connected network

Neural Nets: A Compact Illustration

- Note: Hidden layer pre-act z_{nk} and post-act h_{nk} will be shown together for brevity
- Different layers may use different non-linear activations. Output layer may have none.

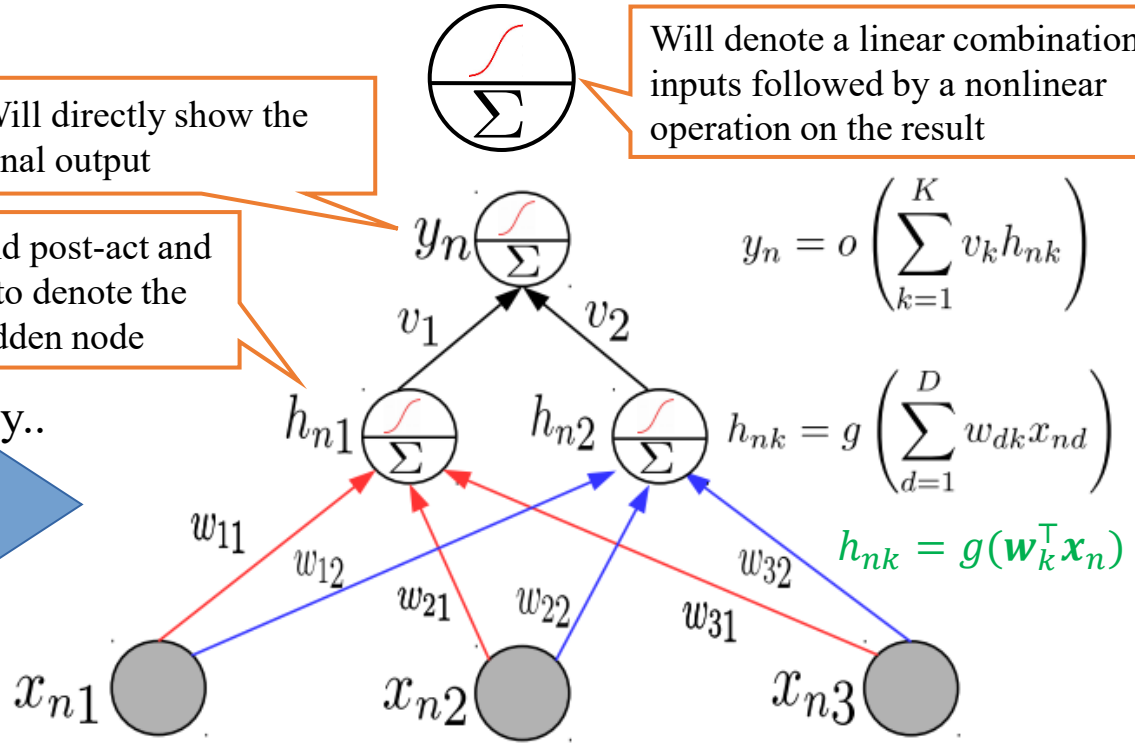


Will directly show the final output

Will combine pre-act and post-act and directly show only h_{nk} to denote the value computed by a hidden node

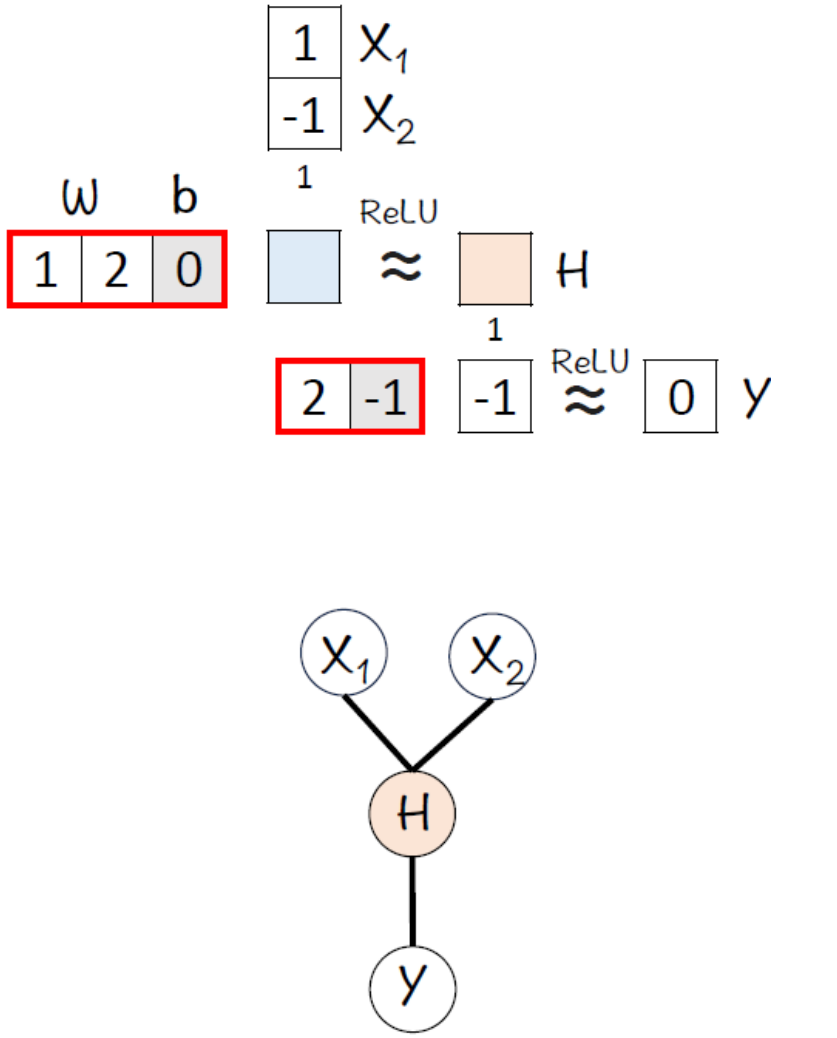
More succinctly..

Will denote a linear combination of inputs followed by a nonlinear operation on the result



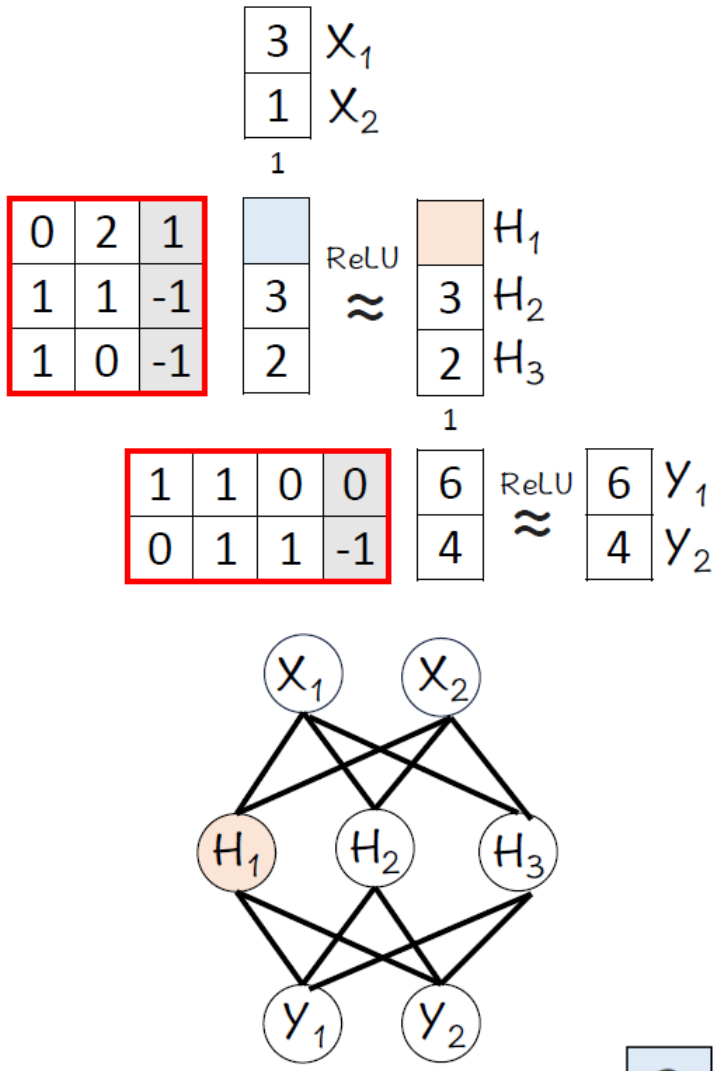
- Denoting $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$, $\boldsymbol{w}_k \in \mathbb{R}^D$, $\boldsymbol{h}_n = g(\boldsymbol{W}^\top \boldsymbol{x}_n) \in \mathbb{R}^K$ ($K = 2, D = 3$ above).
- Note: g applied elementwise on pre-activation vector $z_n = W^\top x_n$

Exercise: Hidden Layer



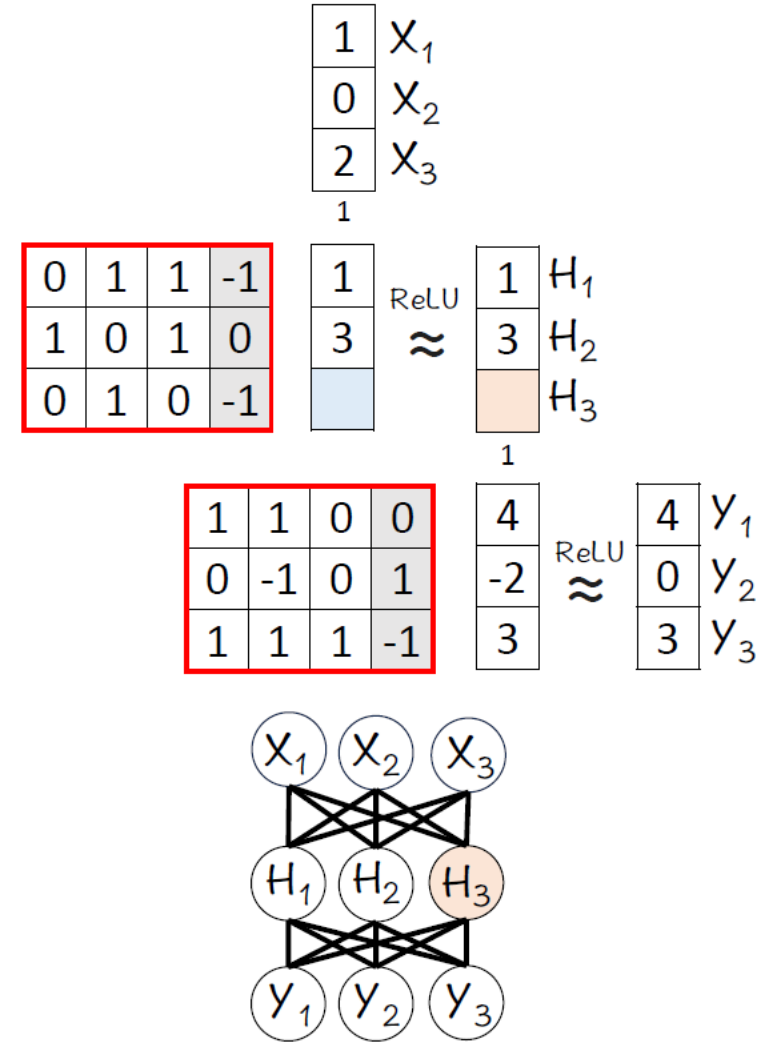
Exercise 1

-1 0



Exercise 2

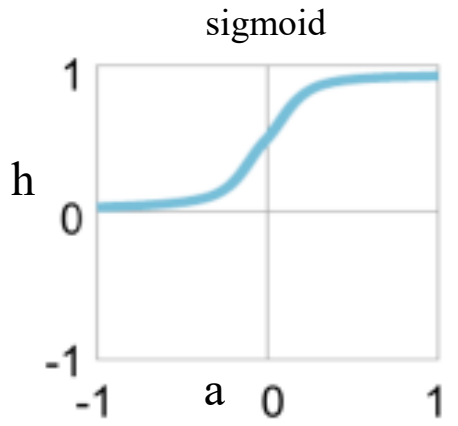
3 3



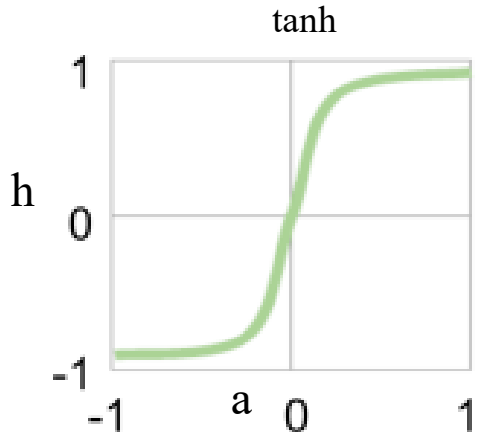
Exercise 3

-1 0

Activation Functions: Some Common Choices



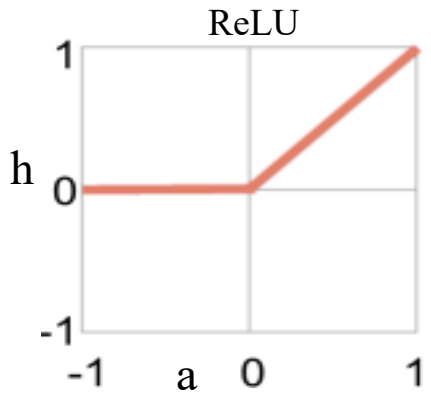
For sigmoid as well as tanh, gradients saturate (become close to zero as the function tends to its extreme values)



Preferred more than sigmoid. Helps keep the mean of the next layer's inputs close to zero (with sigmoid, it is close to 0.5)

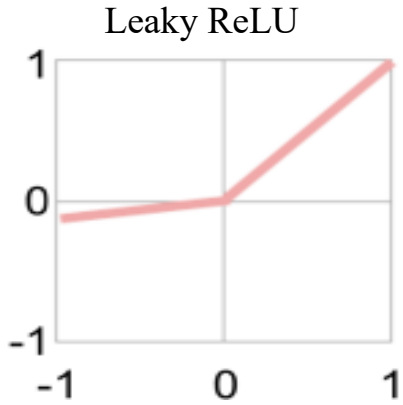
Sigmoid: $h = \sigma(a) = \frac{1}{1+\exp(-a)}$

tanh (tan hyperbolic): $h = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)} = 2\sigma(2a) - 1$



ReLU and Leaky ReLU are among the most popular ones (also efficient to compute)

Helps fix the dead neuron problem of ReLU when a is a negative number



Without nonlinear activation, a deep neural network is equivalent to a linear model no matter how many layers we use

ReLU (Rectified Linear Unit): $h = \max(0, a)$

Leaky ReLU: $h = \max(\beta a, a)$

where β is a small postive number

Exercises: Activation Functions-ReLU, Sigmoid, Tanh

SORBONNE UNIVERSITÉ

3	X_1
2	X_2

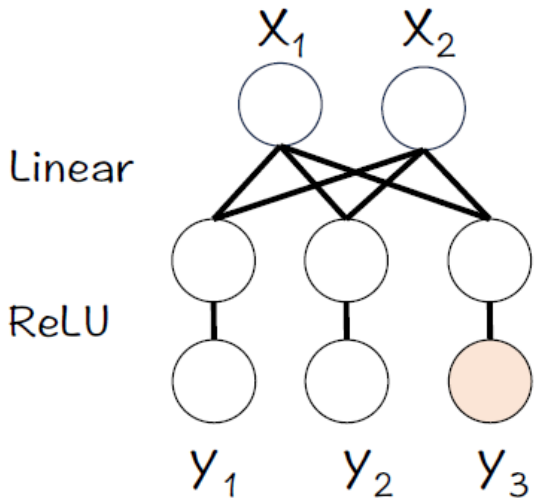
W		b
1	-1	0
1	-2	0
1	-3	0

1

1
-1
-3

ReLU ≈

1	y_1
0	y_2
	y_3



Exercise 1

0

3	X_1
2	X_2

W		b
1	-1	0
1	-2	0
1	-3	0

1

1
-1
-3

Sigmoid σ{3} ≈

0.5	y_1
	y_2
0	y_3

3 Level Quantization for hand calculation

Integer	Sigmoid
>= 2	1
-1, 0, 1	0.5
<= -2	0

Exercise 2

0.5

3	X_1
2	X_2

W		b
1	-1	-2
1	-2	-2
1	-3	-2

1

-1
-3
-5

Tanh{3} ≈

0	y_1
	y_2
-1	y_3

3 Level Quantization for hand calculation

Integer	Sigmoid	Tanh
>= 2	1	1
-1, 0, 1	0.5	0
<= -2	0	-1

Exercise 3

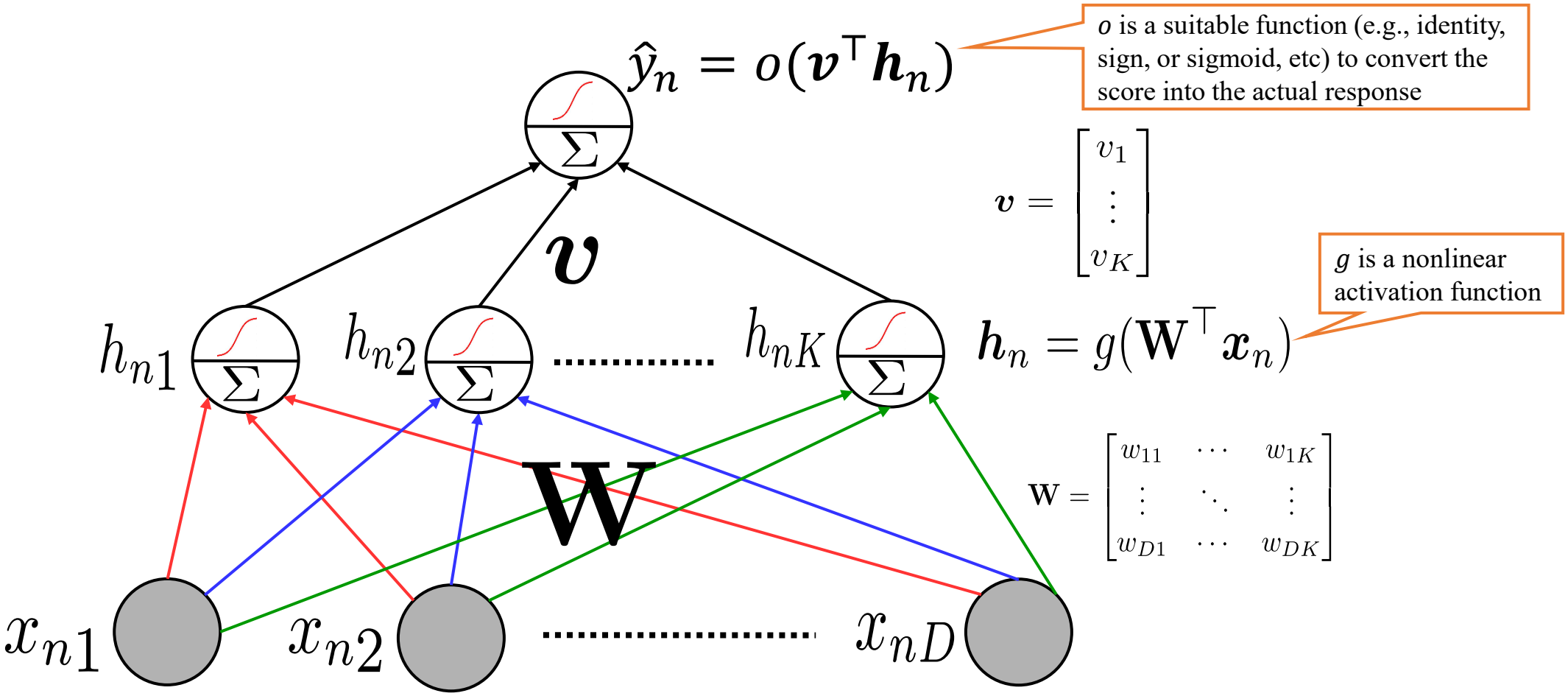
-1

SORBONNE UNIVERSITÉ

Examples of some basic NN/MLP architectures

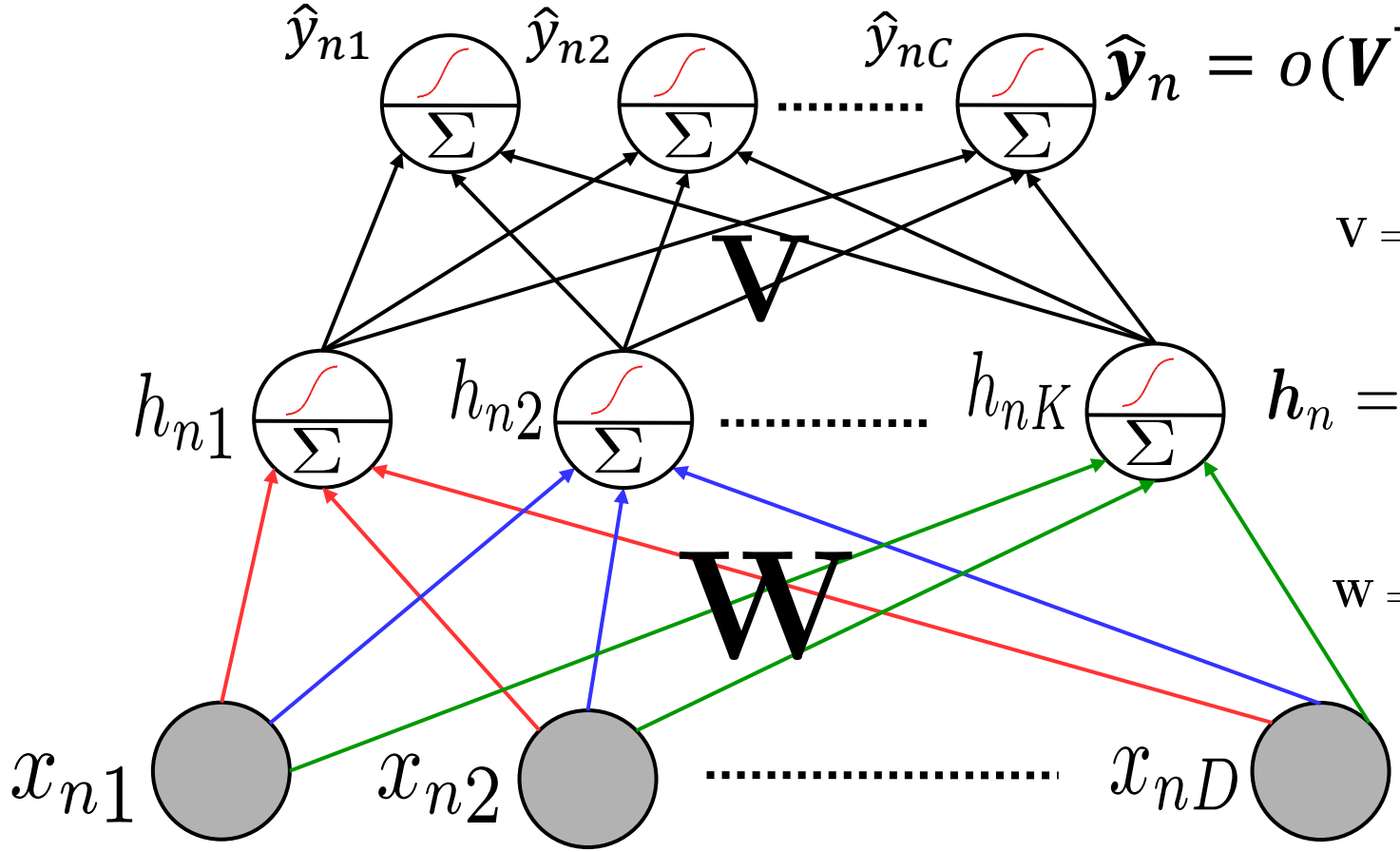
Single Hidden Layer and Single Output

- One hidden layer with K nodes and a single output (e.g., scalar-valued regression or binary classification)



Single Hidden Layer and Multiple Outputs

- One hidden layer with K nodes and a vector of C outputs (e.g., vector-valued regression or multi-class classification or multi-label classification)



$$\hat{\boldsymbol{y}}_n = o(\boldsymbol{V}^\top \boldsymbol{h}_n)$$

o is a suitable function (e.g., identity, sign, or softmax, etc) to convert the score into the actual response vector for x_n

$$\mathbf{V} = \begin{bmatrix} v_{11} & \cdots & v_{1C} \\ \vdots & \ddots & \vdots \\ v_{K1} & \cdots & v_{KC} \end{bmatrix}$$

$$\boldsymbol{h}_n = g(\mathbf{W}^\top \boldsymbol{x}_n)$$

g is a nonlinear activation function

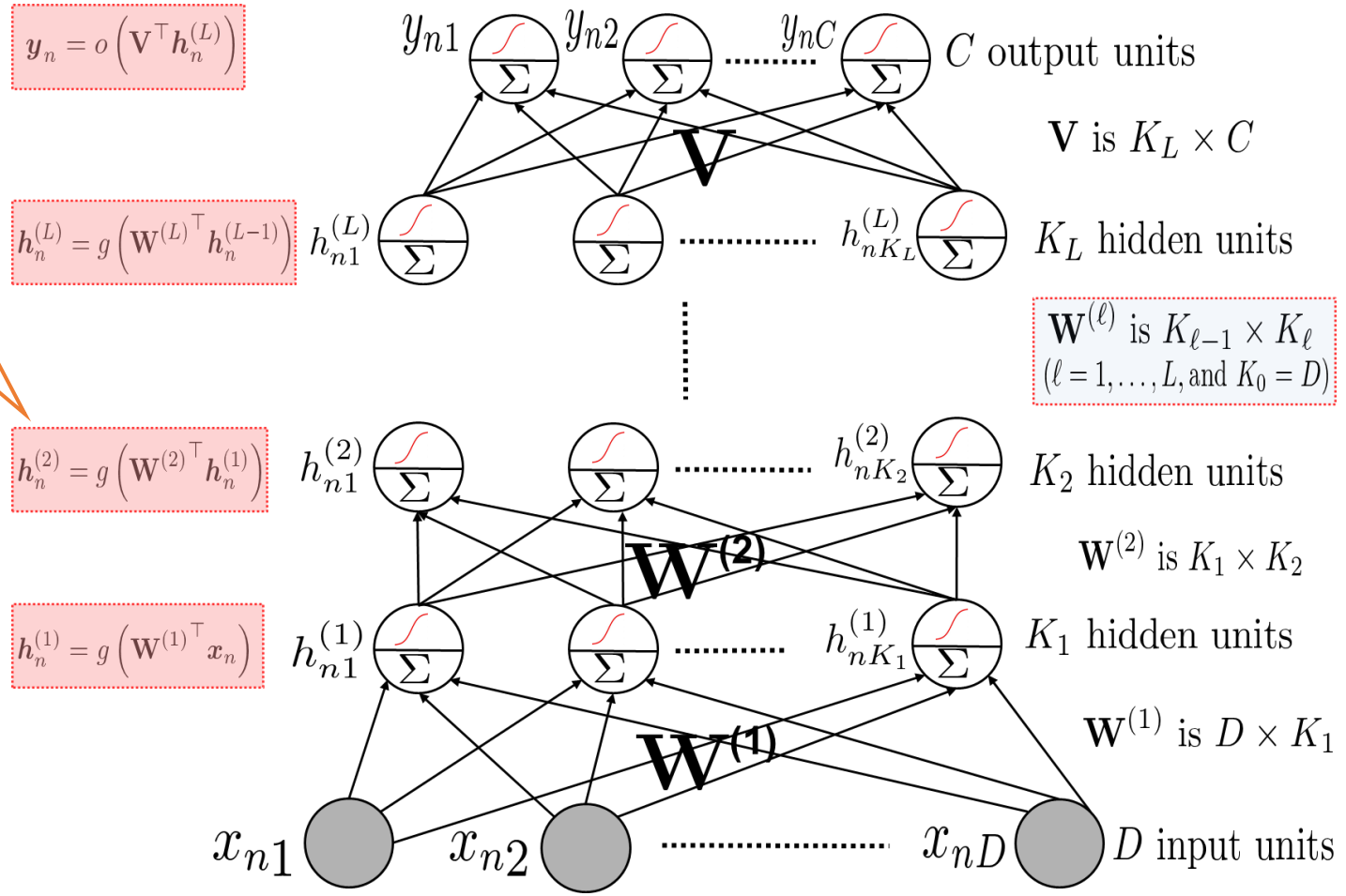
$$\mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1K} \\ \vdots & \ddots & \vdots \\ w_{D1} & \cdots & w_{DK} \end{bmatrix}$$

Multiple Hidden Layers (One/Multiple Outputs)

- Most general case: Multiple hidden layers with (with same or different number of hidden nodes in each) and a scalar or vector-valued output

Each hidden layer uses a nonlinear activation function g (essential, otherwise the network can't learn nonlinear functions and reduces to a linear model)

Note: Nonlinearity g is applied element-wise on its inputs so $h_n^{(\ell)}$ has the same size as vector $W^{(\ell)} h_n^{(\ell-1)}$

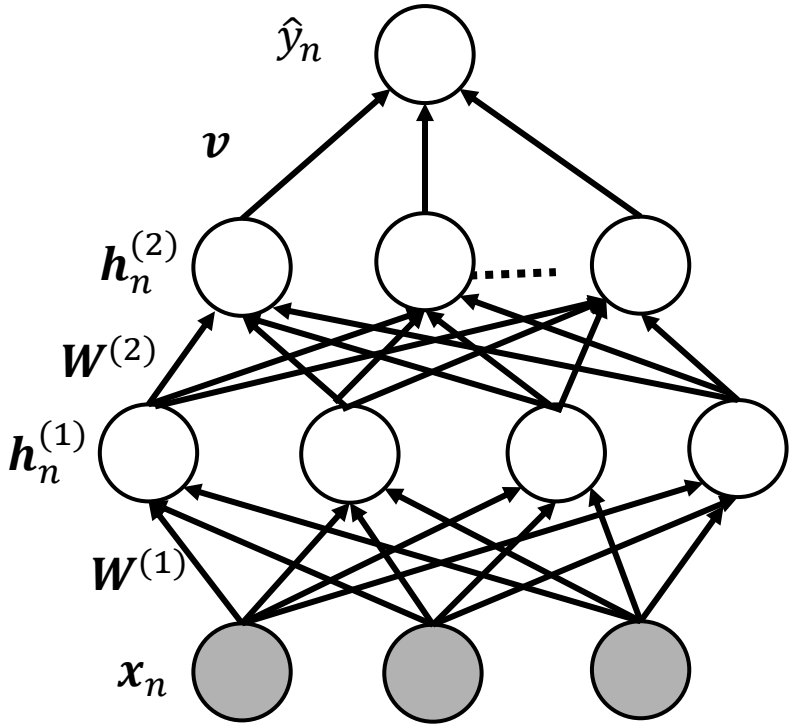


The Bias Term

- Each layer's pre-activations $\boldsymbol{z}_n^{(\ell)}$ have a an add bias term $\boldsymbol{b}^{(\ell)}$ (has the same size as $\boldsymbol{z}_n^{(\ell)}$ and $\boldsymbol{h}_n^{(\ell)}$) as well

$$\boldsymbol{z}_n^{(\ell)} = \boldsymbol{W}^{(\ell)\top} \boldsymbol{x}_n^{(\ell-1)} + \boldsymbol{b}^{(\ell)}$$

$$\boldsymbol{h}_n^{(\ell)} = g(\boldsymbol{z}_n^{(\ell)})$$

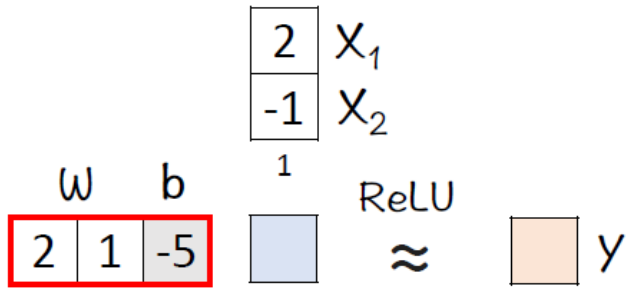


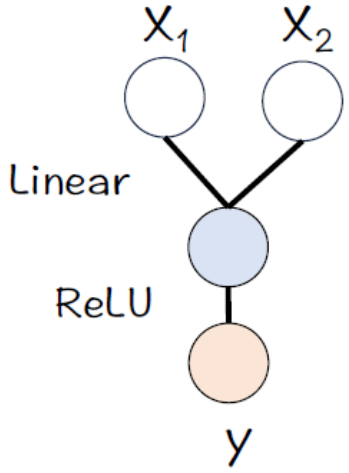
- Bias term increases the expressiveness of the network and ensures that we have nonzero activations/pre-activations even if this layer's input is a vector of all zeros

- Note that the bias term is the same for all inputs (does not depend on n)

- The bias term $\boldsymbol{b}^{(\ell)}$ is also learnable

Exercises: Artificial Neuron

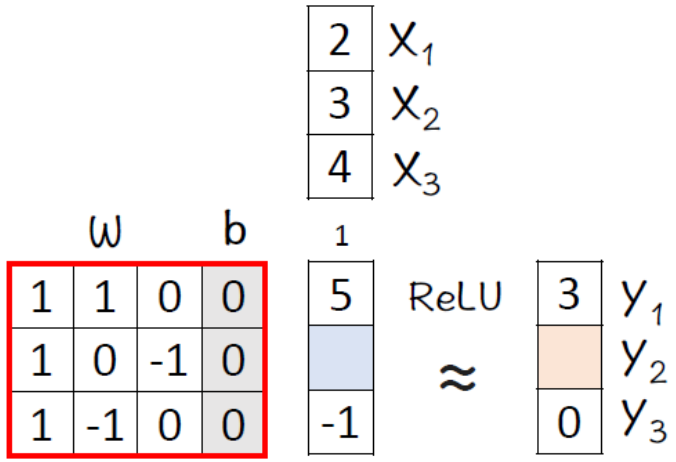


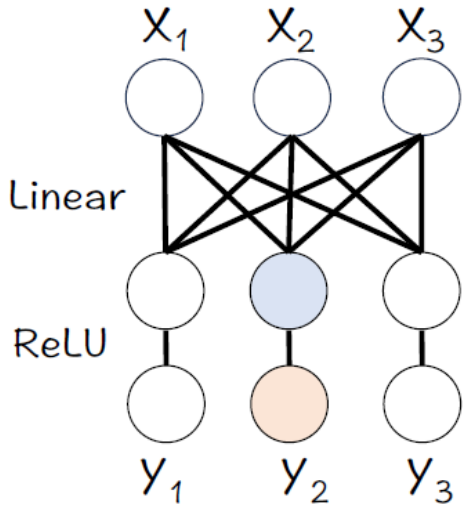


-2

0

Exercise 1

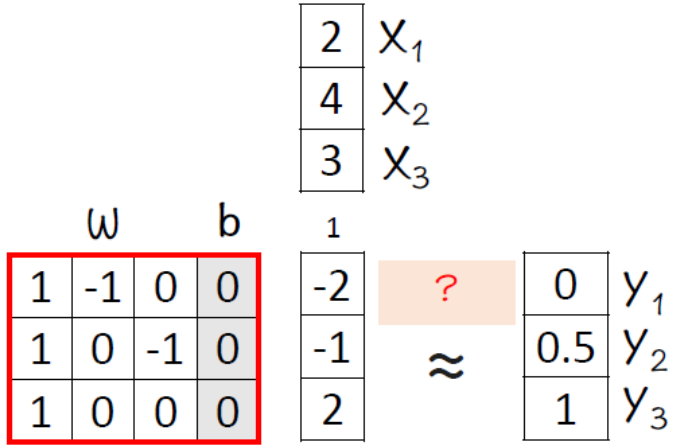


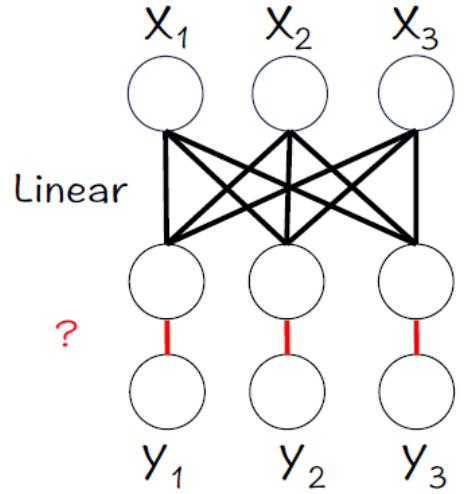


-2

0

Exercise 2



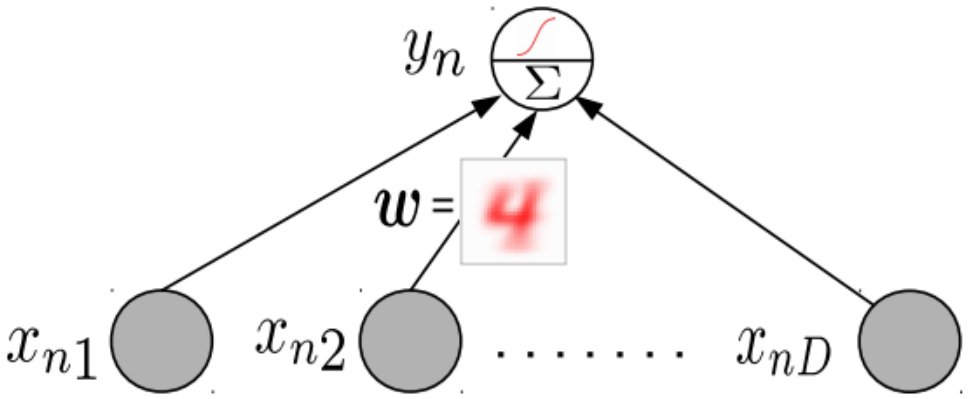


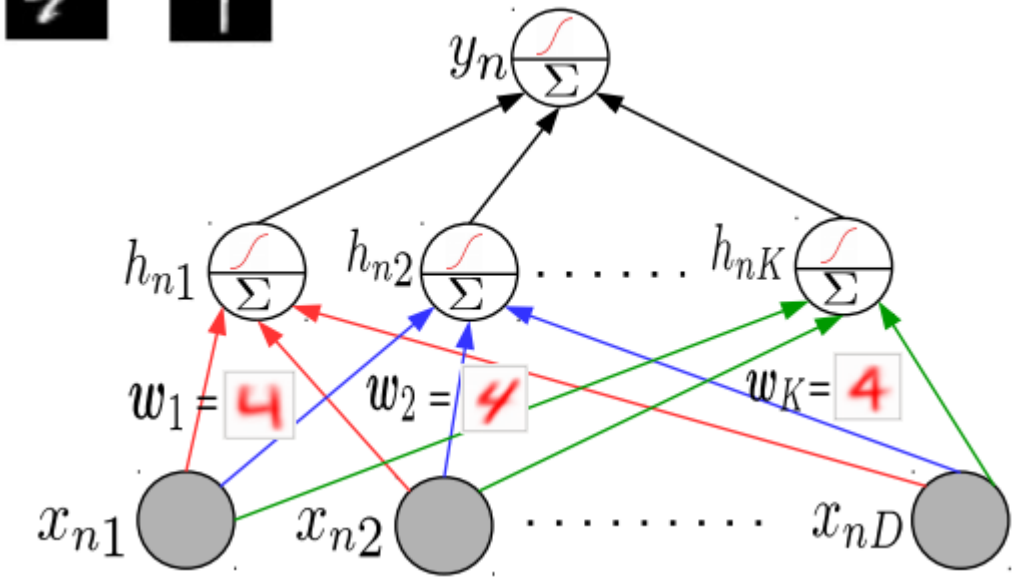
Exercise 3

Sigmoid

Why Neural Networks Work Better?

- Linear models tend to only learn the “average” pattern
  - E.g., Weight vector of a linear classification model represent average pattern of a class
- Deep models can learn multiple patterns (each hidden node can learn one pattern)
  - Thus, deep models can learn to capture more subtle variations that a simpler linear model





Backpropagation

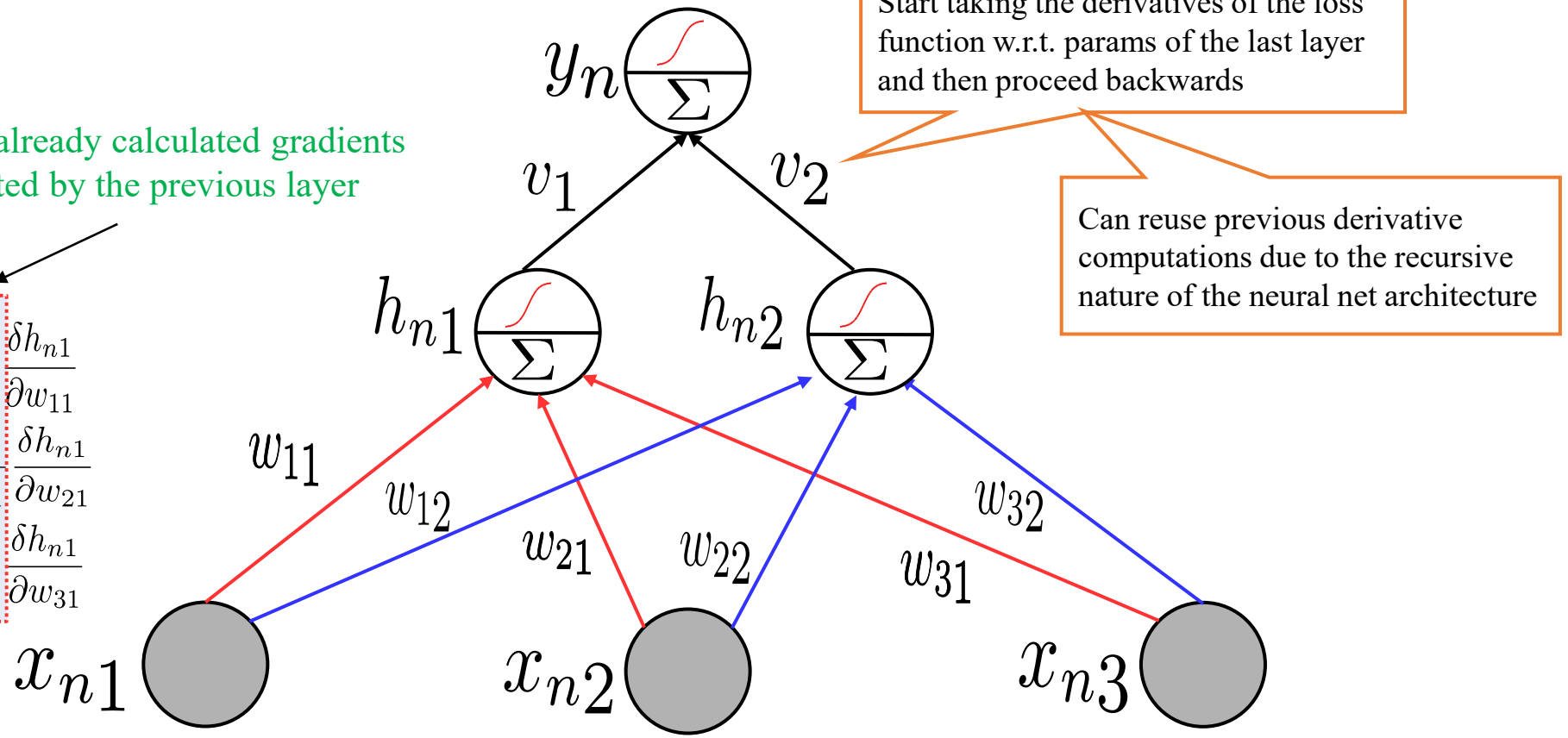
- Backpropagation = Gradient descent using chain rule of derivatives
- Chain rule of derivatives: Example, if $y = f_1(x)$ and $x = f_2(z)$ then $\frac{\partial y}{\partial z} = \frac{\partial y}{\partial x}\frac{\partial x}{\partial z}$

Start taking the derivatives of the loss function w.r.t. params of the last layer and then proceed backwards

Can reuse previous derivative computations due to the recursive nature of the neural net architecture

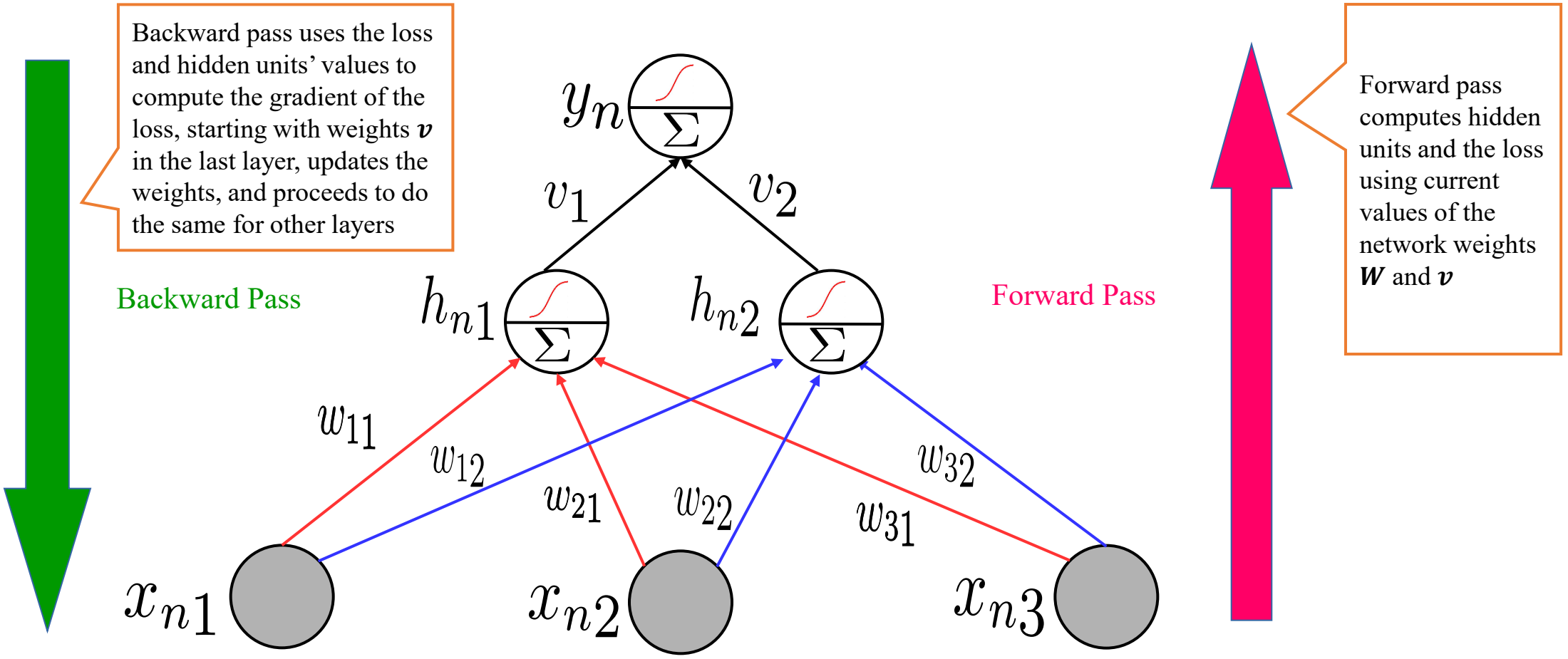
Reuse already calculated gradients computed by the previous layer

$$\frac{\delta \mathcal{L}}{\partial w_{11}} = \frac{\delta \mathcal{L}}{\partial h_{n1}}\frac{\delta h_{n1}}{\partial w_{11}}$$
$$\frac{\delta \mathcal{L}}{\partial w_{21}} = \frac{\delta \mathcal{L}}{\partial h_{n1}}\frac{\delta h_{n1}}{\partial w_{21}}$$
$$\frac{\delta \mathcal{L}}{\partial w_{31}} = \frac{\delta \mathcal{L}}{\partial h_{n1}}\frac{\delta h_{n1}}{\partial w_{31}}$$



Backpropagation

- Backprop iterates between a forward pass and a backward pass

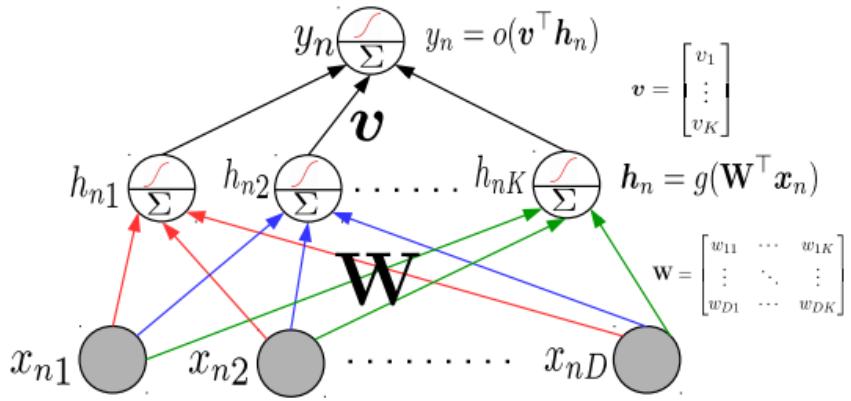


- Software frameworks such as Tensorflow and PyTorch support this

Backpropagation through an example

SORBONNE UNIVERSITÉ

Consider a single hidden layer MLP



Assuming regression (o = identity), the loss function for this model

$$\begin{aligned} \mathcal{L} &= \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \boldsymbol{v}^\top \boldsymbol{h}_n\right)^2 \\ &= \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \sum_{k=1}^{K} v_k h_{nk}\right)^2 \\ &= \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \sum_{k=1}^{K} v_k g(\boldsymbol{w}_k^\top \boldsymbol{x}_n)\right)^2 \end{aligned}$$

- To use gradient methods for $\mathbf{W}$, $\boldsymbol{v}$, we need gradients.
- Gradient of $\mathcal{L}$ w.r.t. $\boldsymbol{v}$ is straightforward

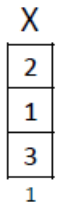
$$\frac{\partial \mathcal{L}}{\partial v_k} = -\sum_{n=1}^{N}\left(y_n - \sum_{k=1}^{K} v_k g(\boldsymbol{w}_k^\top \boldsymbol{x}_n)\right) h_{nk} = \sum_{n=1}^{N} e_n h_{nk}$$

- Gradient of $\mathcal{L}$ w.r.t. $\mathbf{W}$ requires chain rule

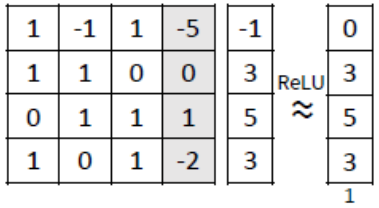
$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_{dk}} &= \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial h_{nk}} \frac{\partial h_{nk}}{\partial w_{dk}} \\ \frac{\partial \mathcal{L}}{\partial h_{nk}} &= -(y_n - \sum_{k=1}^{K} v_k g(\boldsymbol{w}_k^\top \boldsymbol{x}_n)) v_k = -e_n v_k \\ \frac{\partial h_{nk}}{\partial w_{dk}} &= g'(\boldsymbol{w}_k^\top \boldsymbol{x}_n) x_{nd} \qquad \text{(note: } h_{nk} = g(\boldsymbol{w}_k^\top \boldsymbol{x}_n)\text{)} \end{aligned}$$

- Forward prop computes errors e_n using current $\mathbf{W}$, $\boldsymbol{v}$. Backprop updates NN params $\mathbf{W}$, $\boldsymbol{v}$ using grad methods
- Backprop caches many of the calculations for reuse

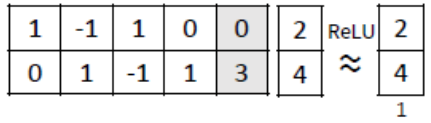
Exercise: Backpropagation



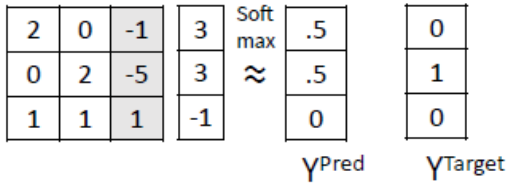
Layer 1

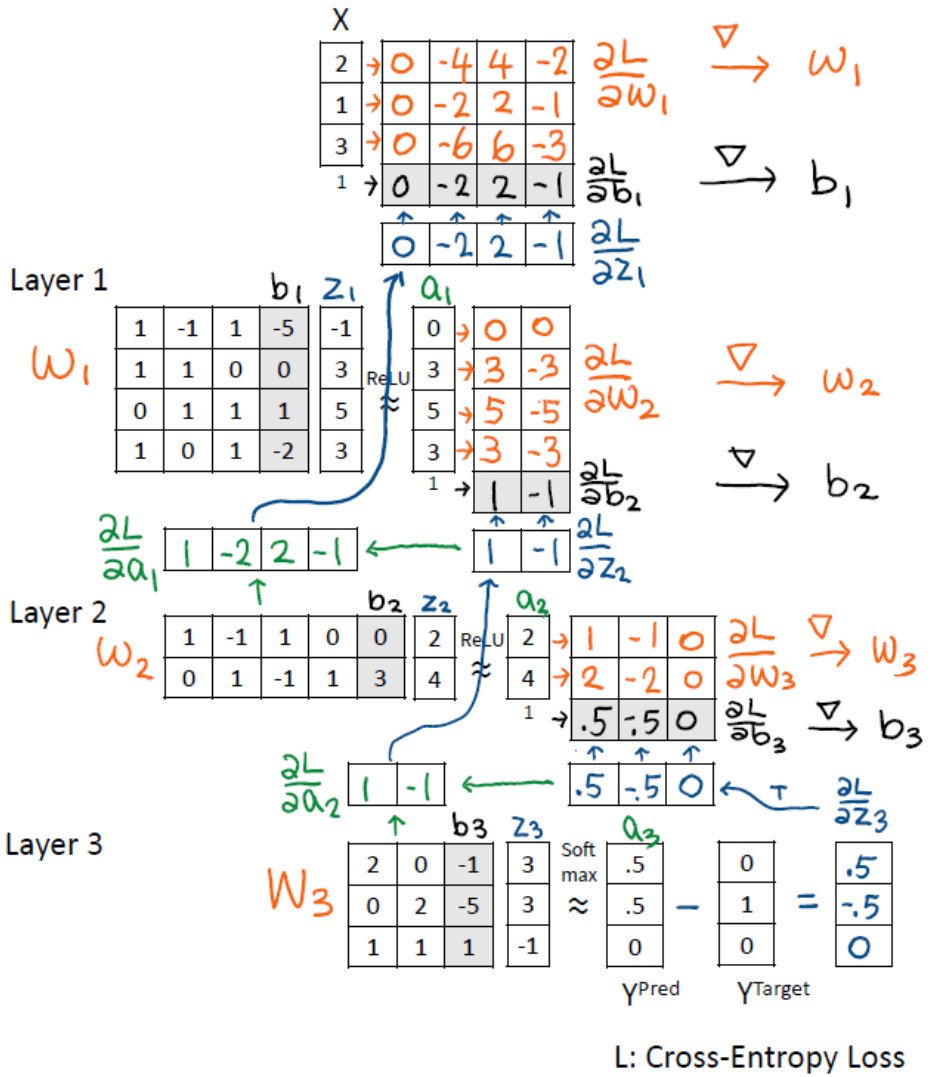


Layer 2



Layer 3





L: Cross-Entropy Loss

Exercise: Backpropagation

SORBONNE UNIVERSITÉ

Steps:

1. Forward Pass: Given a multi layer perceptron (3 levels), an input vector X, predictions Y^{Pred} = [0.5,0.5,0], and ground truth label Y^{Target} =[0,1,0].

2. Backpropagation: Insert cells to hold our calculations.

3. Layer 3 - Softmax (blue): Calculate ∂L/∂z3 directly using the simple equation: $Y^{Pred} - Y^{Target}$ = [0.5,-0.5,0]. This simple equation is the benefit of using Softmax and Cross Entropy Loss together.

4. Layer 3 - Weights (orange) & Biases (black): Calculate ∂L/∂W3 and ∂L/∂b3 by multiplying ∂L/∂z3 and [a2|1].

5. Layer 2 - Activations (green): Calculate ∂L/∂a2 by multiplying ∂L/∂z3 and W3.

6. Layer 2 - ReLU (blue): Calculate ∂L/∂z2 by multiplying ∂L/∂a2 with 1 for positive values and 0 otherwise.

7. Layer 2 - Weights (orange) & Biases (black): Calculate ∂L/∂W2 and ∂L/∂b2 by multiplying ∂L/∂z2 and [a1|1].

8. Layer 1 - Activations (green): Calculate ∂L/∂a1 by multiplying ∂L/∂z2 and W2.

9. Layer 1 - ReLU (blue): Calculate ∂L/∂z1 by multiplying ∂L/∂a1 with 1 for positive values and 0 otherwise.

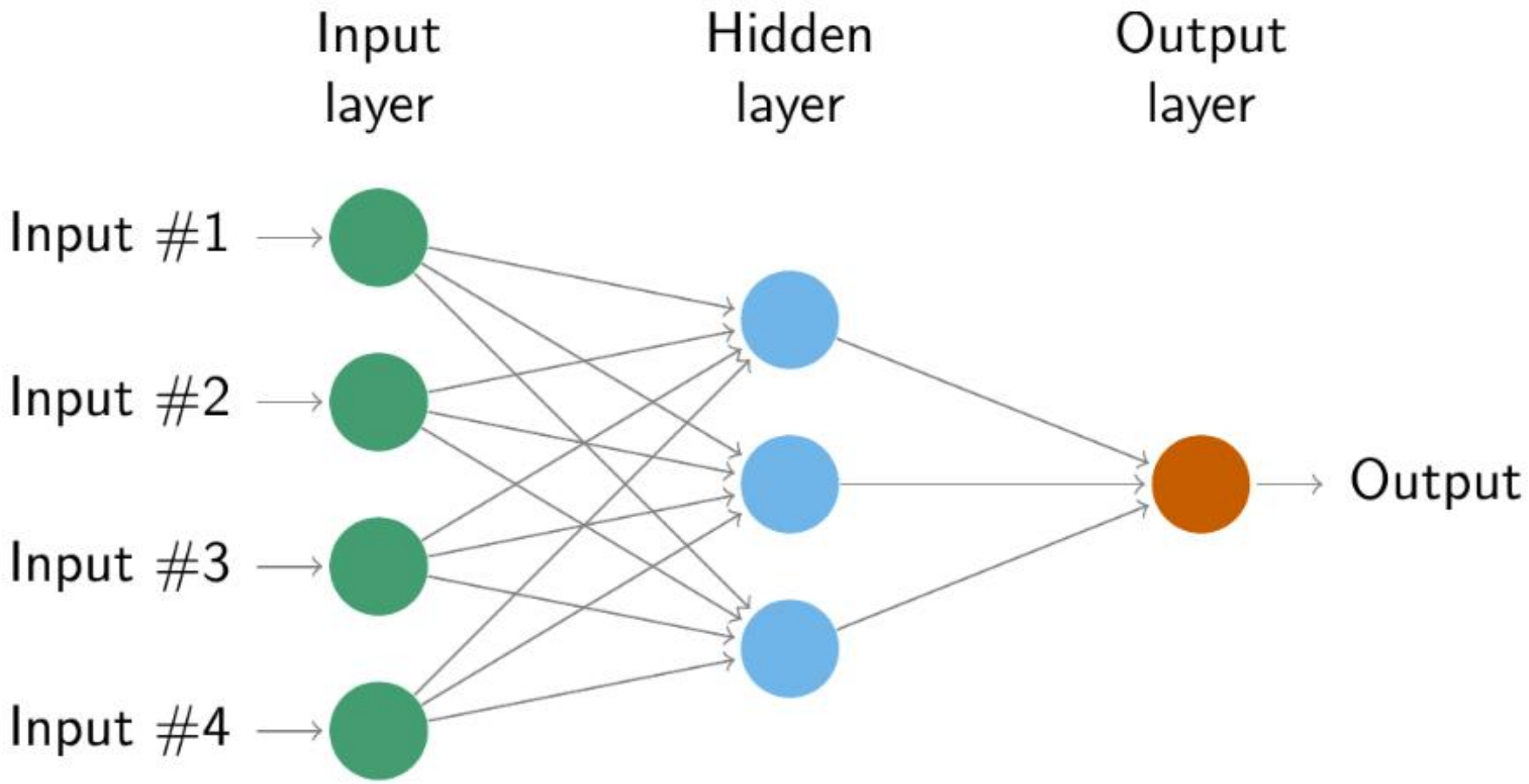
10. Layer 1 - Weights (orange) & Biases (black): Calculate ∂L/∂W1 and ∂L/∂b1 by multiplying ∂L/∂z1 and [x|1].

11. Gradient Descent: Update weights and biases (typically a learning rate is applied here).

SORBONNE UNIVERSITÉ

Autoregressive Neural Network

Feed-forward Neural Network



ARNN for Forecasting

SORBONNE UNIVERSITÉ

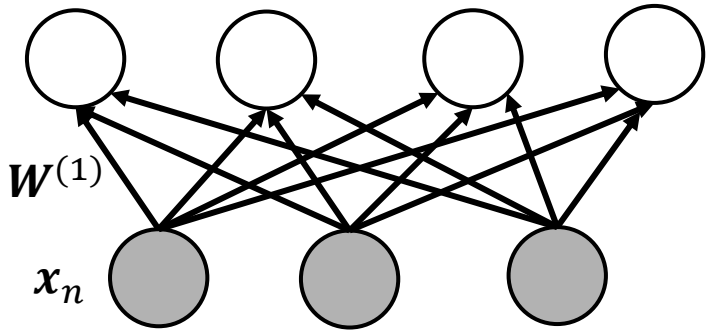
- With time series data, lagged values of the time series can be used as inputs to a neural network, typically named as Autoregressive neural network (ARNN)
- ARNN(p, k) is a feed-forward network with one hidden layer, with p lagged inputs and k nodes in the hidden layer.
- ARNN$(p, 0)$ model is equivalent to an ARIMA$(p, 0,0)$ model, but without the restrictions on the parameters to ensure stationarity.
- With seasonal data, it is useful to also add the last observed values from the same season as inputs.
- For time series, the default is the optimal number of lags (according to the AIC) for a linear AR(p) model. If k is not specified, it is set to $k = (p + 1)/2$ (rounded to the nearest integer).
- When it comes to forecasting, the network is applied iteratively.

Limitations/Shortcomings of MLP & ARNN

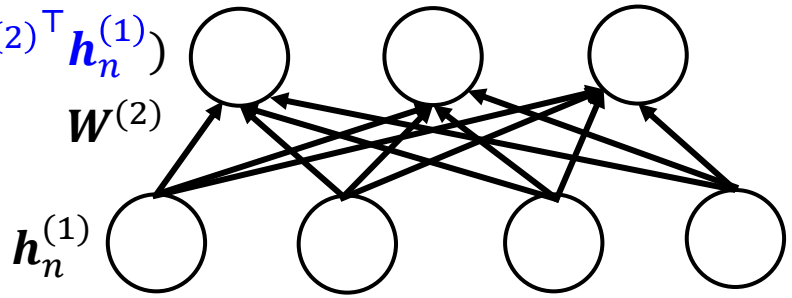
SORBONNE UNIVERSITÉ

- MLP uses fully connected layers defined by matrix multiplications + nonlinearity

$\boldsymbol{h}_n^{(1)} = g(\boldsymbol{W}^{(1)\top}\boldsymbol{x}_n)$



$\boldsymbol{h}_n^{(2)} = g(\boldsymbol{W}^{(2)\top}\boldsymbol{h}_n^{(1)})$

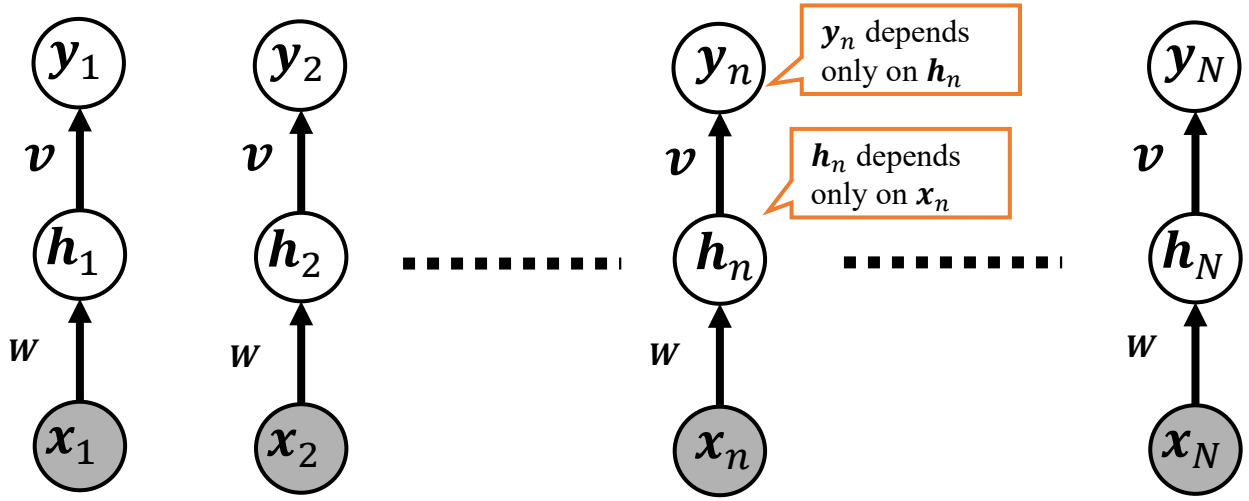


- MLP ignores structure (e.g., spatial/sequential) in the inputs
  - Not ideal for data such as images, text, etc. which are flattened as vectors when used with MLP

- Fully connected nature of MLP requires massive number of weights
  - Recall that each layer is fully connected so each layer needs a massive number of weights!

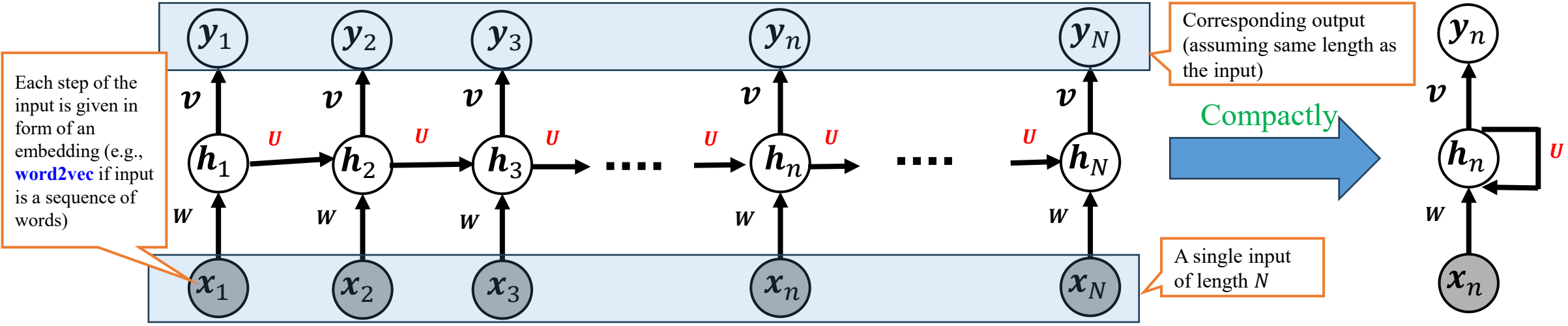
Recurrent Connections in Deep Neural Networks

- Feedforward nets such as MLP assume independent observations



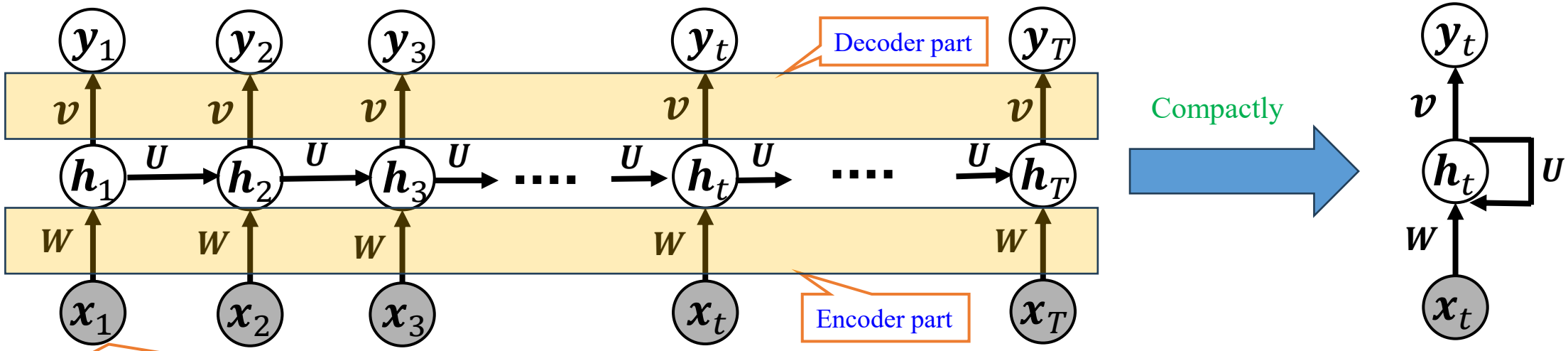
Feedforward neural networks are not ideal when inputs $[x_1, x_2, \ldots, x_N]$ and/or outputs $[y_1, y_2, \ldots, y_N]$ represent sequential data (e.g., sequence of words, video (sequence of frames), etc.

- A recurrent structure can be helpful if each input and/or output is a sequence



RNNs

- RNNs are used when each input or output or both are sequences of tokens

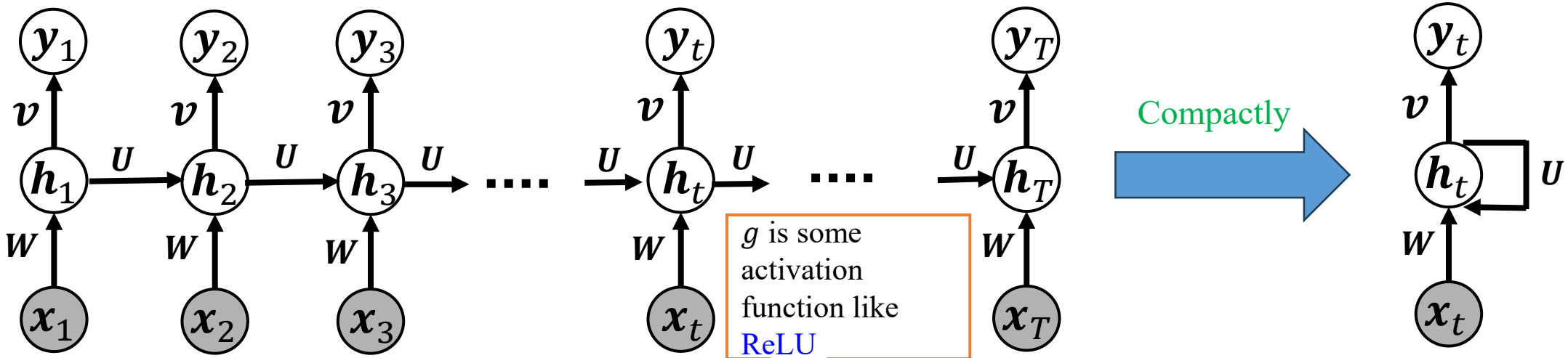


If the input is a word sequence, then each x_n represent the corresponding word's embedding (either a pre-computed word embedding like word2vec or a learned word embedding)

- Hidden state h_t is supposed to remember everything up to time $t-1$. However, in practice, RNNs have difficulties remembering the distant past
  - Variants such as LSTM, GRU, etc mitigate this issue to some extent
- Slow processing is another major issue (e.g., can't compute h_t before computing h_{t-1})

Recurrent Neural Networks

- A basic RNN's architecture (assuming input and output sequence have same lengths)
- RNN has three sets of weights $\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{v}$



g is some activation function like ReLU

- $\boldsymbol{W}$ and $\boldsymbol{U}$ model how h_t at step t is computed: $\boldsymbol{h_t} = g(\boldsymbol{W x_t} + \boldsymbol{U h_{t-1}})$
- $\boldsymbol{v}$ models the hidden layer to output mapping, e.g., $\boldsymbol{y_t} = o(\boldsymbol{v h_t})$

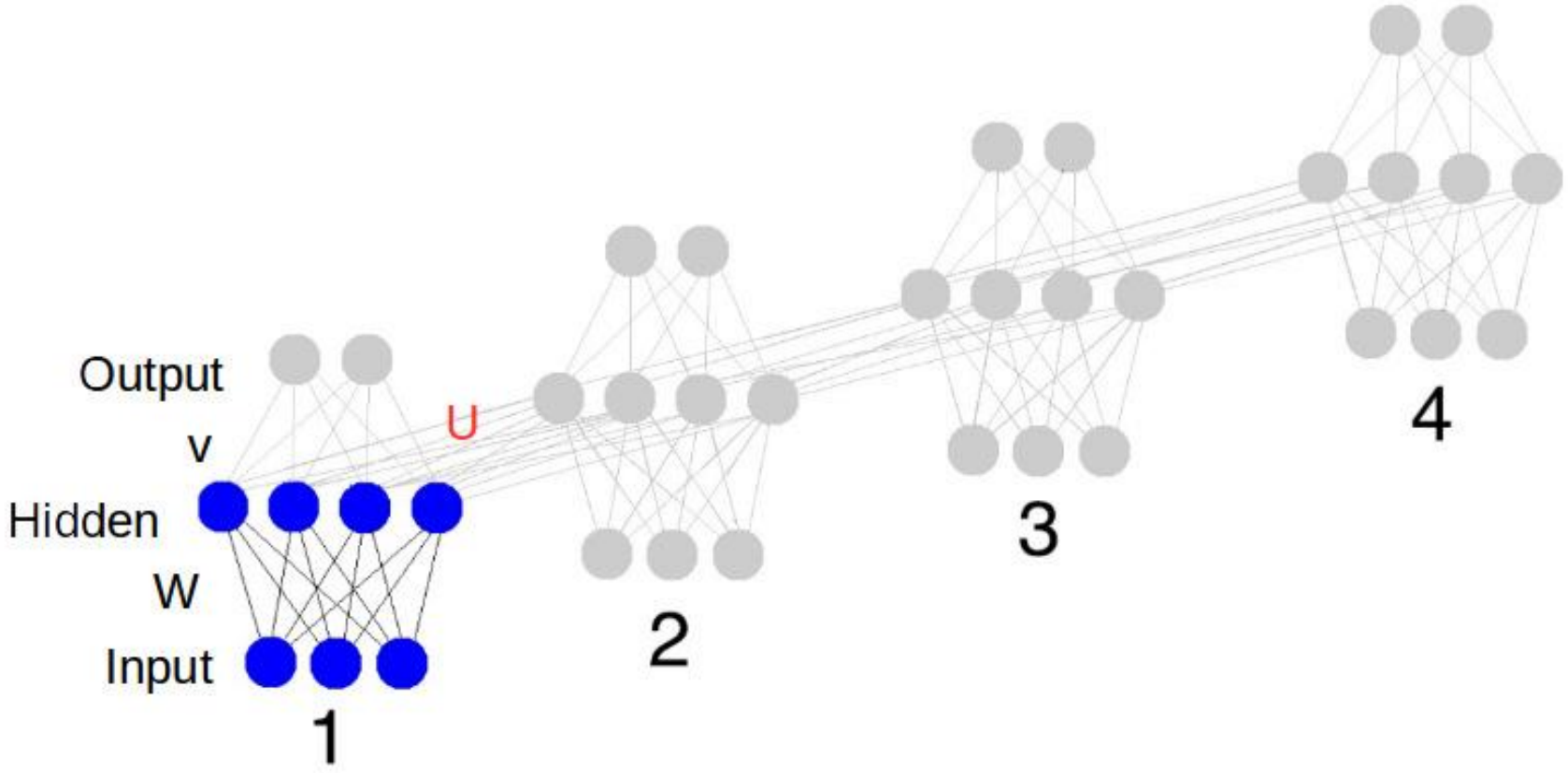
o depends on the nature of y_t. If it is categorical then o can be softmax

- Important: Same $\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{v}$ are used at all steps of the sequence (weight sharing)

Given in form of an embedding (e.g., word embedding if x_1 is a word)

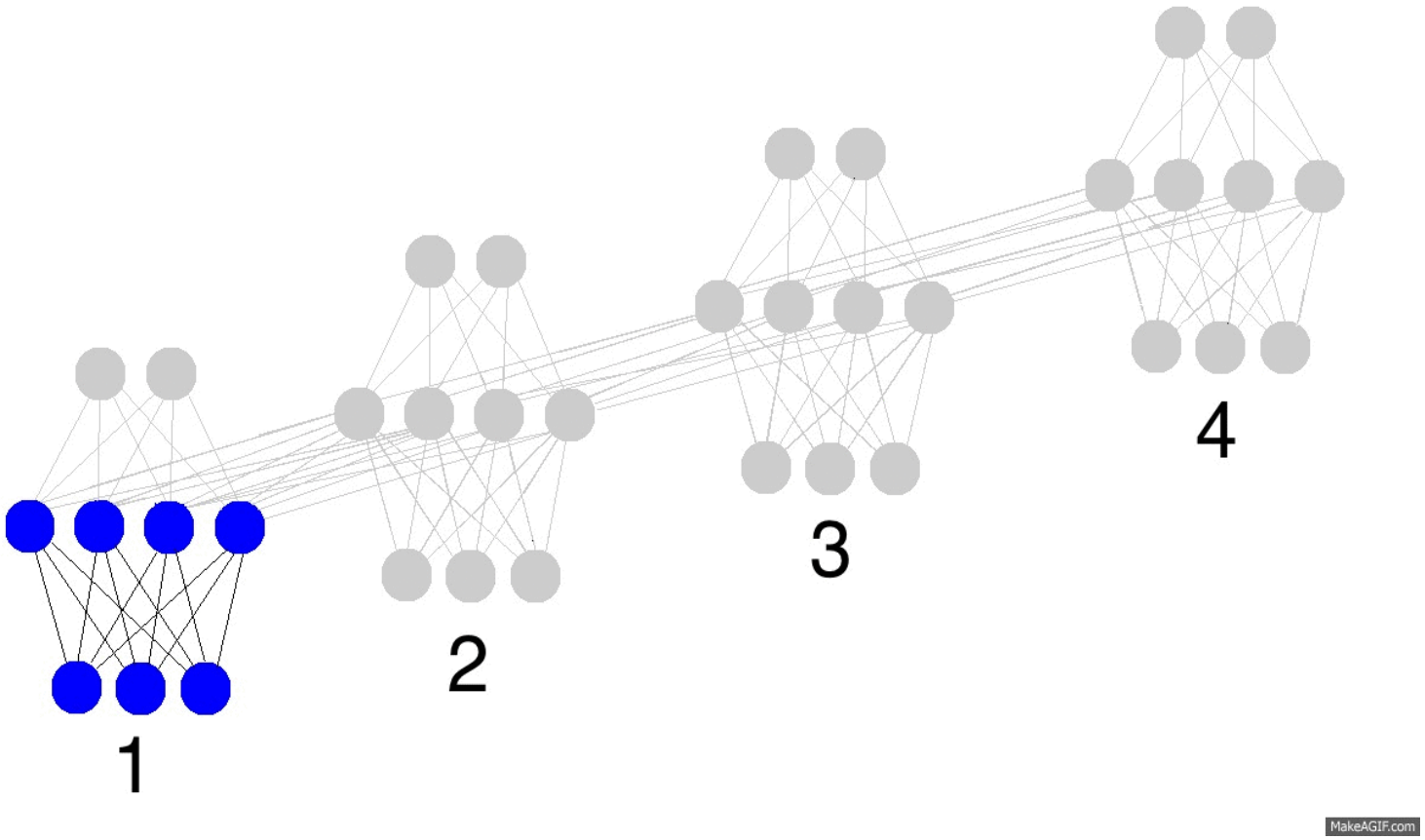
Recurrent Neural Nets (RNN)

- A more “micro” view of RNN (the transition matrix U connects the hidden states across observations, propagating information along the sequence)



Pic source: https://iamtrask.github.io/

RNN in Action



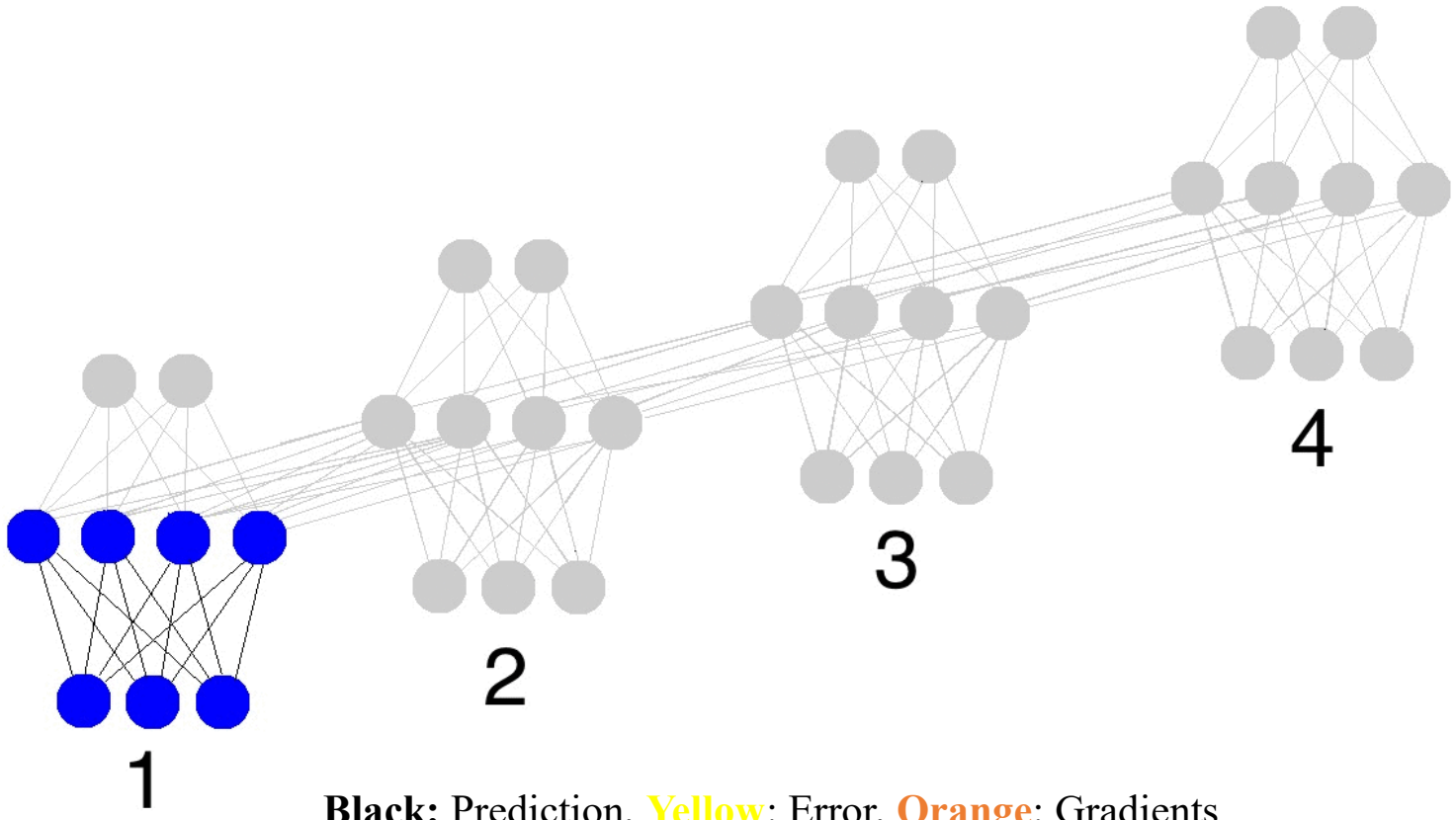
Pic source: https://iamtrask.github.io/

Workflow of RNN:

- The gif above reflects the magic of recurrent networks.
- It depicts 4 timesteps. The first is exclusively influenced by the input data.
- The second one is a mixture of the first and second inputs. This continues on.
- You should recognize that, in some way, network 4 is "full".
- Presumably, timestep 5 would have to choose which memories to keep and which ones to overwrite.
- This is very real. It's the notion of memory "capacity".
- As you might expect, bigger layers can hold more memories for a longer period of time.

Training RNN

- Trained using Backpropagation Through Time (forward propagate from step 1 to end, and then backward propagate from end to step)
- Think of the time-dimension as another hidden layer and then it is just like standard backpropagation for feedforward neural nets



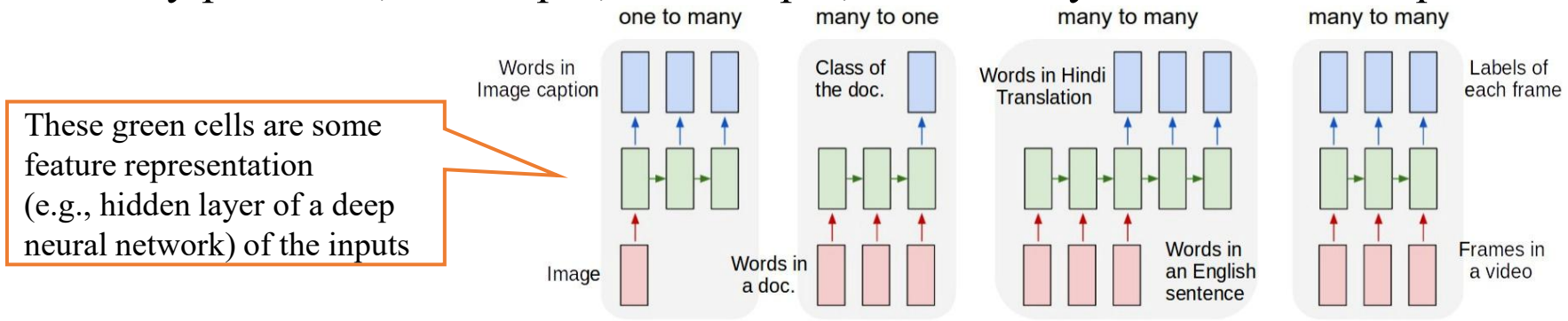
Black: Prediction, **Yellow**: Error, **Orange**: Gradients

- They learn by fully propagating forward from 1 to 4 (through an entire sequence of arbitrary length), and then backpropagating all the derivatives from 4 back to 1.
- You can also pretend that it's just a funny shaped normal neural network, except that we're re-using the same weights (synapses 0,1,and h) in their respective places.
- Other than that, it's normal backpropagation.

Pic source: https://iamtrask.github.io/

RNN Applications

- In many problems, each input, each output, or both may be in form of sequences

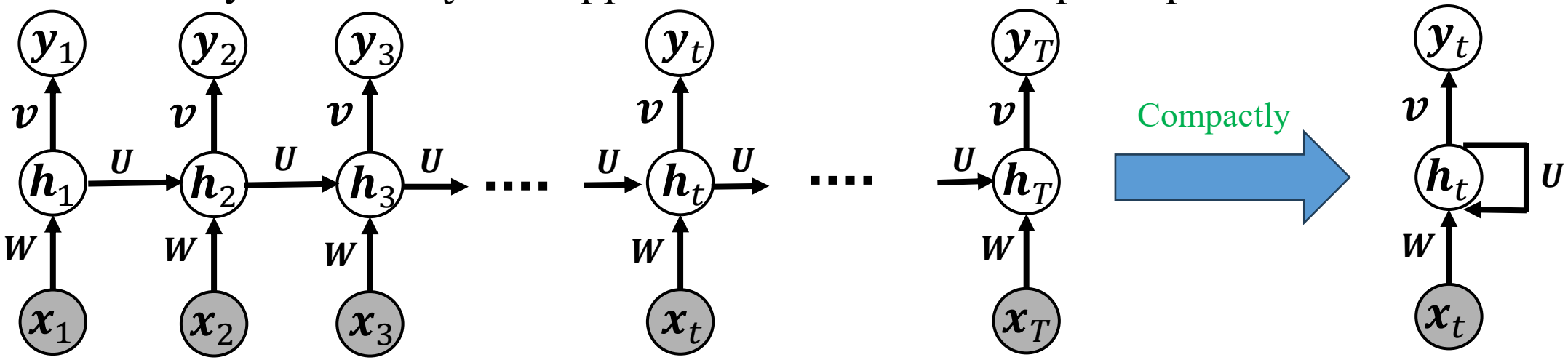


- Different inputs or outputs need not have the same length
- Some examples of prediction tasks in such problems
  - Image captioning: Input is image (not a sequence), output is the caption (word sequence)
  - Document classification: Input is a word sequence, output is a categorical label
  - Machine translation: Input is a word sequence, output is a word sequence (in different language)
  - Stock price prediction: Input is a sequence of stock prices, output is its predicted price tomorrow
  - No input – just output (e.g., generation of random but plausible-looking text)

RNNs: Long Distant Past is Hard to Remember

SORBONNE UNIVERSITÉ

- The hidden layer nodes h_t are supposed to summarize the past up to time $t-1$



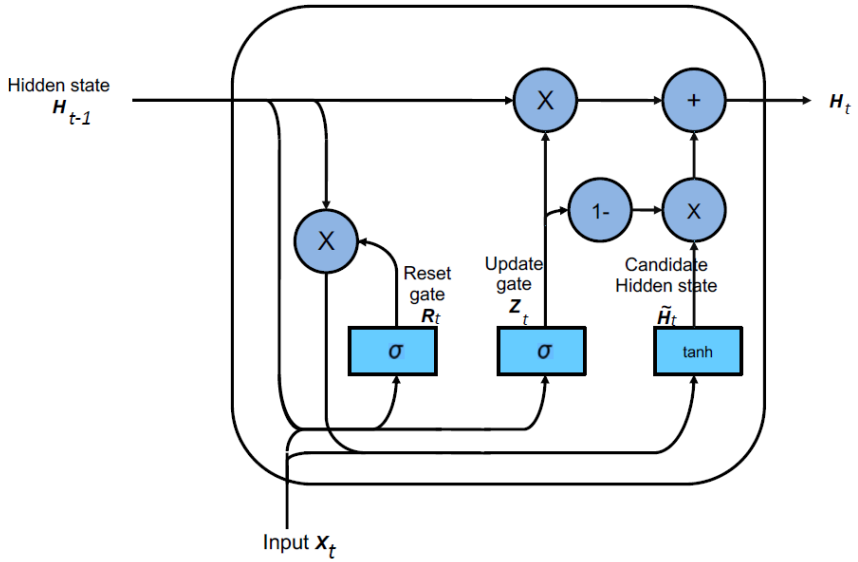
- In theory, they should. In practice, they can't. Some reasons
  - Vanishing gradients along the sequence too (due to repeated multiplications)
    – past knowledge gets "diluted"
  - Hidden nodes also have limited capacity because of their finite dimensionality
- Various extensions of RNNs have been proposed to address forgetting
  - Gated Recurrent Units (GRU), Long Short Term Memory (LSTM)

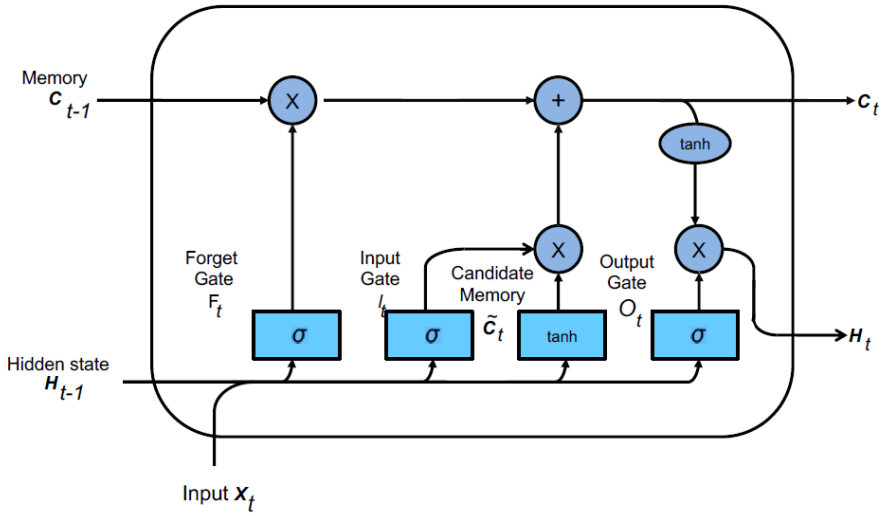
GRU and LSTM

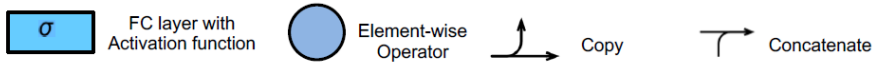
- Essentially an RNN, except that the hidden states are computed differently
- Recall that RNN computes the hidden states as

$$\boldsymbol{h}_t = tanh(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1})$$

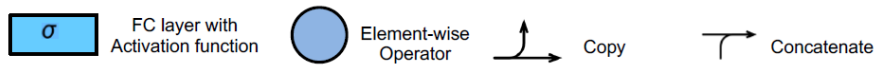
- For RNN: State update is multiplicative (weak memory and gradient issues)
- GRU and LSTM contain specialized units and "memory" which modulate what/how much information from the past to retain/forget







Pic source: https://d2l.ai/



LSTM

SORBONNE UNIVERSITÉ

- In contrast, LSTM maintains a "context" $\boldsymbol{C}_t$ and computes hidden states as

$$\hat{C}_t = \tanh(\mathbf{W}^c \boldsymbol{x}_t + \mathbf{U}^c \boldsymbol{h}_{t-1}) \quad \text{("local" context, only up to immediately preceding state)}$$

$$i_t = \sigma(\mathbf{W}^i \boldsymbol{x}_t + \mathbf{U}^i \boldsymbol{h}_{t-1}) \quad \text{(how much to take in the local context)}$$

$$f_t = \sigma(\mathbf{W}^f \boldsymbol{x}_t + \mathbf{U}^f \boldsymbol{h}_{t-1}) \quad \text{(how much to forget the previous context)}$$

$$o_t = \sigma(\mathbf{W}^o \boldsymbol{x}_t + \mathbf{U}^o \boldsymbol{h}_{t-1}) \quad \text{(how much to output)}$$

$$C_t = C_{t-1} \odot f_t + \hat{C}_t \odot i_t \quad \text{(a modulated additive update for context)}$$

$$h_t = \tanh(C_t) \odot o_t \quad \text{(transform context into state and selectively output)}$$

- Note: $\odot$ represents elementwise vector product. Also, state updates now additive, not multiplicative. Training using backpropagation through time.

- Many variants of LSTM exists, e.g., using $\boldsymbol{C}_t$ in local computations, Gated Recurrent Units (GRU), etc. Mostly minor variations of basic LSTM above.

Exercise: RNN

SORBONNE UNIVERSITÉ

Recurrent Neural Network (RNN)

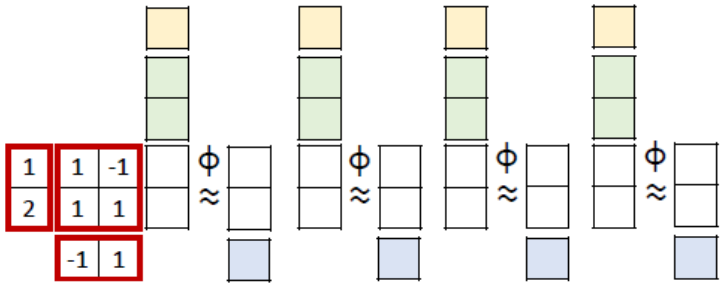
Input Sequence X | 3 | 4 | 5 | 6 |

Parameters A | 1 | -1 | / | 1 | 1 | B | 1 | / | 2 | C | -1 | 1 |

Activation Function ϕ: ReLU

Hidden States H_0 | 0 | / | 0 |

Output Sequence Y | | | | |



Recurrent Neural Network (RNN)

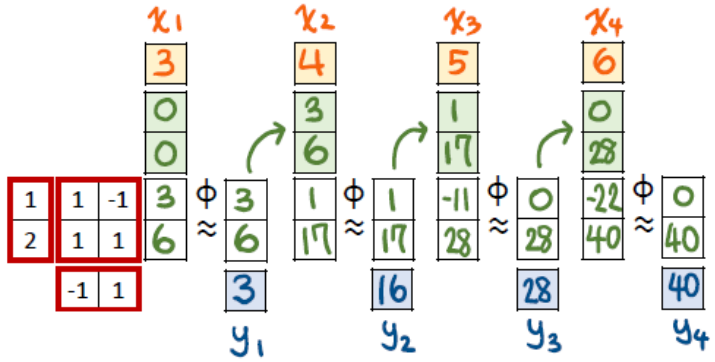
Input Sequence X | 3 | 4 | 5 | 6 |

Parameters A | 1 | -1 | / | 1 | 1 | B | 1 | / | 2 | C | -1 | 1 |

Activation Function ϕ: ReLU

Hidden States H_0 | 0 | / | 0 |

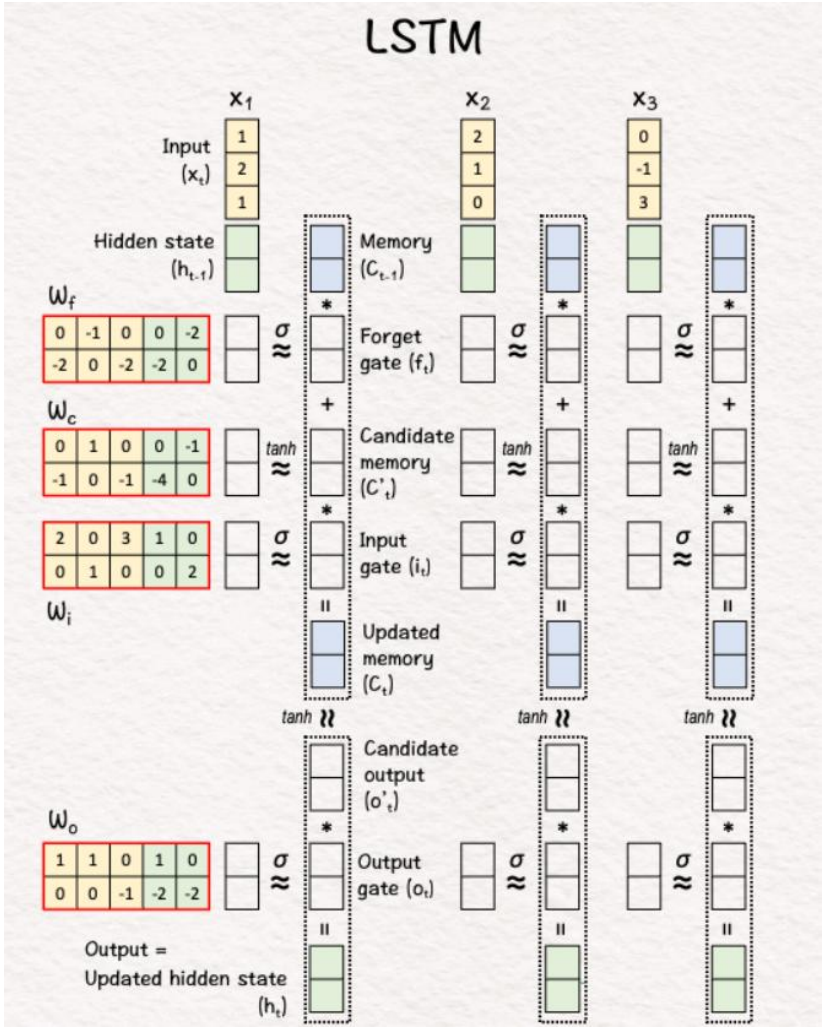
Output Sequence Y | | | | |

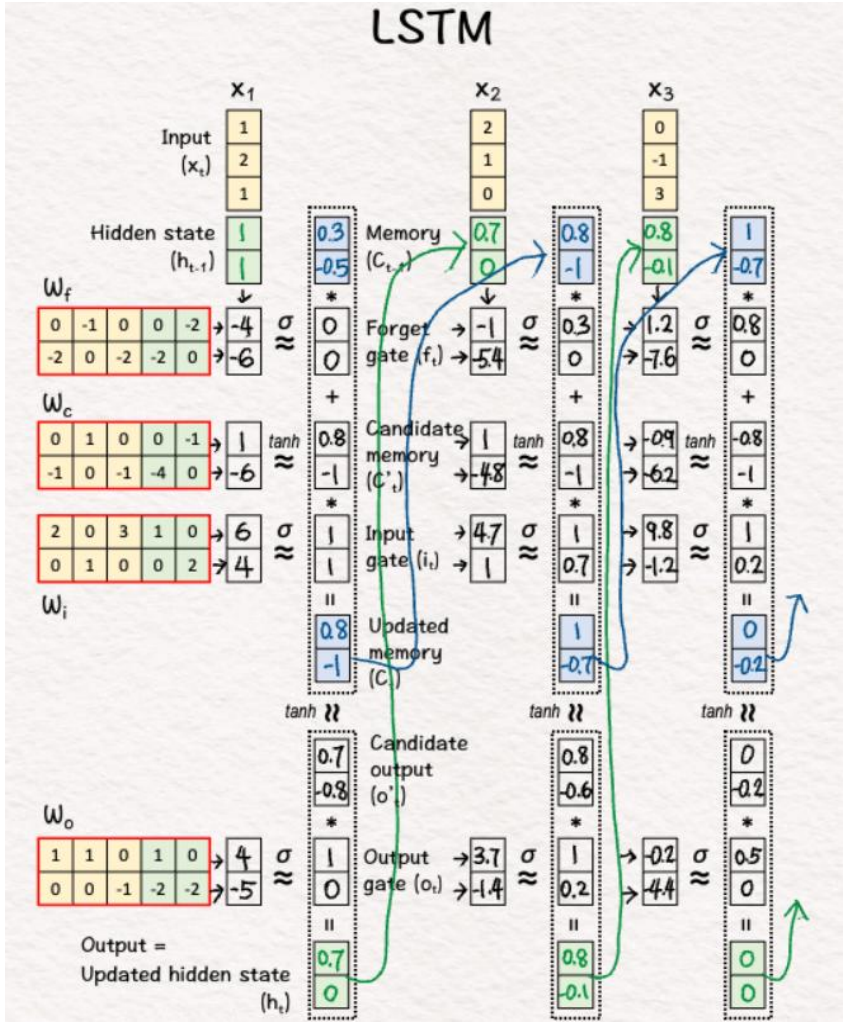


Pic source: https://www.byhand.ai/

Exercise: LSTM

Initialize: Randomly set the previous hidden state h0 to [1, 1] and memory cells C0 to [0.3, -0.5]





Pic source: https://www.byhand.ai/

SOTA Forecasting Methods

SORBONNE UNIVERSITÉ

Pretrained Models

Single model used across all tasks

- LLMTime
- ForecastPFN
- Lag-Llama
- Monai
- Chronos

Task-specific Models

Separate model fine-tuned for each task

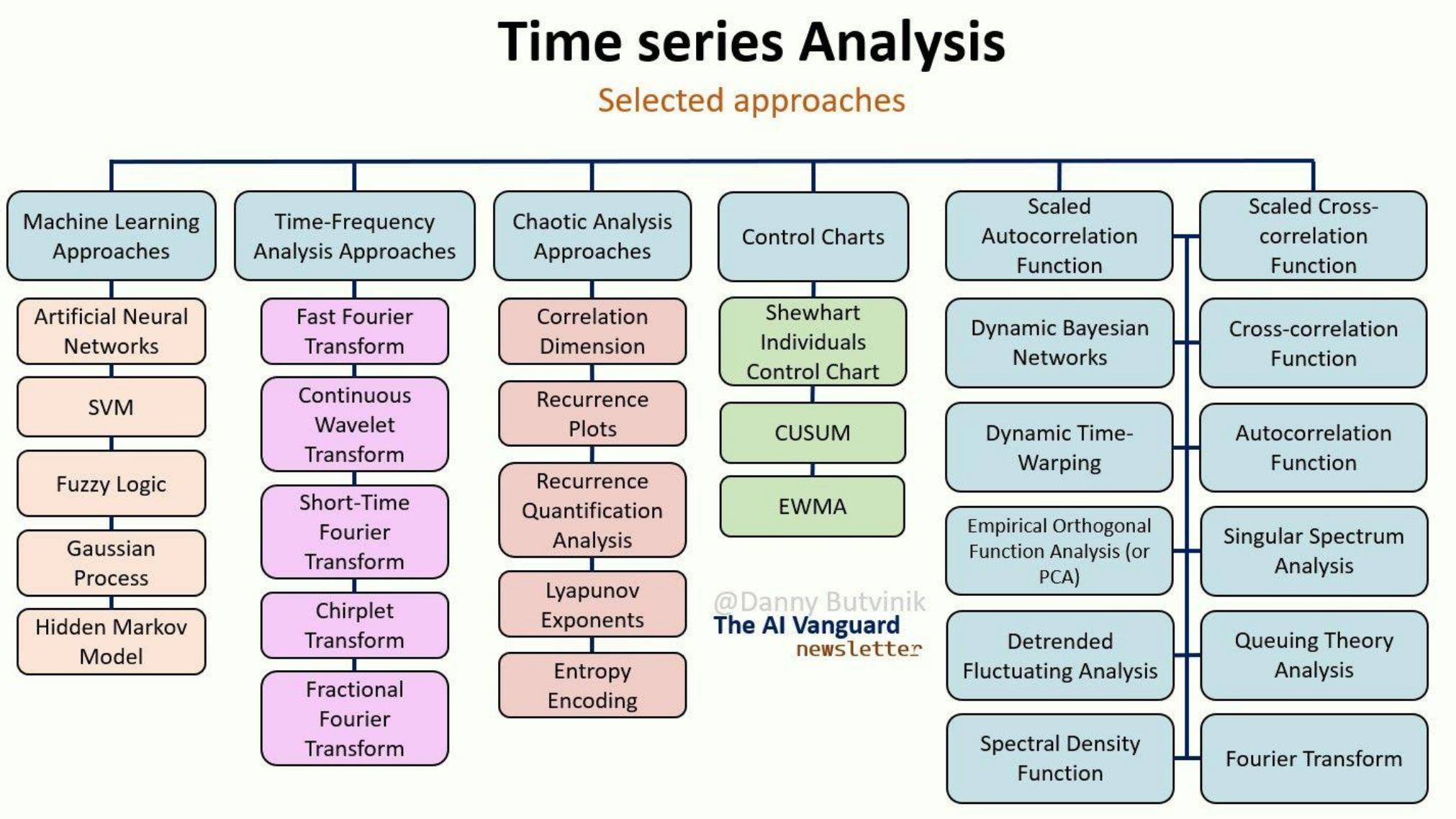
- PatchTST
- DeepAR
- EWNet
- TFT
- DLinear
- NBEATS
- NHITS
- CatBoost

Local Models

Separate model for each time series

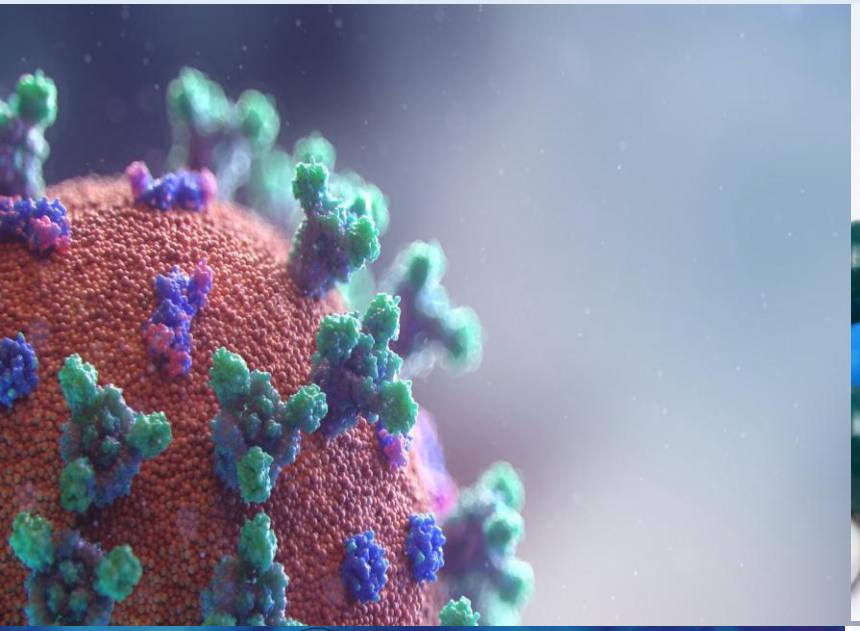
- Naïve
- ETS
- ARIMA

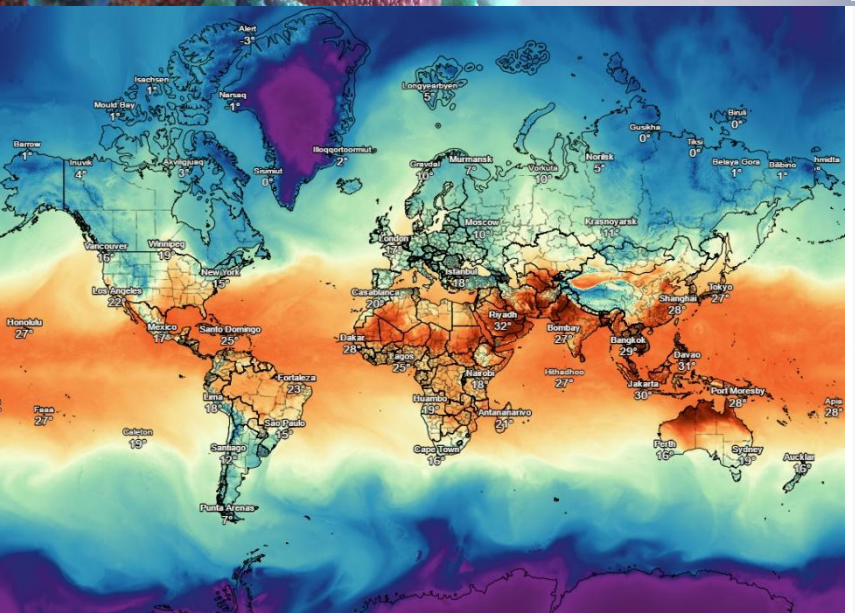
Time Series Approaches

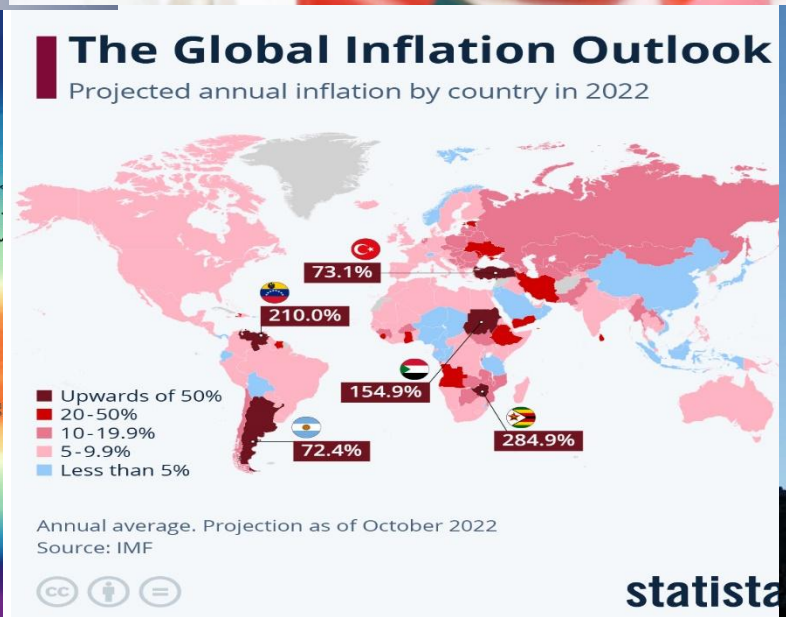


What can we forecast?

SORBONNE UNIVERSITÉ







Reference

HAPPY FORECASTING

"A good forecaster is not smarter than everyone else, he merely has his ignorance better organised."