Recurrent Neural Network

- Designed for processing sequential data
- output of node depends on current input & all previous outputs (Implicitly all previous inputs)
- useful for tasks requiring context / temporal dependence → Time Series forecasting / Language Modelling
- Defining feature = hidden state / memory → capture temporal r/s in sequential data.
- Eg Input: stock mkt prices collected over time

  └ yesterday price y_0

  Today price y_1

  Tmr price TBP

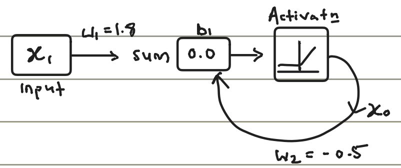
- Foundation for Long Short-Term Memory Networks & Transformers.

- Fundamental for NLP, Time Series Analysis & Computer Vision

How RNN works

Suppose we know x_0 & x_1 & model takes in single input

1) Input x_0 as input
2) At output of hidden layer → output layer as prediction / → feed back loop
3) Go through feedback loop
4) pass in x_1 as next input & sum



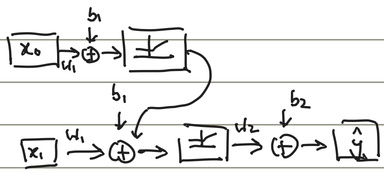
$$(x_0 \cdot w_1 + b_1) + (x_1 \cdot w_1 + b_1)$$

Unrolling of RNN as conceptualization

- Does not create multiple physical copies of network
- Same weights & biases used across all networks
- No. of "copies" dependent on how many data points needed for prediction

  ↳ eg $\underbrace{\text{yesterday}}_{1} + \underbrace{\text{today}}_{1}$ to predict tomorrow

  1 + 1 = 2 "copies"



- Weights & bias remains similar.
- More unrolling = harder to train (vanishing / Exploding gradient)

  └ if weight x2 & memory = 50, output = 2^{50} → ∞

  Sol^n: limit weights to below 1.

  Problem: vanishing gradient

  if weight x0.5 & memory = 50, output = 0.5^{50} → 0

  Solution: Long Short-term memory networks.

RNN Math

$$\hat{y}_t = \underbrace{W_o}_{\text{output weight}} \cdot h_t + \underbrace{\beta_o}_{\text{Output bias}}$$

$$h_t = \sigma(\underbrace{W_h}_{\text{Weight of hidden layer}} \cdot \underbrace{h_{t-1}}_{\text{hidden state from previous time step}} + \underbrace{W_x}_{\text{Weight of Input layer}} \cdot \underbrace{x_t}_{\text{Input}} + \underbrace{\beta_h}_{\text{Bias of hidden layer}})$$

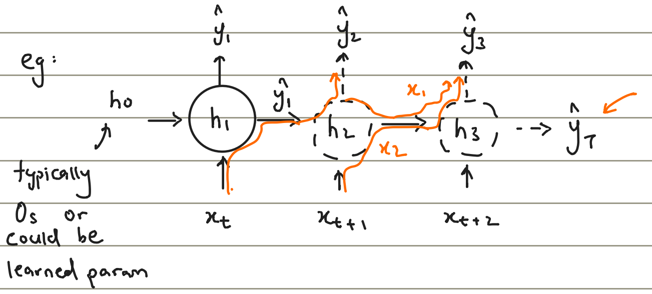
$$h_t = \sigma(W_h \cdot h_o + W_x \cdot x_t + \beta_h)$$
$$\hat{y}_t = W_o h_t + \beta_o$$

$$h_{t+1} = \sigma(W_h \cdot h_t + W_x x_{t+1} + \beta_h)$$
$$\hat{y}_{t+1} = W_o h_{t+1} + \beta_o$$
$$\vdots$$

Hidden state h_t

- captures memory of sequence up to that point
- Prediction made with current input & past inputs

eg:



Final prediction. Intermittent predictions like $\hat{y}_1, \hat{y}_2 \dots$ may or may not be kept depending on tasks.

$$p(\hat{y}_1 = k \mid x_1)$$
$$p(\hat{y}_2 = k \mid x_1, x_2)$$
$$p(\hat{y}_T = k \mid x_1, \dots x_T)$$

Hidden Layer / Hidden State of RNN

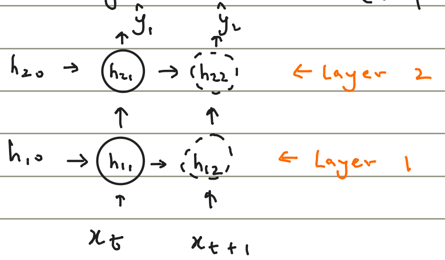
1) Single Hidden Layer

↳ hidden state updated at each time step based on $\underbrace{\text{Input}}_{x_i}$ & $\underbrace{\text{previous hidden state}}_{h_{t-1}}$

- common for time-series prediction or sequence classification

time-series prediction	sequence classification
· stock price forecast	· sentiment analysis
· Economic Indicators	· speech recognition
· Demand forecasting	· NLP.

2) Multi Layer Hidden state (Deep RNN)



Allows model to learn more abstract and hierachical features of sequence data

Input Dimensions for RNN

$N \times T \times D$ (samples × time steps × feature dimensions)

(T: Sampling frequency. D: No. features.)

Stock Price eg:
- N = 30 companies
- T = 30 days of price
- D = 4 features (OHLC)
- $30 \times 30 \times 4$

Healthcare:
- N = 100 patients
- T = 24 hours (data / hour)
- D = 3 features (heart rate, blood Pressure, Temp.)
- $100 \times 24 \times 3$.

* In TensorFlow, T is constant size (pad if needed) Input shape = $(T \times D)$

M: No. of hidden units

K: No. of output units : Regression can also be multidimensional

Simple RNN in Tensor Flow

input = Input (shape = (T, D)) * For Regression task, don't need activation fn

x = SimpleRNN (<No_hidden_nodes>) (input)

x = Dense (<No_output_nodes>) (x)

model = Model (input, x)

output tensor

Input tensor → shape & type of Input

- Keras functional API
- specify inputs & outputs in NN

model.summary() : See all layers & relevant details

model.layers [<layer_no.>].get_weights()

↳ print shape, can <u>unpack</u> as needed.

L 3 layers = 3 to unpack.

Input Dim - D x H - Hidden Dim

H x H

H x O - output Dim