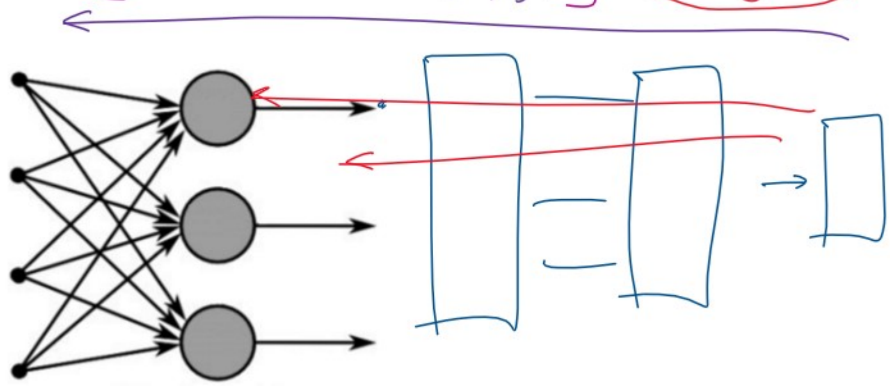
RNN Architecture

26 October 2024 19:59

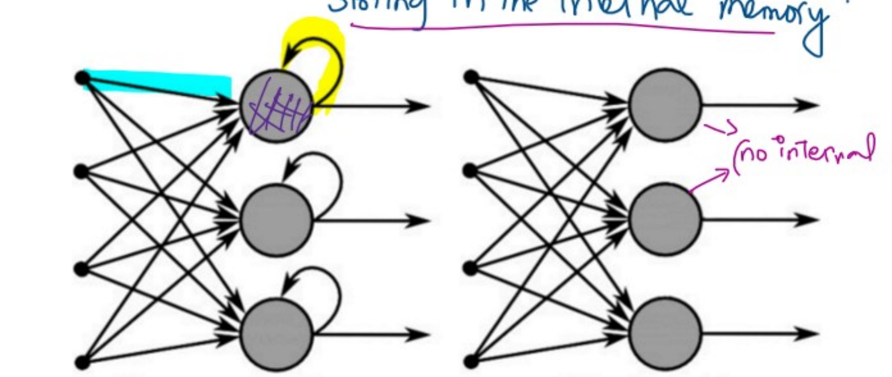
[forward + backward propagation] → Training phase



Feed-Forward Neural Network

Forward Propagation → Prediction phase

storing in the internal memory.



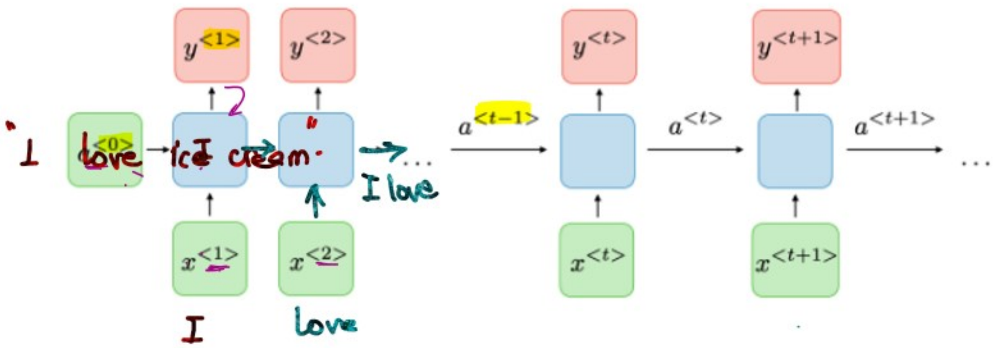
(no internal memory)

Recurrent Neural Network

Feed-Forward Neural Network

[stock prices] → RNN

RNNs allow previous outputs to be used as inputs while having hidden states.



Here,

① $x^{<t>}$ represents sequence of inputs

Green boxes at the bottom denote the input sequences laleled as $x^{<1>}$, $x^{<2>}$ ...... $x^{<t>}$

Note: input is a sentence then each $x^{<t>}$ could represent a word or character.

② Hidden states $a^{<t>}$

Blue rectangles in the middle represent the hidden states. These hidden states maintain the memory of the network. Each hidden state $a^{<t>}$ carries the information from the past inputs upto the current state 't'

In a basic RNN setup, the hidden state $a^{<t>}$ is computed using the current input $x^{<t>}$ and previous hidden state $a^{<t-1>}$

③ Outputs $y^{<t>}$

Red boxes at the top denote the outputs of the RNN at each time step; labeled as $y^{<1>}$, $y^{<2>}$, $y^{<3>}$ ... and so on.

1. Processing the first word: I

* RNN starts with an initial hidden state, usualy initialized to zeros or random values.

* RNN starts with an initial hidden state, usually initialized to zeros or random values.
* the first input word 'I' is passed into the network.
* RNN processes the word 'I' and updates its hidden state $a^{<1>}$ or $h^{<1>}$

2. Processing the second word: "love"

* second word "love" is passed into RNN along with the updated hidden state $h^{<1>}$ or $a^{<1>}$
* RNN uses the hidden state $h^{<1>}$ or $a^{<1>}$ along with the new word 'love' to produce a new hidden state $h^{<2>}$ or $a^{<2>}$

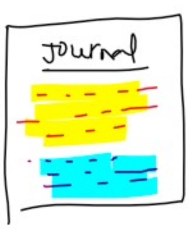
'i love'

3. Processing the third word: "ice"

$h^{<3>}$ or $a^{<3>}$ – i love ice

4. Processing the fourth word: "cream

$h^{<4>}$ or $a^{<4>}$: i love ice cream.



sentence #1 I walked into the bank → Financial Institution

sentence #2 I walked on the bank of the Ganges.

(side of the river)

Processing the word 'I' → Pronoun

Transforms the word 'I' into an embedding vector or numerical representation.

It captures the semantic meaning of the word 'I'.

semantic refers to the meaning and interpretation of the words, phrases, sentences and symbols. keeping the context

Semantic refers to the meaning and interpretation of the words, phrases, sentences and symbols. keeping the context

→ Brick in the wall 

→ we are just brick in the wall.

Note: once the processing the word 'I' is complete. The hidden state $h^{<1>}$ or $a^{<1>}$ now holds the information about the word 'I' → like recognizing it as a pronoun.

↓

[subject of a sentence]

In essence, $h^{<1>}$ or $a^{<1>}$ is now aware that the sentence has a subject.

$$h^{<1>} \text{ or } a^{<1>} = \tanh(W_{hx} \cdot x^{<1>} + W_{hh} \cdot h^{<0>} + b_h)$$

- W_{hx} : weight matrix that connects the input to hidden state
- W_{hh} : weight matrix that connects the previous hidden state to the current hidden state
- b_h : bias term
- tanh : activation function

Processing the word Love"

When the RNN receives the second word "love". It transforms it into an embedding vector $x^{<2>}$ which captures the meaning of love.

It also updates the hidden state by combining $x^{<2>}$ with the hidden state $h^{<1>}$ or $a^{<1>}$ (previous) which already holds information about 'I'

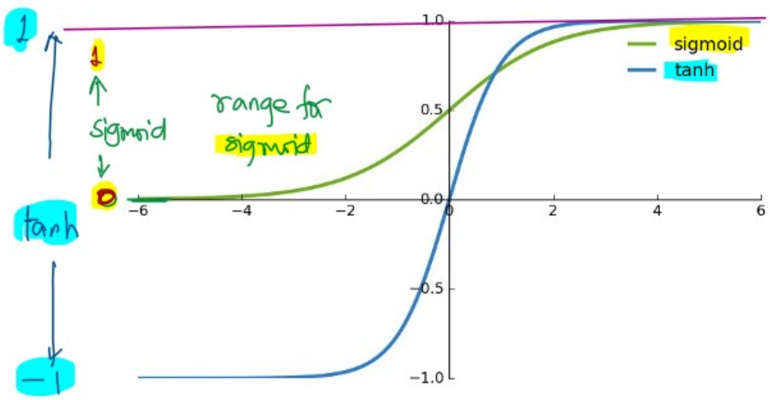
$$h^{<2>} \text{ or } a^{<2>} = \tanh(W_{hx} \cdot x^{<2>} + W_{hh} \cdot h^{<1>} + b_h)$$

Why do we use **tanh** as activation function?

tanh function ranges (y values) from −1 to 1 which means it tanh function ranges (y values) from -1 to 1 which means it produces both negative and positive values which is advantageous as it centers the data around zero

↓

making optimization and convergence faster



Note: RNNs often use tanh as the activation function as they need to capture both positive and negative relationships in sequential data.

I love Ice-cream

I donot like Ice cream

Target: To build a small scale next word prediction model — only focussed on our example:

I love ice → cream

Recurrent Neural Networks (RNNs) Key Features

– how is it different from other neural networks

* Internal Memory: RNNs have an internal state that allows them to retain and leverage information from previous inputs — which is crucial for tasks where context matters.

- **Sequential data Handling:** RNNs are specially designed for working with sequences, making them ideal for applicaltons like speech recognition, language modeling, and time-series analysis.

- **Contextual Understanding:** RNNs can interpret new data in the context of prior information, which is essential for understanding the meaning and the flow in natural language tasks

- **Dynamic Adaption:** RNNs continuously update internal states, allowing them to adjust to evolve patterns within sequences.