26/7/24

COURSE 5

Examples of sequence data:

→ Speech recognition
→ music generation
→ sentiment classification
→ DNA seq. analysis
→ Video Activity recognition
→ Name entity recognition

Both I/O are seq.
Only input or output is seq.

Example: Name Entity Recognition

x: Harry Potter & Hermoine Granger invented a new spell

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ..... $x^{<t>}$ ..... $x^{<9>}$

y: 1 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ ..... $y^{<t>}$ ..... $y^{<9>}$

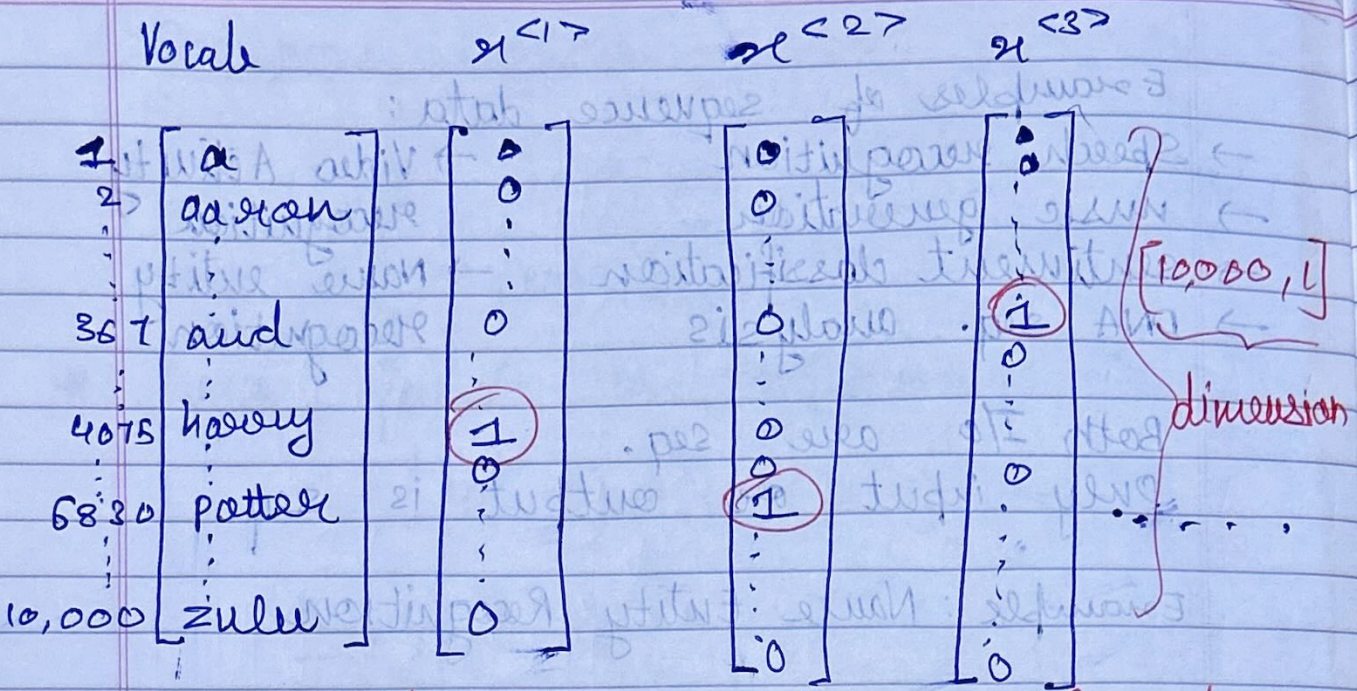
$X^{(i)<t>}$ ⇒ t^{th} element of i^{th} training eg.

$T_x^{(i)} = 9$ (here) ⇒ length of i^{th} example

$T_x = T_y$ (here)

* One - Hot Notation:

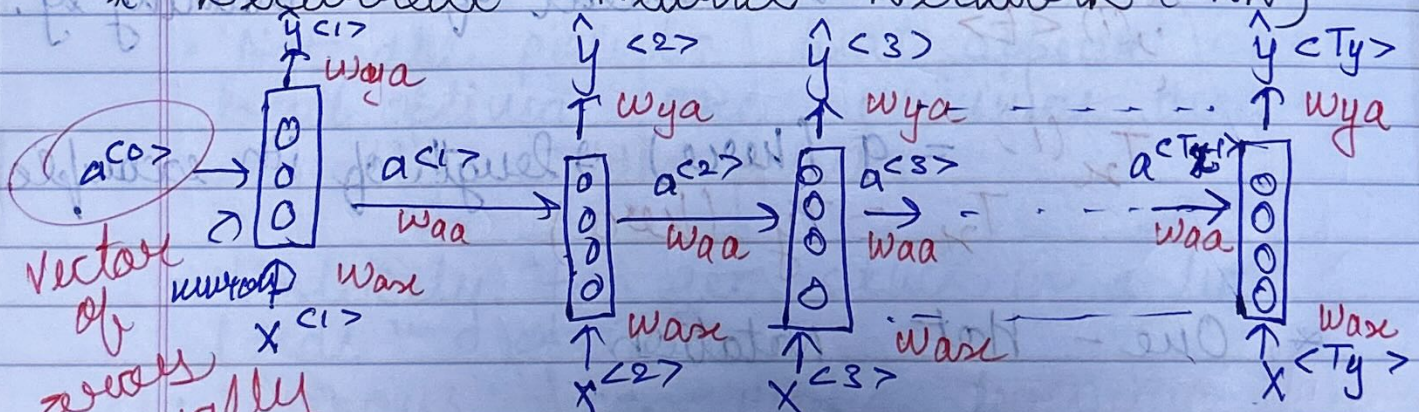
We need a vocabulary for the tasks like above, the size of which is generally 30,000 or 50,000.
We'll use 10,000 here for learning purpose



One-hot, since only one element is 1

* Why not a Standard Network?
  - I/O can be different lengths in different examples.
  - Doesn't share features learned across different positions of text.

* Recurrent Neural Network (RNN)



- Scans data from left to right & the parameters are shared.

- Disadvantage : Considers data only previous to the current element.

Eg:

He said, "Teddy Roosevelt was a President"
He said, "Teddy Bears are on sale!"

Here, based off first 3 words cannot decide what Teddy is.

- Forward Propagation:

$a^{<0>} = \vec{0}$

$a^{<1>} = g(W_{aa} a^{<0>} + W_{ax} x^{<1>} + b_a)$

$\hat{y}^{<1>} = g(W_{ya} a^{<1>} + b_y)$

→ tanh/ReLU

$a^{<t>} = g(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a)$

$\hat{y}^{<t>} = g(W_{ya} a^{<t>} + b_y)$

if classifier, sigmoid

multiplied by a to compute y.

$a^{<t>} = g(W_a [a^{(t-1)}, x^{<t>}] + b_a)$

- compressing a 2 parameter matrix to a 1 parameter matrix.

$W_a = [W_{aa} : W_{ax}]$ (100, 10100)

100 | 10,000

$[a^{<t-1>}, x^{<t>}]$

$a^{<t-1>}$ (100)

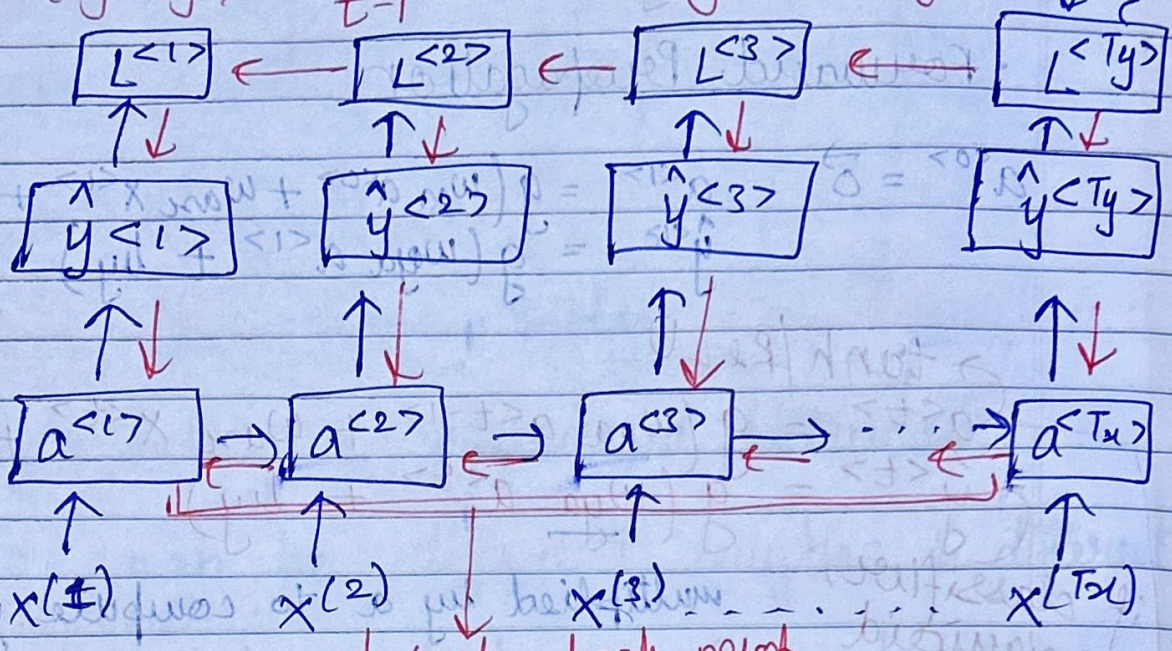
$x^{<t>}$ (10,000)

(10,100, 100)

$[W_{aa} : W_{ax}]_{(1,2)} \begin{bmatrix} a^{(t-1)} \\ x^{(t)} \end{bmatrix}_{(2,1)} = W_{aa} a^{(t-1)}_{(1,1)} + W_{ax} x^{(t)}_{(1,1)}$

$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{(t)} \log \hat{y}^{(t)} - (1 - y^{<t>}) \log (1 - \hat{y}^{<t>})$$

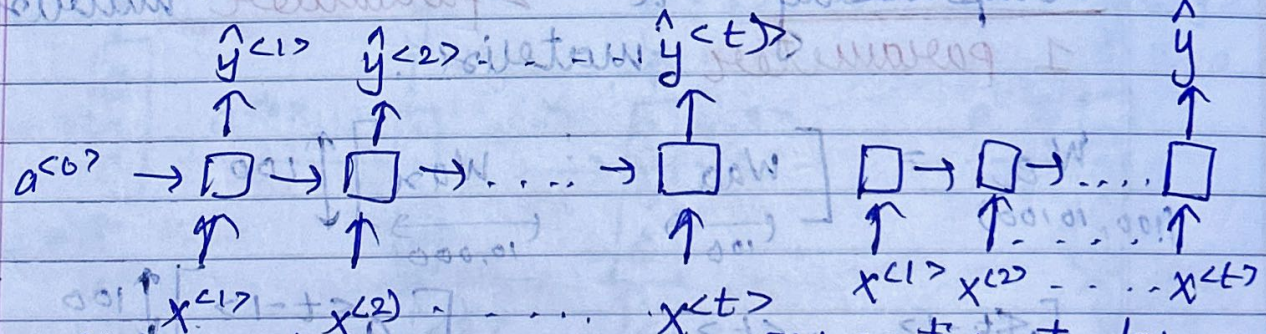
$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$



most imp back prop

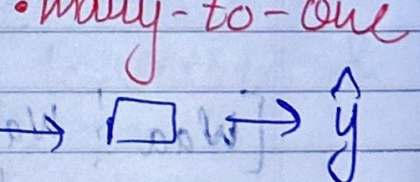
Backpropagation through time.

* Examples of RNN architecture:



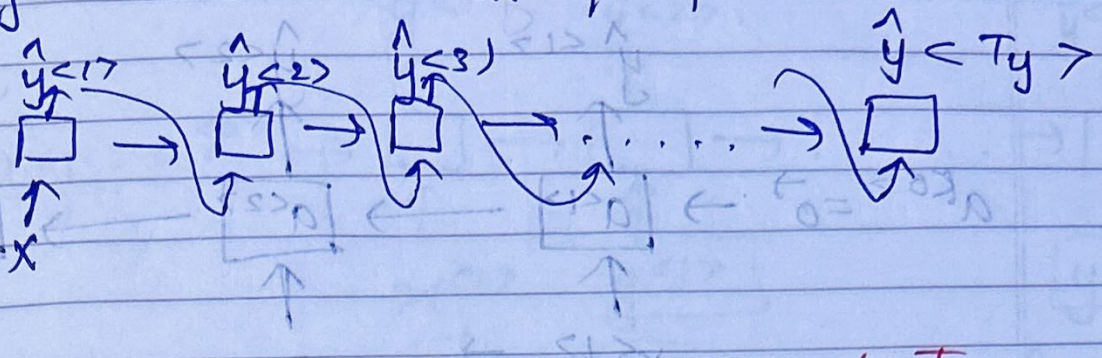
Eg: Name entity recognition
• many to many
$T_x = T_y$
• one to one

Eg: sentiment class.
• many-to-one



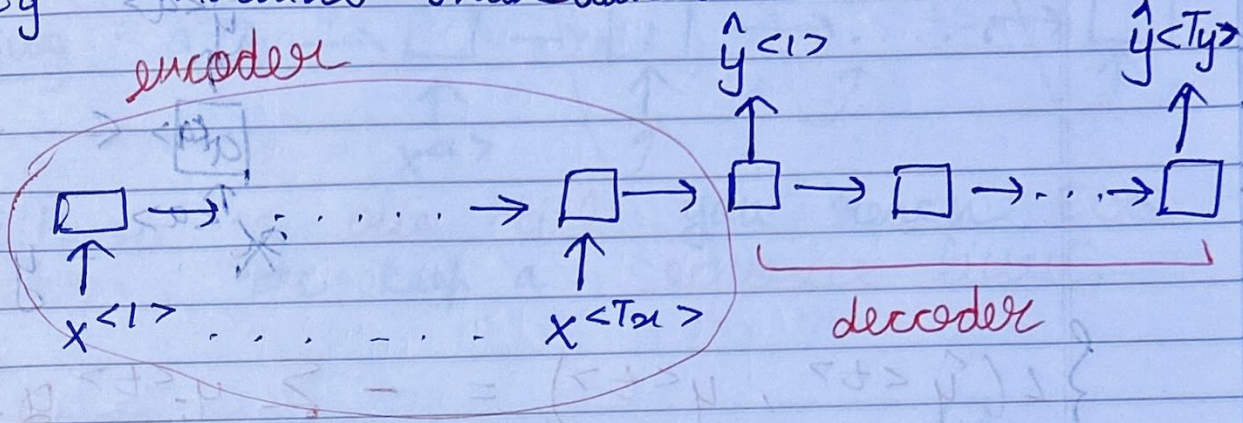
- one-to-many

Eg: Music Generation / Seq Generation



- many-to-many $T_x \neq T_y$

Eg: Machine Translation



* Language Modelling with an RNN:

1. Training Set: large corpus of english text.

→ Tokenise (using one-hot or other)

Cats average 15 hours of sleep a day. <EOS>

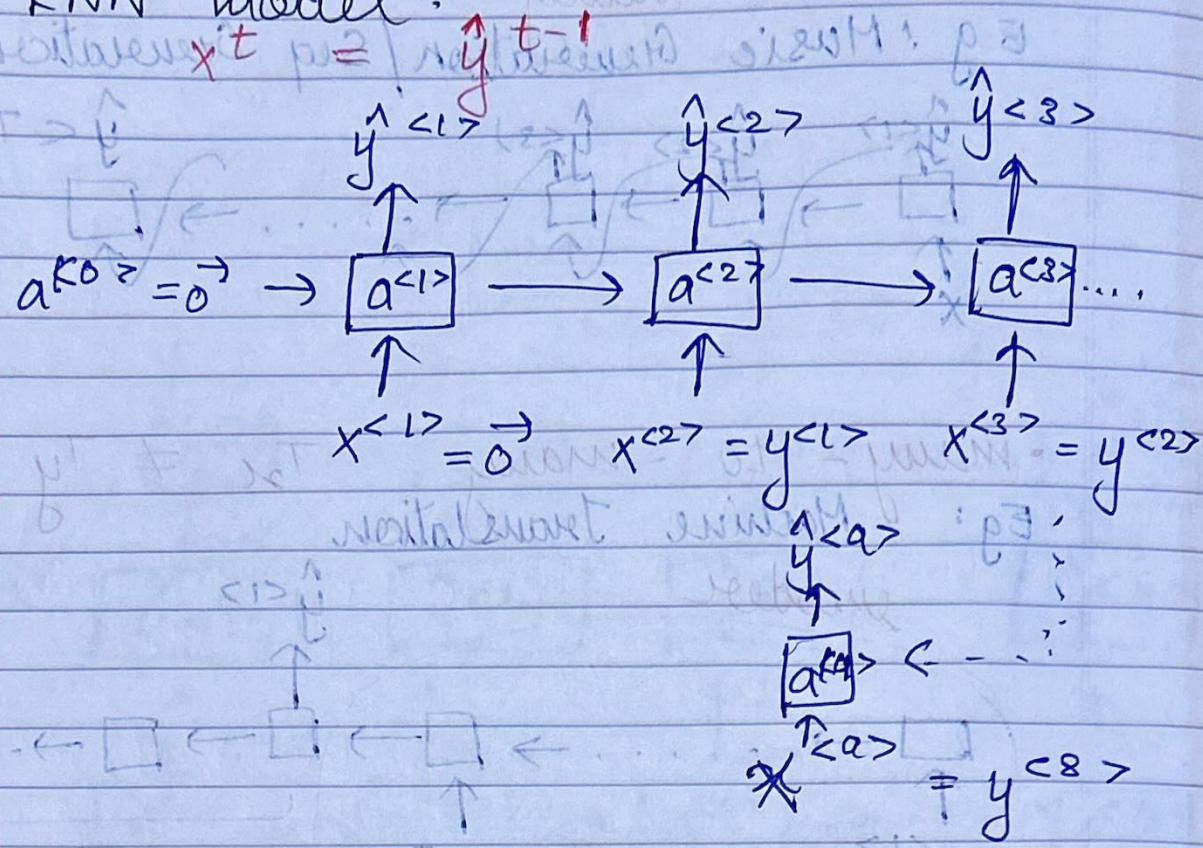
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<8>}$ $y^{<9>}$

for word not in our dictionary, we give a unique token.

The Egyptian Mau is a breed of cat. <EOS>

Mau: not in our dict → <UNK> unknown word

RNN model:



softmax loss

$$L(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

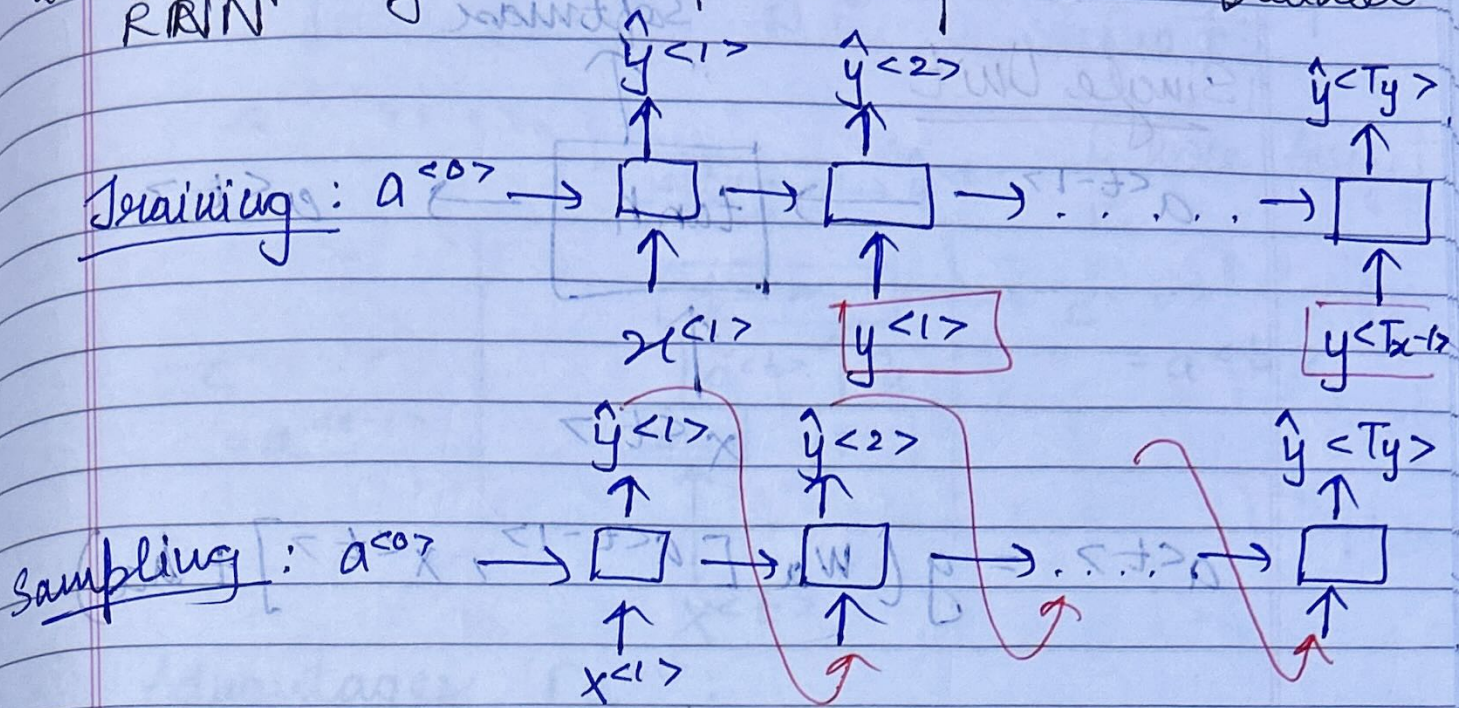
$$L = \sum_t L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

softmax probability

$$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>}) \, P(y^{<2>} | y^{<1>}) \, P(y^{<3>} | y^{<1>}, y^{<2>})$$

- We continue with this process till we reach the EOS token.

* Sampling a sequence from a trained RNN



Either go out till you reach EOS token, or keep a sequence limit.

* Character-level language model:

Vocabulary: [a, b, ..., z, ␣, ., ,, ..., 9, A, ..., Z]

pros: no worry about unknown word tokens

cons: can end up with large sequence & it is also computationally expensive

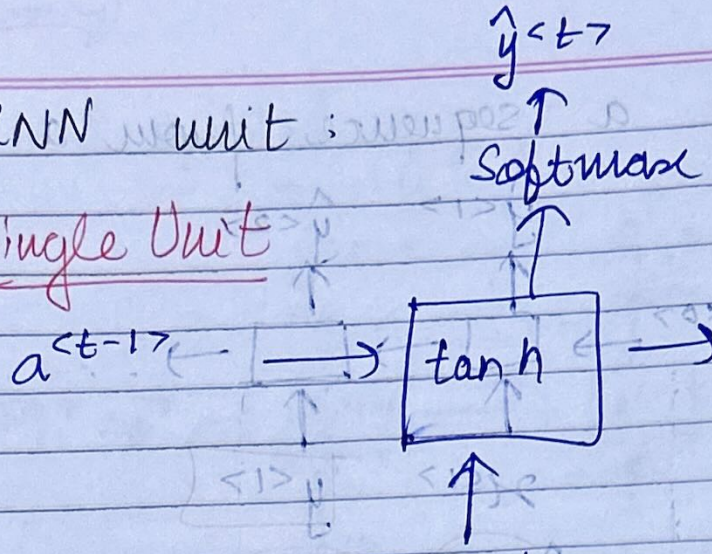
- used only in more specialised applications

* Vanishing gradients is a problem to be dealt with in RNN

** Exploding gradient (not the problem here) ** → apply Gradient Clipping

* RNN unit :

Single Unit



$$a^{<t>} = g(W_a [a^{<t-1>}, x^{<t>}] + b_a)$$

* GRU (simplified)

c = memory cell. $c^{<t>} = a^{<t>}$ (in RNN)

$$\tilde{c}^{<t>} = \tanh(W_c [a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u [a^{<t-1>}, x^{<t>}] + b_u)$$

uppercase gamma

update → gamma u, always lies b/w 0 & 1

→ gamma for gate

→ the above func^n decides whether or not to update the gate value at the appropriate time.

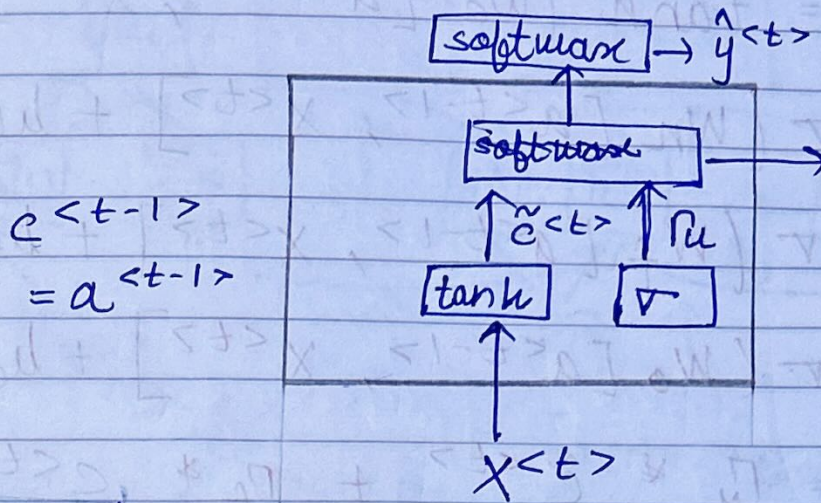
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

The cat, which already ate..... was full.

$\Gamma_u = 1$ $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 0$ ... $\Gamma_u = 0$ $= 1$.

$c^{<t>} = 1$

& were forget it.



Advantages Γ_u :

→ can be very small, thus $c^{<t>} = c^{<t-1>}$

→ Γ_u can be -ve, as long as it is -ve, by round off the update gate will be zero.

$c^{<t>}$, $\tilde{c}^{<t>}$, $\Gamma_u \Rightarrow$ same dimension

* Full GRU :

$$\tilde{c}^{<t>} = \tanh\left(W_c\left[\Gamma_r * c^{<t-1>}, x^{<t>}\right] + b_c\right)$$

$$\Gamma_u = \sigma\left(W_u\left[c^{<t-1>}, x^{<t>}\right] + b_u\right)$$

$$\Gamma_r = \sigma\left(W_r\left[c^{<t-1>}, x^{<t>}\right] + b_r\right)$$

(relevance gate)

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

LSTMs

→ wiser & faster versions of LSTM.

$$\tilde{c}^{<t>} = \tanh\left(W_c\left[a^{<t-1>}, x^{<t>}\right] + b_c\right)$$

update $\Gamma_u = \sigma\left(W_u\left[a^{<t-1>}, x^{<t>}\right] + b_u\right)$

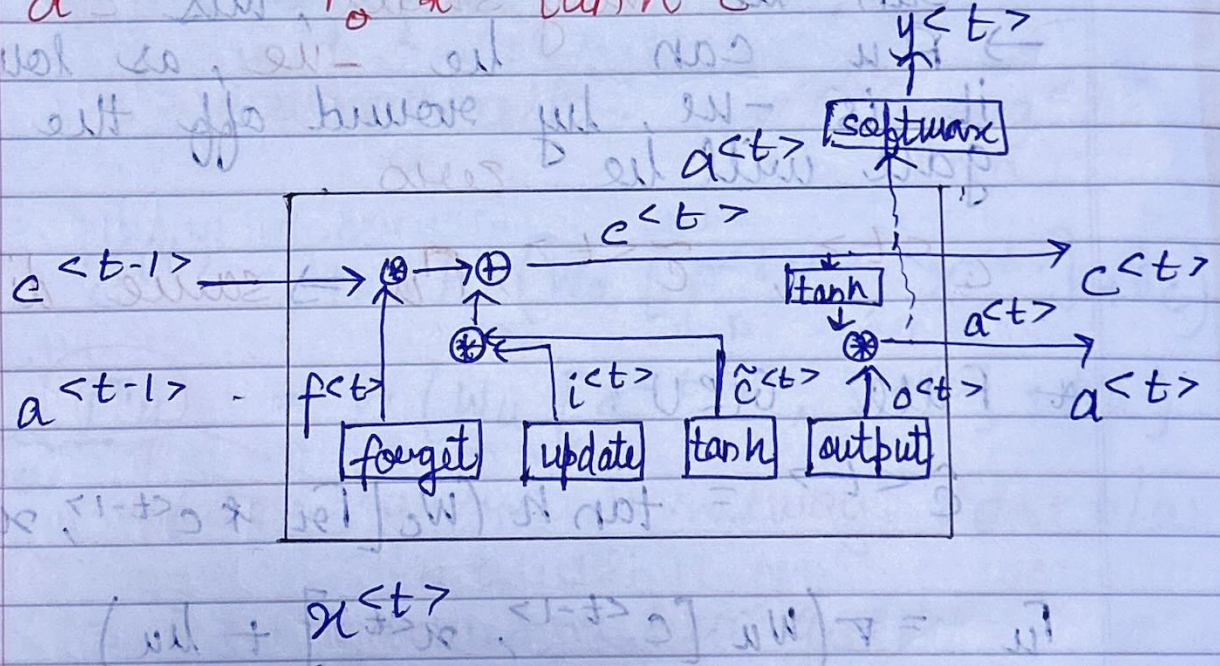
forget $\Gamma_f = \sigma\left(W_f\left[a^{<t-1>}, x^{<t>}\right] + b_f\right)$

output $\Gamma_o = \sigma\left(W_o\left[a^{<t-1>}, x^{<t>}\right] + b_o\right)$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

peephole connection

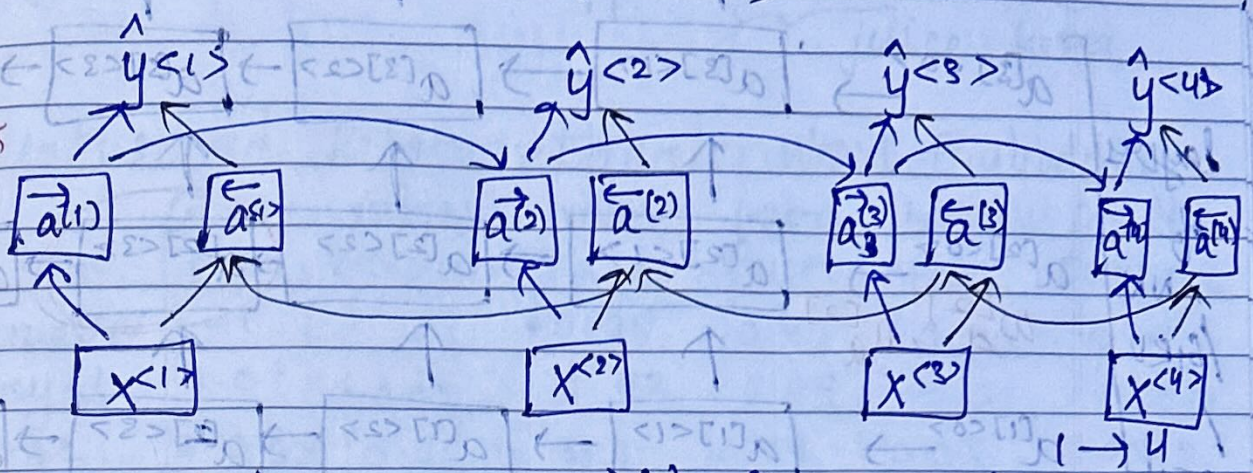
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$



- Re Peephole Connection, $c^{<t-1>}$ shows that it can go for all 3 gates computation as $a^{<t-1>}$, $x^{<t>}$ determine/affect the value of the gates
- $c^{<t-1>}$, first element affects only first element of gate, thus a one-to-one

connection.

* Bidirectional RNN (BRNN)



- acyclic graph, $\vec{a}^{(1)}$ (forward) left-to-right
$\overleftarrow{a}^{(1)}$ (backward) right to left

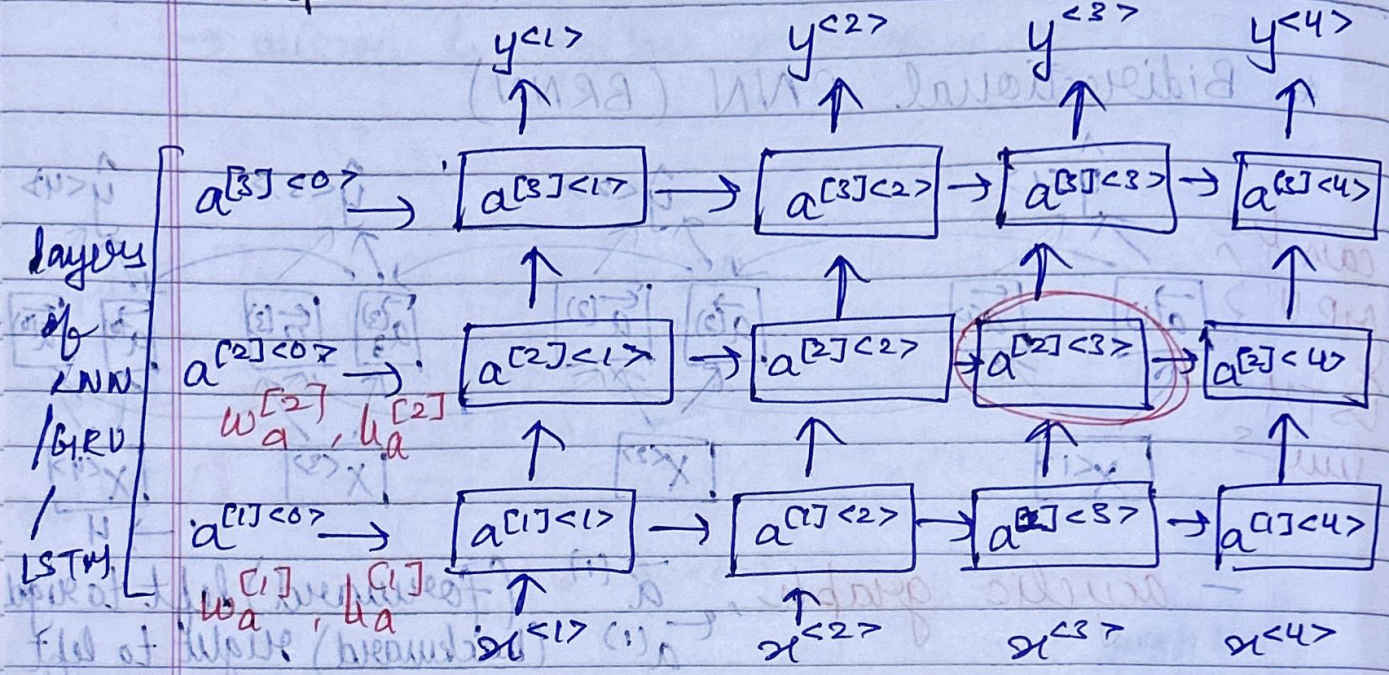
$$\hat{y} = g\left(W_y \left[\vec{a}^{(t)}, \overleftarrow{a}^{(t)}\right] + b_y\right) \qquad 4 \rightarrow 1$$

- At step $\overleftarrow{a}^{(3)}$, we can input both info from past & present, which goes into both forward & backward things at this step, & info from future.

Thus to understand Teddy it takes:
He said, "Teddy Roosevelt ...."
takes both input to determine what Teddy is.

- Disadvantage: You need the entire sequence to make correct predictions
- BRNN w/ LSTM, & other hybrid stacked versions are often used for good efficiency.

* Deep RNN:



$$a^{[2]<3>} = g\left(W_a^{[2]}\left[a^{[2]<2>}, a^{[1]<3>}\right] + b_a^{[2]}\right)$$

obj: Computationally expensive!