Week14
Sunday, May 22, 2022 5:23 PM

Sequence Models :-

Speech recognition, Music recognition, Sentiment analysis, Machine translation.

x : Harry Potter and Hermione Granger invented a new spell

$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ . . . . . $x^{(9)}$

$T_x = 9$

→ y : 1 1 1 1

$y^{(1)}$ $y^{(2)}$

$T_y = 9$ for single example.

$x^{(i)<t>}$ $T_x^{(i)} = 9$ for above example

$y^{(i)<t>}$ $T_y^{(i)} = 9$ i^{th} example

Vocabulary

↳ a, aaron, . . . , zulu → 10,000 words

Harry → one hot encoding for all words. → 10,000 ×1

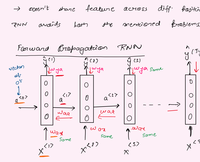
Features :-

→ i/p and o/p text can be of diff. length in diff. examples } important

→ doesn't share features across diff. positions of text.

This avails learn the mentioned features

Forward Propagation RNN



staying to translate

x only uses the info of few words. Eg.

He said, "Teddy was best president" → name of person

He said, "Teddy bear was at gate" → not a name

Equations :-

$a^{<0>} = 0$ → vector of 0's.

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

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

} important

Simplified version

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

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

Easy version :-

$a^{<t>} = g[W_a [a^{<t-1>}, x^{<t>}] + b_a)$ → weight for one vector.

$\begin{bmatrix} W_{aa} & W_{ax} \end{bmatrix} = W_a$

$100 \times 10{,}100 \rightarrow 100 \times 10{,}000 \quad 100 \times$

$\begin{bmatrix} a^{<t-1>}, x^{<t>} \end{bmatrix} = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} \begin{matrix} 100 \\ 10{,}000 \end{matrix}$ | 10100

Back propagation through time :-

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

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

⊙ length of input sequence = length of output sequence. } easy part

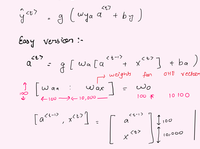
So, it was easy to write equations & back prop.

Different type of RNN :- Important

Sentiment classifications :-

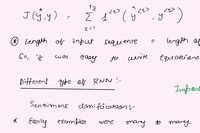
x Early examples were many to many.

x S.C is many to one.

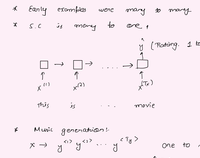


x Music generation :

$x \rightarrow y^{<1>}, y^{<2>} \cdots y^{<T_y>}$ one to many



x Machine translation :



Many to many.

Important

Language model and sequence Generation :-

Apple and Pear salad
. . . . . fear . .
} → which one is correct?
↳ Language model will tell us the probability $P(y^{<1>}) * P(y^{<2>}) \cdots$

P(Sentence) = ?

$P(y^{<1>}, y^{<2>}, \ldots y^{<T_y>})$ → output of sequence. (probability all words)

⊙ Training set : large corpus of english text.

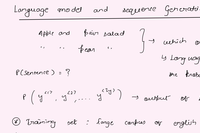
↳ Tokenize, one vectors, add <EOS> end of sentence

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

$y^{<1>}$ $y^{<2>}$ . . . . . $y^{<8>}$

Mau (cat) } not in corpus.

$P(y^{<3>} | \text{cats average})$



Main Idea (MI) :- predicted words are causing f/w.

Exploding words are rounded forward. → not very accurate.

Loss function :-

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

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

⊙ Vanishing gradients :-

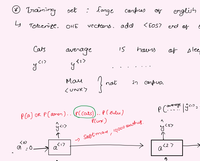
The cat . . . . . was fault.

The cats . . . . . were fault.

* RNNs cannot retain the info in long n/w.

In that case vanishing gradient is a problem in backprop.

⊙ Gated Recurrent Network :-



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

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

C = Memory cell.

$c^{<t>} = a^{<t>}$ → Assume

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

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

Remember gate :-

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

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

memorize that cat was singular.

GRU representation.

↳ Try to remember equations → simplify.

Now, how relevant $c^{<t-1>}$ is to $\tilde{c}^{<t>}$? we can use Γ_r

∴ All equations become :

$\tilde{c}^{<t>} = a^{<t>}$

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

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

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

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

if z is highly negative, then Γ_u is close to 0, which means $c^{<t-1>}$ gives relevant to $c^{<t>}$, so if calculations go wrong.

⊙ Long Short Term memory :-

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

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

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

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

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

$c^{<t>}$