[RECURRENT NN]

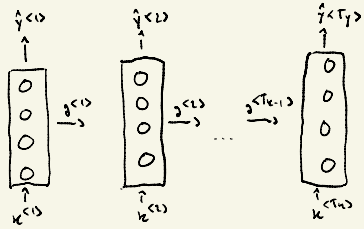
Notation

RNN ——→

- Limitation: uses only info earlier in the sequence to make predictions.

BRNN solves this limitation

$a^{<0>} \longrightarrow$



T_x = size of x
T_y = size of y $\rightarrow T_x = T_y$ (only for this case)

Forward propagation:

$a^{<0>} = \vec{0}$; W_{aa}, W_{ya} = parameters Matrices

$a^{<1>} = g_1(W_{aa}\, a^{<0>} + W_{ax}\, x^{<1>} + b_a)$ ← tanh / relu

$\hat{y}^{<1>} = g_2(W_{ya}\, a^{<1>} + b_y)$ ← i.e Sigmoid

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

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

$$\begin{bmatrix} a^{<t>} = g(W_a [a^{<t-1>}, x^{<t>}] + b_a) \\ \hat{y}^{<t>} = g(W_{ya}\, a^{<t>} + b_y) \end{bmatrix}$$

$$\begin{bmatrix} W_a = [W_{aa} \mid W_{ax}] \\ [a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} \end{bmatrix}$$

Different types of RNN

$T_x = T_y$: many-to-many arch.

$T_x \neq T_y$:

- Sentiment classification: many-to-one arch. { x = text, y = 0/1, 0 ... 5
- Sequence generation: one-to-many arch.
- Machine translation: many-to-many, where $T_x \neq T_y$

Sequence generation

- Language modelling:

→ Speech recognition: $\underset{\text{Probability}}{P}(\text{sentence}) = ?$

$= P(y^{<1>}, y^{<2>}, \ldots y^{<T_y>})$

Probability of sequence of words

→ Build a language model:

1) Training set: large corpus of english text
2) Tokenize: one-hot encoding, <EOS>, <UNK>
3) $\hat{y}^{<3>}$ → softmax with N outputs where N = corpus size
   - given the first 2 words are cats average
     What is the probability of $y^{<3>}$?
     $P(? \mid y^{<1>}, y^{<2>})$

→ A sequence model, models the chance of any particular sequence of words.

- Sampling sequence model: allows you to generate new sequences from your RNN language model.

Issues with RNN

- Vanishing gradient (VG), RNN fail to capture long term dependencies.
- " " much harder to solve than exploding gradients.

Gated Recurrent Unit (GRU)

- Helps with VG . it introduces a new cell, called C (memory cell) • $C^{<t>} = a^{<t>}$
- it memorises old values in the sequence

LSTM

- Same purpose of GRU: allow you to learn very long connections.
- Ruling equations are:

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

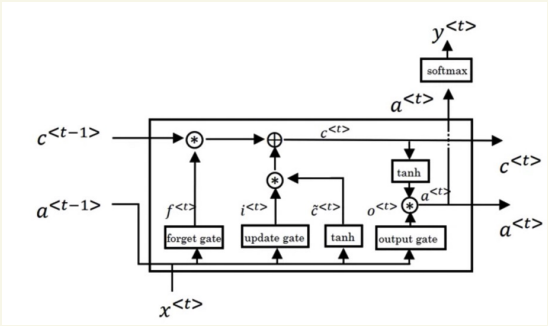
(update) $$\Gamma_u = \sigma\left(W_u\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$

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

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

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + \Gamma_f * c^{\langle t-1 \rangle}$$

$$a^{\langle t \rangle} = \Gamma_o * \tanh c^{\langle t \rangle}$$



- Easier to build deeper NN with GRU. But LSTM is more used

Bidirectional RNN and Deep RNN

- BRNN: getting information from the future
- Adding a backward layer for each forward layer