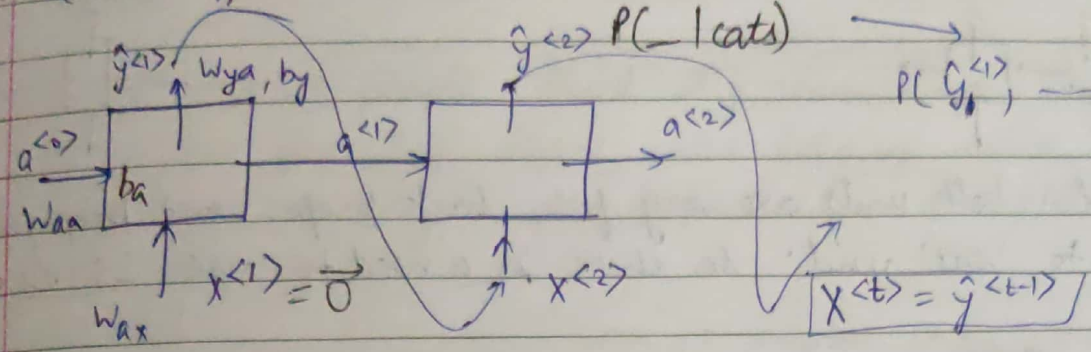
Date 14/11/19

* Language Modelling - What is the probab. associated with the sentence or word?

P(The apple & the pair salad) ≈ 3.3×10^{-13}
P(The apple & the pear salad) ≈ 5.7×10^{-10} ✓ (more common)



$P(\hat{y}_1^{<1>}, \ldots, \hat{y}_4^{<4>})$.

$a^{<0>}$ is initialised to $\vec{0}$ (No input)

$x^{<7>}$ = Most frequently occuring words is not a good idea as all our generated sent. by RNNs will start with 'The'.

$$a^{<t>} = g_1(W_{ax} x^{<t>} + W_{aa} a^{<t-1>} + b_a)$$ (g_1: tanh)

$$\hat{y}^{<t>} = g_2(W_{ya} a^{<t>} + b_y)$$ (g_2: softmax)

Idea:- We select the 1st word, based on the probab of words in the corpus. Eg- P(the) = 0.07, there are 7% chances of choosing 'the' out of all possible words/choices

P(cats) = 0.0004
$P(\hat{y}^{<1>} \mid \text{cats})$
$P(\hat{y}^{<3>} \mid \text{cats average})$

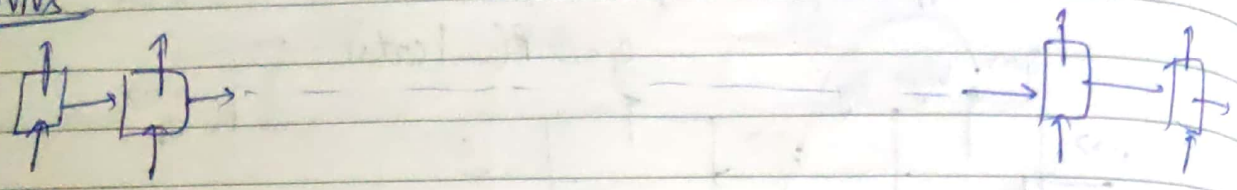
- <EOS> : End of Sent. marker
- To deal with <UNK>, we can even have a model at character level as no chr (ASCII) can be unknown.

LSTMs

RNNs suffer from Vanishing Gr. prblm much more than DNNs or CNNs.

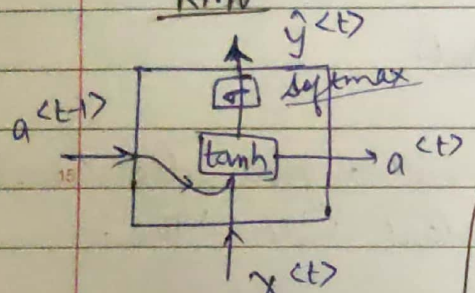
The cat, which already ate fruits like apple, pear, etc., was full.

RNNs

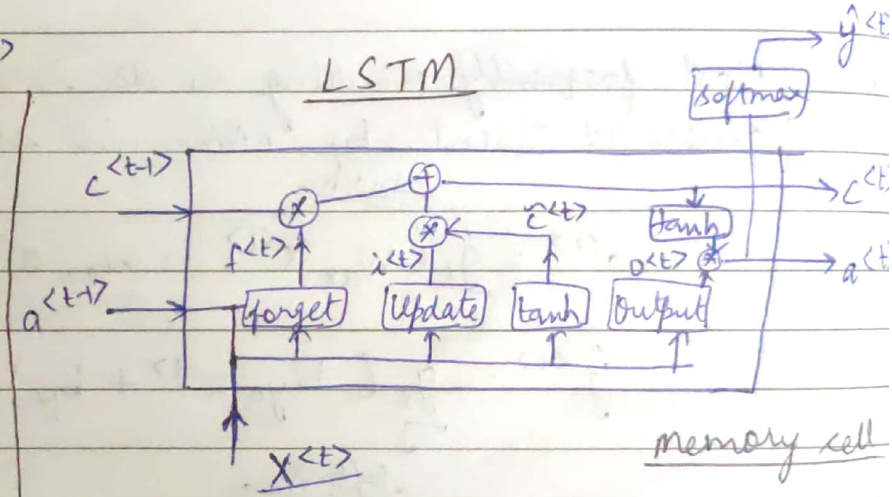


As both units are very far, back prop. won't propagate errs to 'cat' unit. So, there is a need to use LSTMs.

RNN ⟶ Doesn't capture long term dependencies.

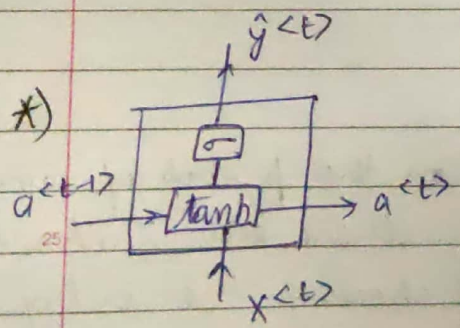


LSTM



memory cell

*)



(forget gate)

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

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

$$a^{<t>} = g_1(W_{ax} x^{<t>} + W_{aa} a^{<t-1>} + b_a)$$

$$\hat{y}^{<t>} = g_2(W_{ya} a^{<t>} + b_y).$$

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

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

update $c^{<t-1>}$ & change it to $c^{<t>}$.

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

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

In our terminology, $W_f = [W_{fa}, W_{fx}]$. Composed of 2 wts. (Andrew Ng combined both in 1 slide).

$c^{\langle t \rangle}$: Cell state (for memorizing)
$a^{\langle t \rangle}$: Activation of previous unit

[Before LSTMs, people tried skip connections to maintain long term dep. But there were lots of extra conn. resulting in very high # parameters. Hence, LSTMs were invented.

GRUs (much simpler than LSTMs).

$$\tilde{c}^{\langle t \rangle} = \tanh(W_c [\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c).$$

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

(Relevance gate)

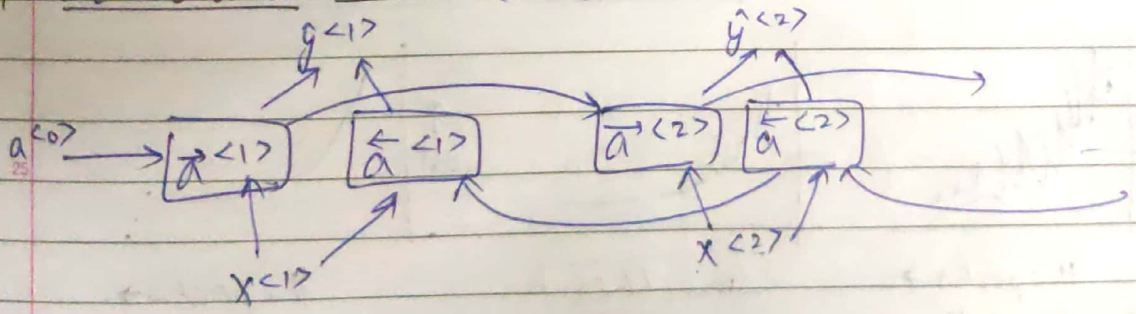
$$\Gamma_r = \sigma(W_r [c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_r).$$

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

$$a^{\langle t \rangle} = c^{\langle t \rangle}.$$

[yes only 2 gates - update and Relevance.

Bidirectional RNNs (BRNN)



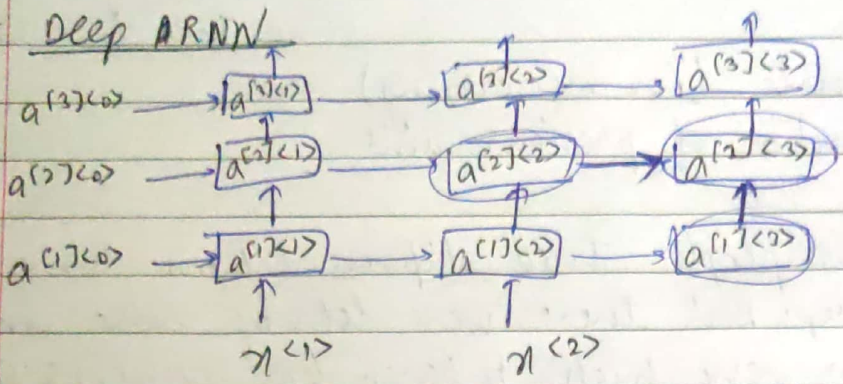
$$\hat{y}^{\langle t \rangle} = g(W_{ya} a^{\langle t \rangle} + b_y) \quad \text{— one dir}^n$$

$$= g(W_{y\overrightarrow{a}} \overrightarrow{a}^{\langle t \rangle} + W_{y\overleftarrow{a}} \overleftarrow{a}^{\langle t \rangle} + b_y)$$

fwd Bcwd

$a^{[1]<2>}$ means 1st layer, 2nd unit / timestamp.

* Deep RNN



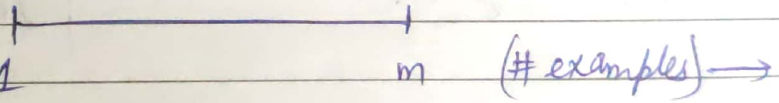
$$a^{[2]<3>} = g\left(W_a^{[2]}\left[a^{[2]<2>}, a^{[1]<3>}\right] + b_a^{[2]}\right).$$

Turing complete archi

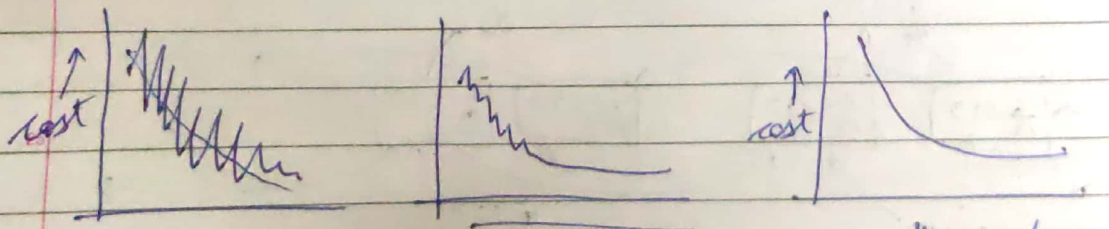
Date 21/11/19

SGD (Mini-batch Gr. Descent)

We can consider following # examples in 1 Gr. desc update -



for 1 ex. | $1 < \_ < m$ | for m exs.



(Stochastic GD) | Smoother (Mini-Batch GD) | (Batch GD)

$x^{\{1\}}$ → Mini-batch 1. | $x^{\{2\}}$ → Mini-batch 2.
$x^{(1)}$ → 1st tr. ex. | $x^{(2)}$ → 2nd tr. ex.

Stochastic Gr. Descent → When batch size = 1. We train on just 1 ex. before updating the parameters (exs. ~~randomly~~ epoch-wise chosen)

With Batch-GD, learning is very slow
With SGD, learning is very fast
While mini-Batch GD takes good of both the things.

We require:
1) Learning should be moderately fast
2) should reach to min. as fast as possible.

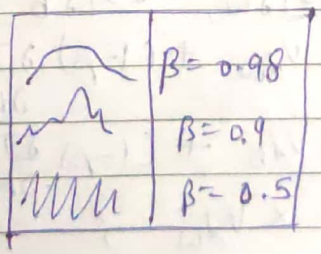
Moving Avg.: Taking avg of obs. within a window. Assign higher wts. to nearer obs. & so on

$$\begin{cases} V_0 = 0 \\ V_1 = 0.9 V_0 + 0.1 \theta_1 \\ V_2 = 0.9 V_1 + 0.1 \theta_2 \\ \vdots \\ V_t = 0.9 V_{t-1} + 0.1 \theta_t \end{cases}$$

$$\boxed{V_t = \beta V_{t-1} + (1-\beta)\theta_t}$$ exponentially weighted average

$\beta = 1$, just considering past avg

$\beta = 0$, just considering current value



Momentum

$V_0 = 0$
$V_0 = \beta V_0 + (1-\beta)\theta_1$
$V_0 = \beta V_0 + (1-\beta)\theta_2$

$V_{100} = 0.9 V_{99} + 0.1 \theta_{100}$
$V_{100} = 0.9(0.9 V_{98} + 0.1 \theta_{99}) + 0.1 \theta_{100}$
(β)

Date 26/11/19

Exp. Weighted Averages

$V_\theta = 0$

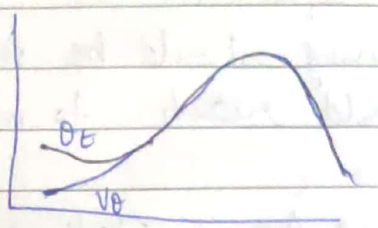
Repeat {

Let new θ_t

$V_\theta := \beta V_\theta + (1-\beta)\,\theta_t$

}

Initially, V_θ will be very less & then it slowly tries to match up with moving weighted averages.

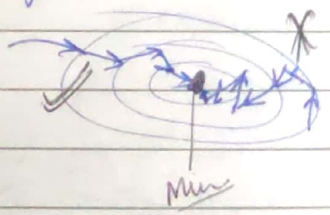


GD with Momentum

Try to make update step more efficient

$$w := w - \alpha \frac{\partial J}{\partial w}$$

Momentum helps in moving down gradients to reach min quickly. It disallows sideways movements when we move away from minima.



Momentum

On iter t:

compute $\partial J/\partial w$, $\partial J/\partial b$ on current mini-batch

$V_{dw} = \beta V_{dw} + (1-\beta)\frac{\partial J}{\partial w}$ → gradient

$V_{db} = \beta V_{db} + (1-\beta)\frac{\partial J}{\partial b}$ → gradient

$w := w - \alpha V_{dw}$; $b := b - \alpha V_{db}$

$$V_{dw} = \beta V_{dw} + (1-\beta)\,dW$$

α: learning rate

β = Momentum

$\frac{V_{dW}}{1-\beta^t}$ (Bias correction to upgrade the V_{dW} value).

RMS prop

$$S_{dW} = \beta S_{dW} + (1-\beta)\, dW^2 \quad \rightarrow \left(\frac{\partial J}{\partial W}\right)^2 \text{ is element-wise multiplication}$$

$$S_{db} = \beta S_{db} + (1-\beta)\, db^2$$

$$W := W - \alpha \frac{dW}{\sqrt{S_{dW}+\epsilon}} \; ; \quad b := b - \alpha \frac{db}{\sqrt{S_{db}+\epsilon}}$$

Adam opti.

V_{dW}, V_{db} (by Momentum)

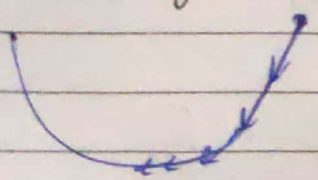
S_{dW}, S_{db} (by RMS prop)

$$V_{dW}^{corrected} = V_{dW} / (1-\beta_1^t) \; , \quad S_{dW}^{corr} = S_{dW} / (1-\beta_2^t)$$

$$\boxed{W := W - \alpha \frac{V_{dW}^{corr}}{\sqrt{S_{dW}^{corr}}+\epsilon}} \; ; \quad \boxed{b := b - \alpha \frac{V_{db}^{corr}}{\sqrt{S_{db}^{corr}}+\epsilon}}$$

$\beta = 0.9, \quad \beta_2 = 0.999, \quad \epsilon = 10^{-8}$

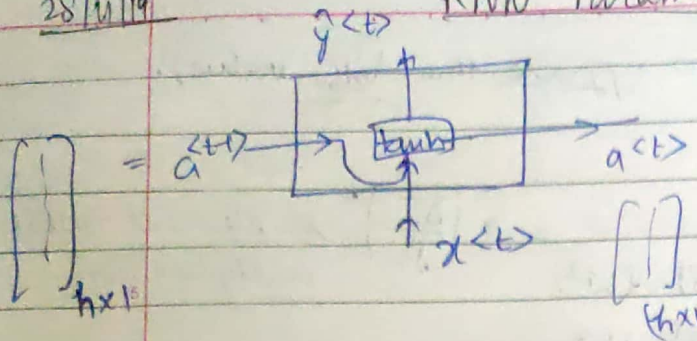
lr decay



As # epochs ↑, lr ↓ so that we don't miss minima.

28/11/19

RNN Parameters



$h \times 1$

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

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

$(h \times 1)$

$x^{<t>} = \begin{bmatrix} | \\ | \end{bmatrix}_{n \times 1}$ {input can be video matrix, word embeddings or one-hot vector.}

①

$$\begin{bmatrix} \\ \end{bmatrix}_{h \times n} \cdot \begin{bmatrix} | \\ | \end{bmatrix}_{n \times 1} + \begin{bmatrix} \\ \end{bmatrix}_{h \times h} \begin{bmatrix} | \\ | \end{bmatrix}_{h \times 1} + \begin{bmatrix} \\ \end{bmatrix}_{h \times 1} = \begin{bmatrix} \\ \end{bmatrix}_{h \times 1}$$

W_{ax} $x^{<t>}$ W_{aa} $a^{<t-1>}$ b_a $a^{<t>}$

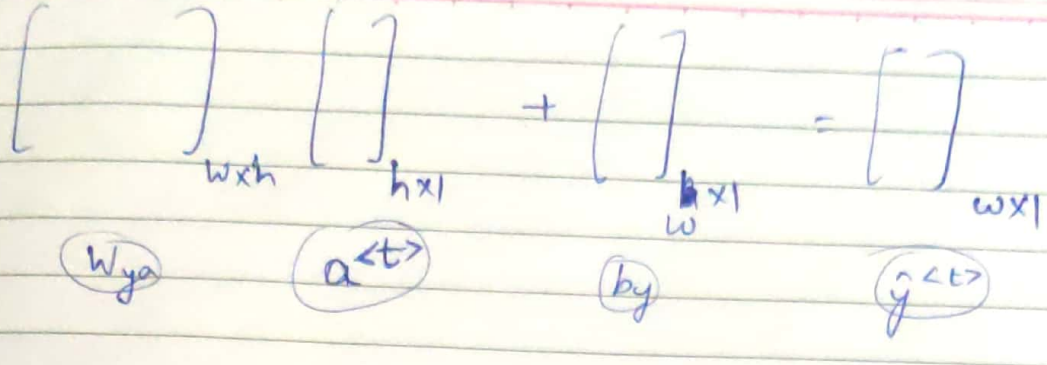
$$\#\text{parameters} = (h \times n) + (h \times h) + (h \times 1)$$
$$= \boxed{h(n + h + 1)}$$

Combining W_{aa} & W_{ax}.

$$\begin{bmatrix} \\ \end{bmatrix}_{h \times (n+h)} \cdot \begin{bmatrix} \\ \end{bmatrix}_{(n+h) \times 1} + \begin{bmatrix} \\ \end{bmatrix}_{h \times 1} = \begin{bmatrix} \\ \end{bmatrix}_{h \times 1}$$

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

$$\Rightarrow a^{<t>} = \tanh\left(W_a [x^{<t>}, a^{<t-1>}] + b_a\right).$$



LSTM parameters

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

$h \times (n+h)$ $(h \times 1)$ $(h \times 1)$ $(h \times 1)$

$(n+h) \times 1$

parameters $= 4 \times [h \times (n+h) + h]$.

↳ 4 gates (forget, update, O/P, tanh).

parameters in GRUs $= 3 \times [h \times (n+h) + h]$.