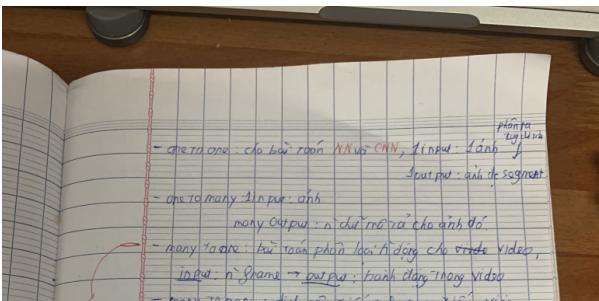
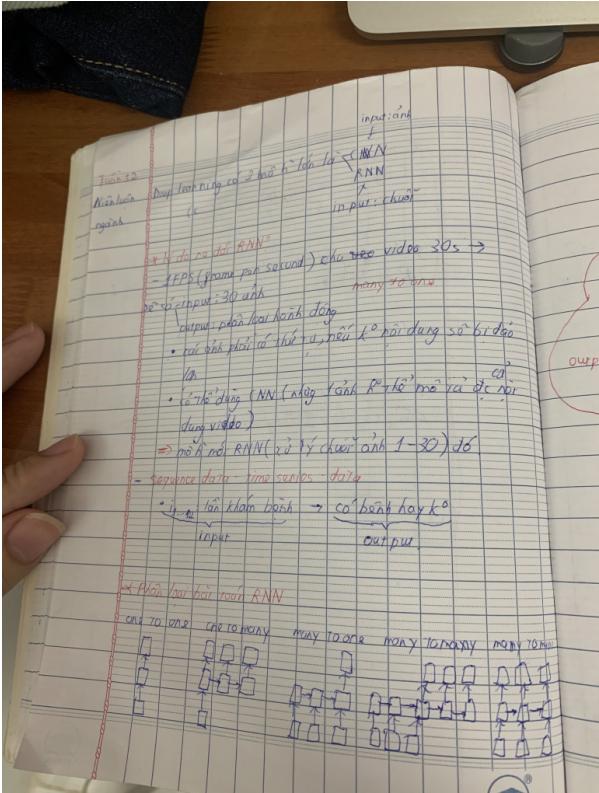


Week 12

Sunday, March 20, 2022 2:37 PM



*: Backpropagation Through Time (BPTT)

3 Tham số cần tìm: w, u, v

Để thực hiện gradient descent: tính đạo hàm của $\frac{\partial L}{\partial w_i}$; $\frac{\partial L}{\partial b}$; $\frac{\partial L}{\partial \theta_0}$

f) Dao hamin với V:

$$\frac{\partial L}{\partial V} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial V}$$

f) Vợ Uvà` M

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \vec{y}} \times \frac{\partial \vec{y}}{\partial s_{30}} \times \frac{\partial s_{30}}{\partial W} = \frac{\partial s_{30}'}{\partial W} + \frac{\partial s_{30}}{\partial s_{29}} \times \frac{\partial s_{29}}{\partial W}$$

$$Do: S_{30} = g (W \times S_{29} + V \times x_{30}) \text{ có } S_{29} \text{ là } E \text{ và } W$$

$$\text{Aufstellung CI: } (f(x) \times g(x))' = f'(x) \times g(x) + f(x) \times g'(x)$$

$$= \sum_{i=0}^{20} \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{30}} * \frac{\partial s_{30}}{\partial s_i} * \frac{\partial s_i}{\partial W}$$

$$V = \prod_{j=1}^{29} \frac{\partial s_{j+1}}{\partial s_j} \text{ và } \frac{\partial s_i^2}{\partial W} \text{ là đạo hàm của } s_i \text{ với } W$$

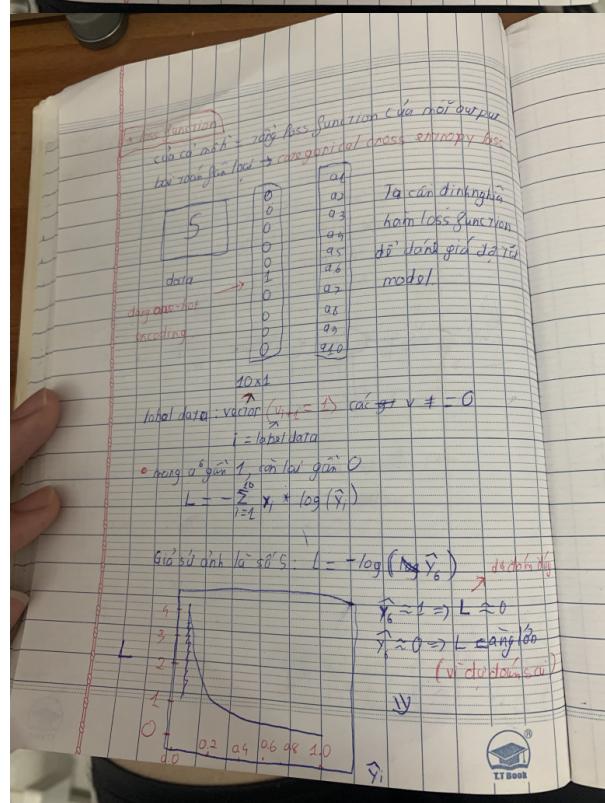
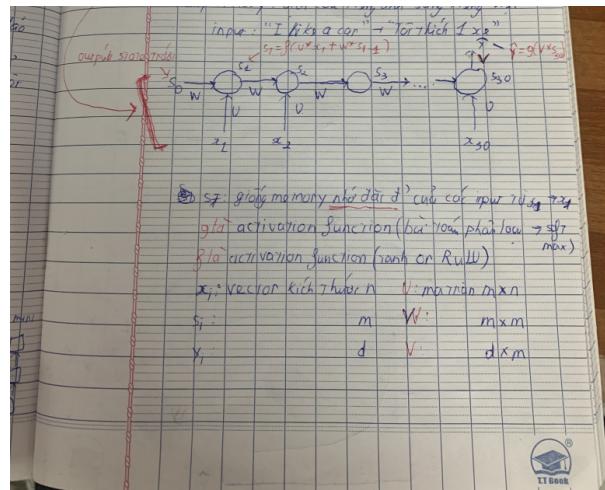
khi s_{i-1} là constant với W .

*: Nhìn vào công thức đạo hàm của W và L

Ta thấy hiện tượng vanishing gradient ở các state đầu tiên → nên cần mở h' tối ưu để

grain kiểm-trúng nay

→ Khi w nhỏ hơn 1 thì khi tính $các$
 $gradient$ $các$ layer đều tiến $sẽ$ phai
nhận $tích$ $của$ $rất$ n $số$ $bé$ < 1 $nên$
 $gradient$ $sẽ$ $\rightarrow 0$, \Rightarrow $bé$ $độ$ n $hết$ ki
 $sẽ$ $trong$ $gradient$ $descent$ $vô$ $ngô$ $hì$



\Rightarrow Bài toán trả Thành bài toán Tìm Min L

vai trò hứa số trong NN sẽ k' đc
để nêu

$$\frac{\partial s_{30}}{\partial s_i} = W^{30-i} \times \prod_{j=1}^{29} (1 - s_j^2)$$

$s_i < 1, W < 1 \Rightarrow$ s'kiai $\delta x q$: $\frac{\partial s_{30}}{\partial s_i} \approx 0 \Rightarrow \frac{\partial L}{\partial W} \approx 0 \Rightarrow$ vanishing gradient

Tìm hứa số W và b

Gradient descent:

giả sử đang ở đỉnh núi và bạn cần xuống thuyền bằng cách lanh nhát
Thuyền là đây là ∇ đ'cđt Tílê
chú ý sẽ cảm nhận tại chỗ đó có chỗ nào dốc nhất thì ra bờ xuống
Tụng b'c 1 cho đến khi cảm nhận xung quanh k' còn dốc nữa (b'ng ph'ng)
 \rightarrow thuyền

LSTM

long short term memory

- Đòm hì RNN tacó:

$$\text{Đạo hàm } L \text{ với } W \text{ ở state } i: \frac{\partial L}{\partial W} = \frac{\partial L}{\partial \vec{y}} \times \frac{\partial \vec{y}}{\partial s_{30}} \times \frac{\partial s_{30}}{\partial s_i} \times \frac{\partial s_i}{\partial W}$$

$= \prod_{j=i}^{29} \frac{\frac{\partial s_{j+1}}{\partial s_j}}{\partial s_j}$

- Giả sử activation function: tan

$$\Rightarrow S_t = \tan(V * x_t + W * s_{t-1})$$

$$\frac{\partial s_i}{\partial s_{i-1}} = (1 - s_i^2) \neq W \Rightarrow \frac{\partial s_{30}}{\partial s_i} = W^{30-i} \times \prod_{j=1}^{29} (1 - s_j^2)$$

$S_i < 1, W < 1 \Rightarrow$ n state $\partial x_q : \frac{\partial S_{30}}{\partial S_i} \simeq 0 \Leftrightarrow \frac{\partial L}{\partial W} \simeq 0 \Rightarrow$ vanishing gradient

state change $\propto a \Rightarrow$ vanishing gradient

\Rightarrow Vanishing gradient
causes slow update

\Rightarrow RNN k° hoc dc cac thong tin o tne do xg. do bi vanishing gradient

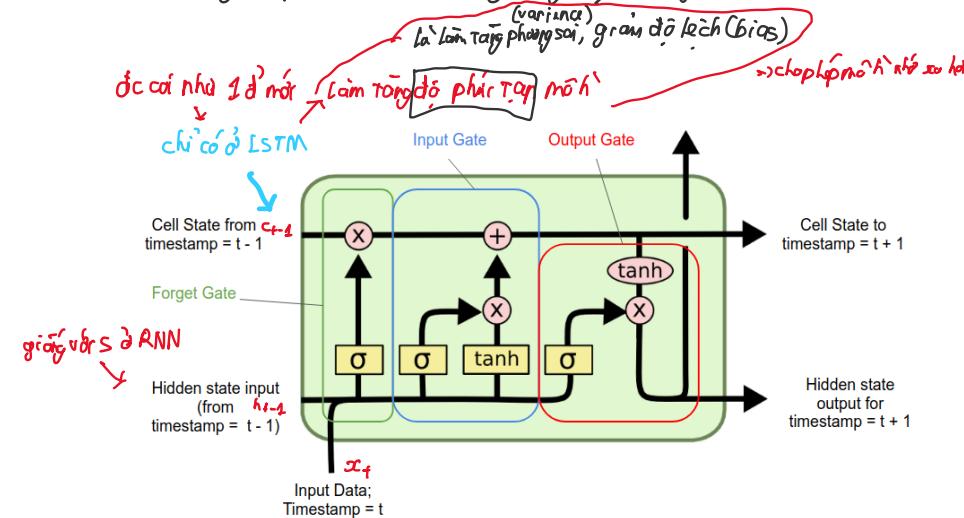
- RNN: mang thông tin từ layer trước ra layer sau nhưng chỉ mang qua dc 1 số state nhất định, sau đó sẽ bị **vanishing gradient**

↪ KNN chỉ học dc các state gần nhau ⇒ **SHORT TERM MEMORY**

Vd: "Mặt trời mọc ở hướng..." ⇒ Đóng

"Tôi là ng` Việt Nam. Tôi đang sống tại N` ng`oai. Tôi có Th`n`i tr`i chay' T`i... ⇒ RNN (Short term memory k`n`i nhau)

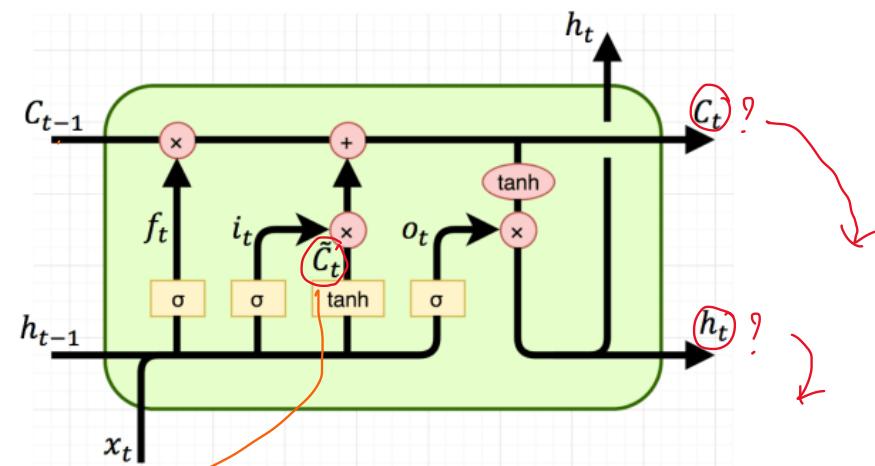
⇒ Long term memory ⇒ LSTM ra đở



$$\text{Forget gate: } f_t = \sigma(v_f \cdot x_t + w_f \cdot h_{t-1} + b_f) \quad \left. \begin{array}{l} 0 < f_t, i_t, o_t < 1 \\ b_f, b_i, b_o \text{ là các h`e s`o' bias} \end{array} \right\}$$

$$\text{Input gate: } i_t = \sigma(v_i \cdot x_t + w_i \cdot h_{t-1} + b_i)$$

$$\text{Output gate: } o_t = \sigma(v_o \cdot x_t + w_o \cdot h_{t-1} + b_o)$$



$\hat{c}_t = \tanh(O_c * x_t + W_c * h_{t-1} + b_c)$ same as in RNN
 $c_t = f_t * c_{t-1} + i_t * \hat{c}_t$, forget gate q' định xem cần lấy bao nhiêu
 từ cell state trước $f_t * c_{t-1}$
 , input gate sẽ q' định lấy bao nhiêu từ input của state
 và hidden layer của layer trước.
 $h_t = o_t * \tanh(c_t)$, output gate q' định xem cần lấy bao nhiêu từ cell state
 để trả thành out put của hidden state
 , ngoài ra h_t cũng được dùng để tính ra output y_t cho state t

