10/18/15

RECURRENT NEURAL NETS (RNNs)

(1) Motivation

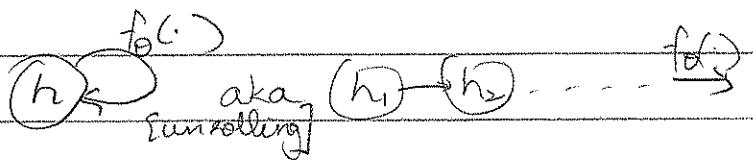
(a) → Sequences $\langle x_1, x_2 \ldots, x_T \rangle$, $\langle y_1, \ldots, y_T \rangle$ are everywhere!

→ How do we model input/hidden/output sequences?
[Especially when sequence length is not fixed/constant]

(b) How do we begin to model "memory"?
→ So our model "remembers" data from past?
→ & remembers what it has already predicted?

(2) ~~Most~~ General Formulation

→ No inputs

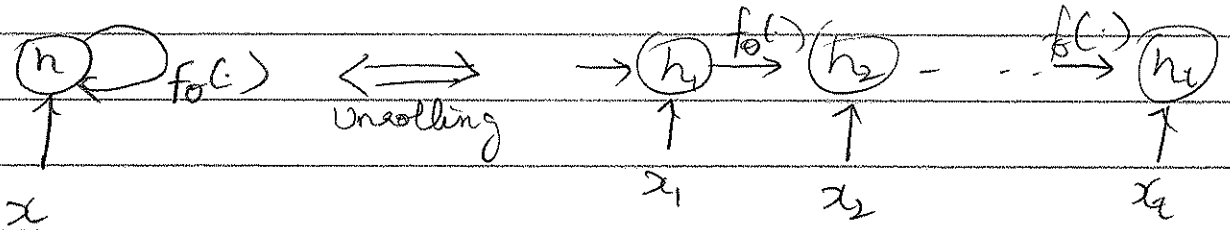


$$h_t = f_\theta(h_{t-1})$$

Think Markov Chain!

f = non-linear function θ = parameter

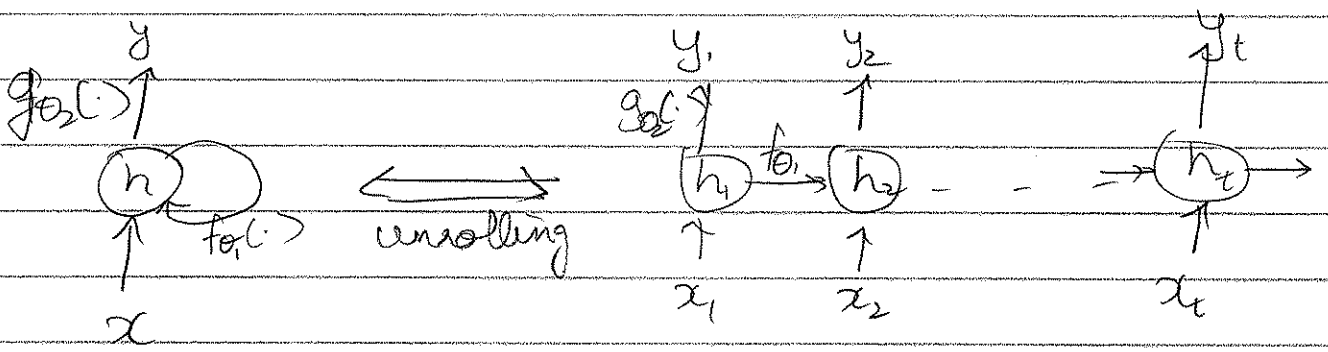
→ Sequential inputs



$$h_t = f_\theta(x_t, h_{t-1})$$

Think HMMs!

→ Sequential inputs & outputs



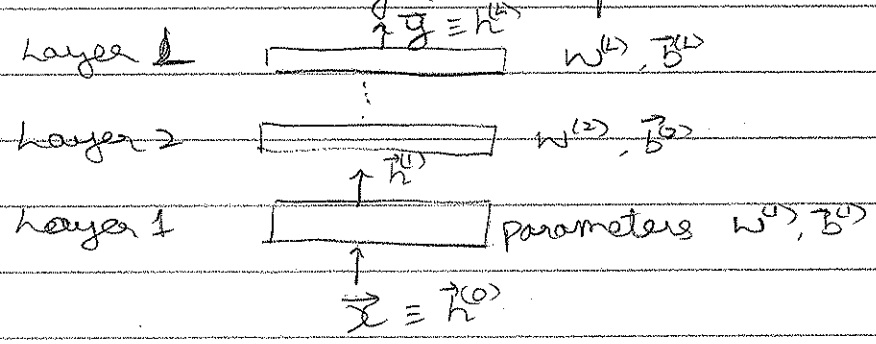
$$h_t = f_{\theta_1}(x_t, h_{t-1})$$

$$y_t = g_{\theta_2}(h_t)$$

Think Kalman Filtering!

(3) RNNs: aka How about we implement $f_\theta(\cdot)$ with neural nets.

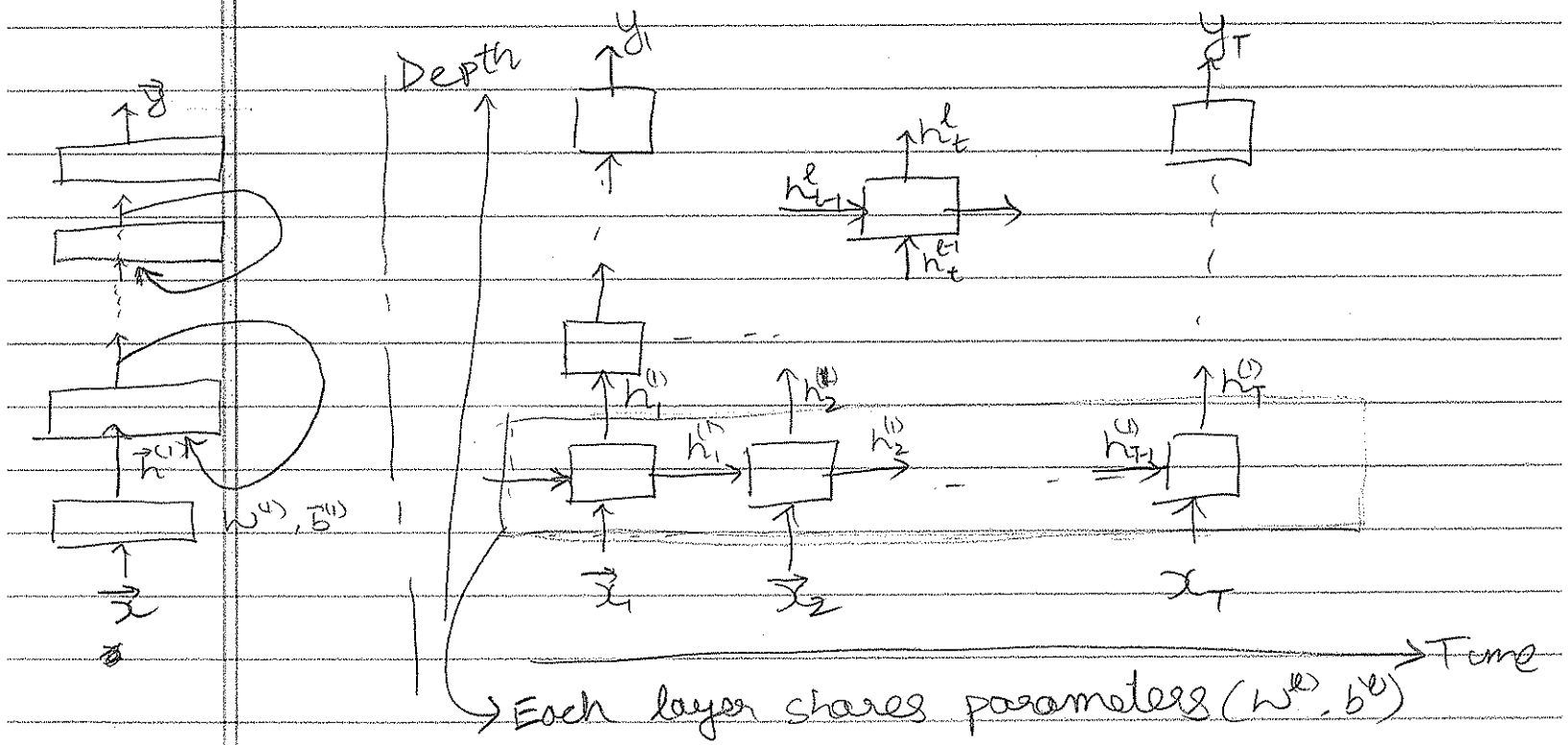
Recall: Multi-Layer Perceptrons (MLPs)



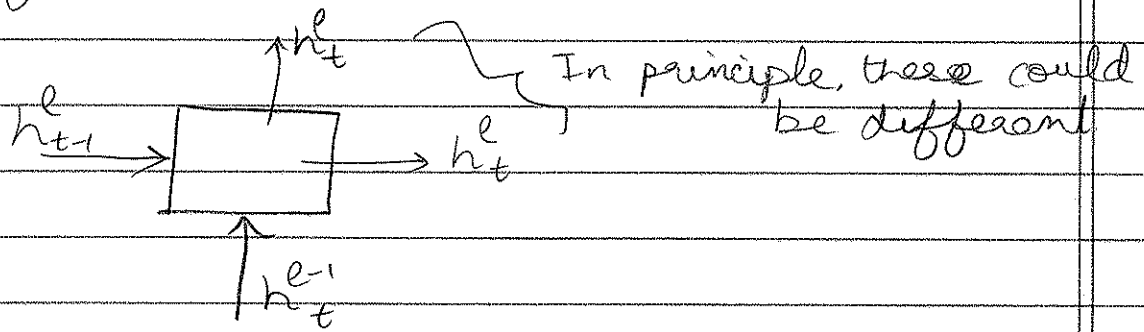
$$\vec{h}^{l} = \sigma(W^{(l)}\vec{h}^{l-1} + \vec{b}^{l})$$

↑ some non-linearity (tanh, sigmoid, ReLu)

RNNs: How about we add feedback loops



Think of a single "cell" or "step" (t, l)



→ Key "Forward-Pass" computation

$$h_t^l = f_\theta(h_t^{l-1}, h_{t-1}^l)$$

→ "Vanilla" RNN

$$\vec{h}_t^l = \sigma\left(W^{(l)} \begin{bmatrix} \vec{h}_t^{l-1} \\ \vec{h}_{t-1}^l \end{bmatrix} + \vec{b}^{(l)}\right) \quad ---(i)$$

element-wise tanh

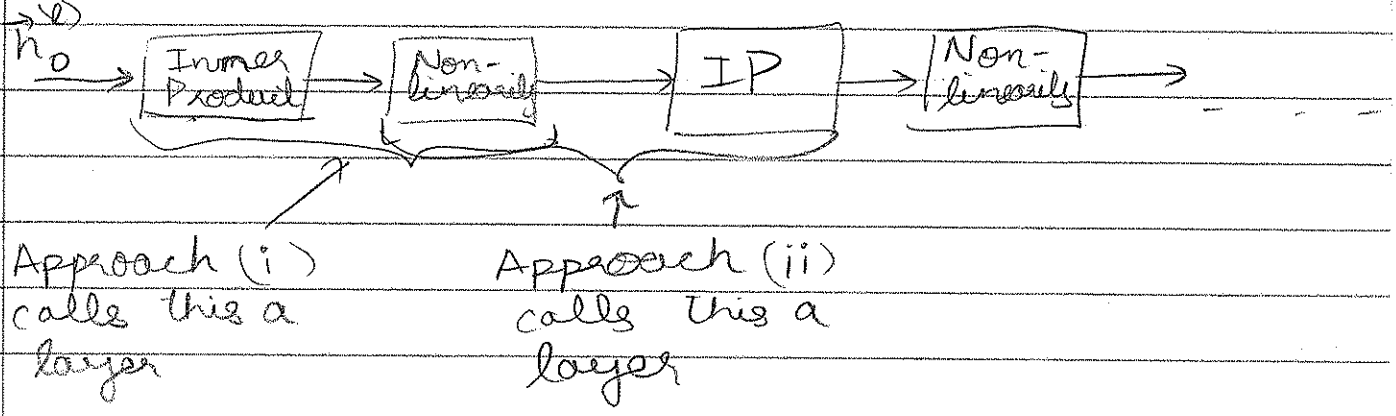
→ Note: Sometimes:

$$\vec{h}_t^l = W^{(l)} \begin{bmatrix} \vec{h}_t^{l-1} \\ \sigma(\vec{h}_{t-1}^l) \end{bmatrix} + \vec{b}^{(l)} \quad --- (ii)$$

→ Claim: The two forms (i) & (ii) are equivalent.

Why?

Proof Sketch: Just different ways of incorporating non-linearity. Either at start or end.



→ So that's a "Vanilla" RNN.

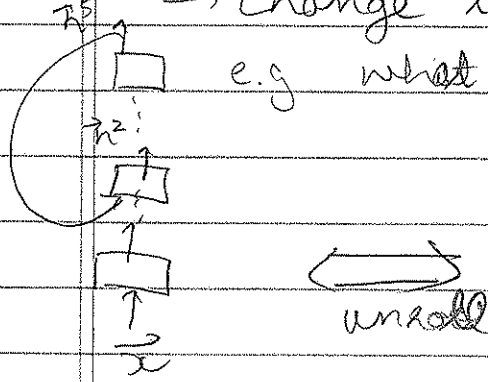
→ Generalizations:

→ change non-linearity (ReLU)

→ change cells altogether (LSTM, GRU)

→ change loops / graph structure

e.g what if layer 5 looped back to layer 2?



what do we get?

[Hint: not a grid]