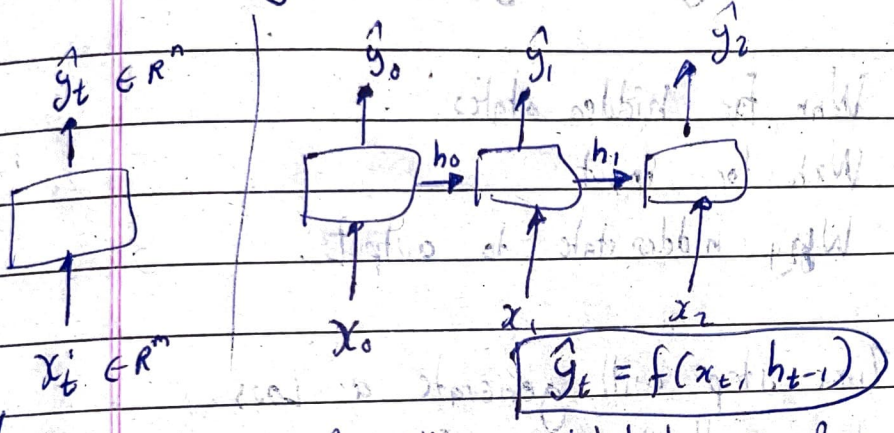
Sequential Data

Sequences

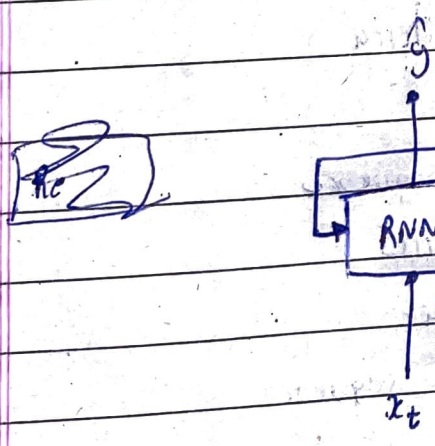
→ Sentiment Classification → many to one
→ Image Captioning → one to many
→ Machine Translation → many to many

Neurons with Recurrence

Handling Individual Time Steps



as $\hat{y}_2$ will be related to x_0 & x_1 as well we introduce h_0 & h_1 (past memory).



Apply # Recurrence Relation at every time step

$h_t = f_w(x_t, h_{t-1})$

RNN State Output & Updation

Input Vector : x_t

Update hidden state : $h_t = \tanh\left(W_{hh}^T h_{t-1} + W_{xh}^T x_t\right)$

Output Vector : $\hat{y}_t = W_{hy}^T h_t$

Note : W_{hh} for hidden states
W_{xh} for input
W_{hy} hidden state to output

→ Each Time step will generate a Loss
Final loss will be Combi of all these losses

Sequence modelling : Criteria

To model sequences

1. Handle variable length sequences
2. Track long term dependencies
3. maintain info about order
4. Share params across the sequence

~~Se~~ Predict the next word

→ NN can't interpret words

∴ Embedding: transform indexes into a vector of fixed size

1. Vocab — Corpus words
2. Indexing — word to index
3. Embedding
   i. one hot embedding
   cat = [0, 1, 00, 0 ...]
   ↑ i-th index
   ii. learned embedding

run walk	dog cat
day sun	happy sad

similar words closer

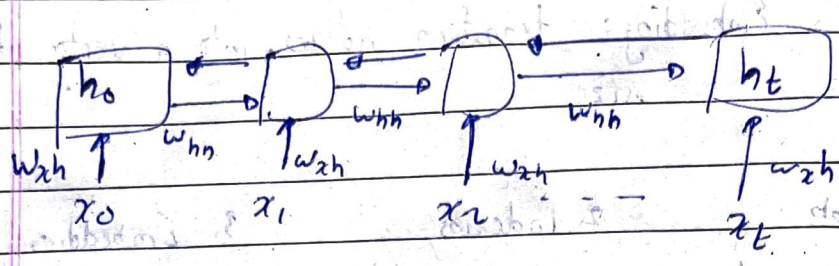
RNN can handle ~~long~~ variable sequence length as different sequence length are just different number of time steps

Long term dependency met by RNN by using past hidden state

Capture differences in sequence order

Back Prop through time (BPTT)

as we go through horiz axis for grad which is time



gradient wrt h_0 involves many factors of W_{hh} & repeated gradient computation

Problems →

(1) many values > 1 : exploding gradient
grads clipping to scale big grads

(2) Many values < 1 vanishing grads
1. Activation func
2. wt initialisation
3. Network architecture

Errors due to further back time have smaller & smaller grads → Bias params to capture short term dependencies

Trick 1 — ReLU prevents shrinking

Trick 2 — Init wts to Identity matrix & biases to zero
helps preventing wts to shrink to zero

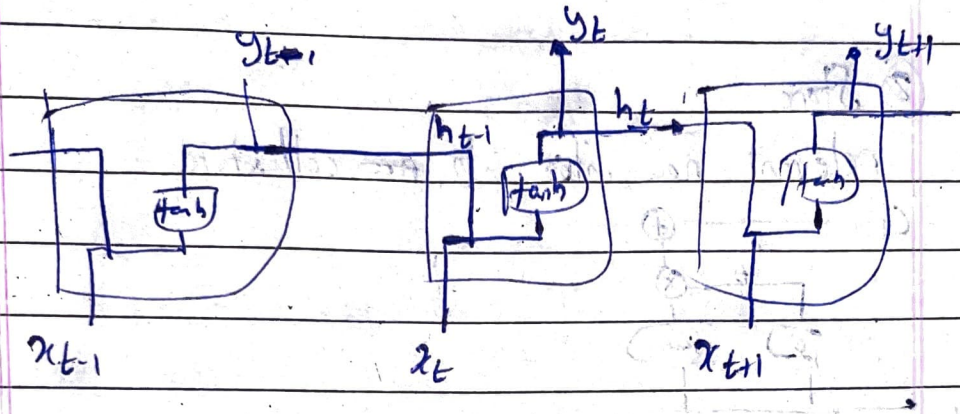
Trick 3 ——→

Trick 3 Gated Cells (LSTM, GRU)

use a more complex recurrent unit with gates control what info passes

Long short term memory (LSTM)

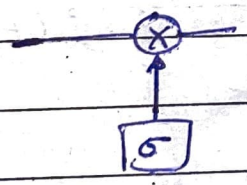
→ Standard RNN



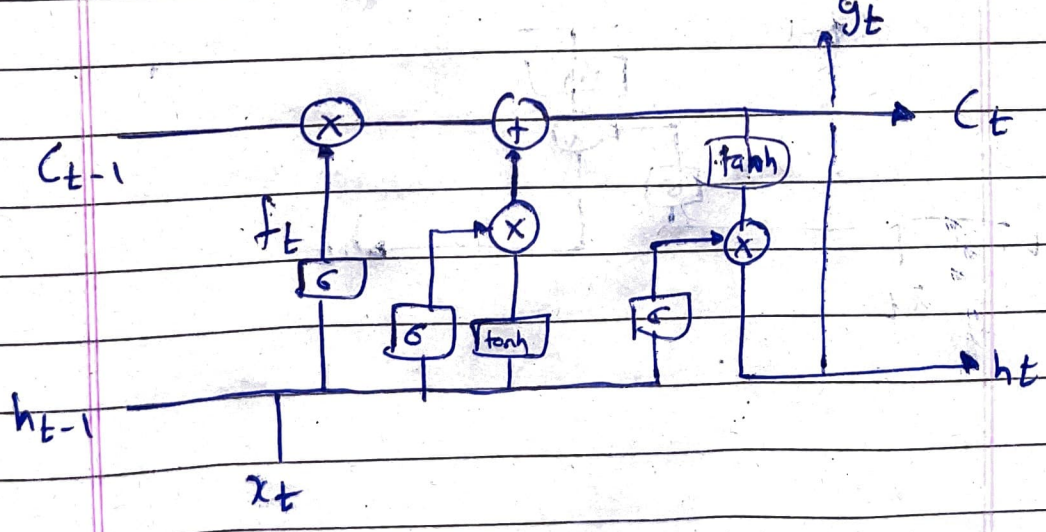
→ LSTM contain computational blocks that control information flow

- a lot more complex

→ Info added or removed using gates



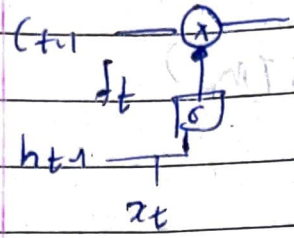
eg. sigmoid layer & pointwise multip^n



LSTM

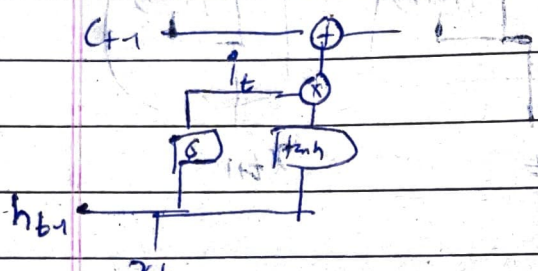
① Forget

irrelevant parts of previous state



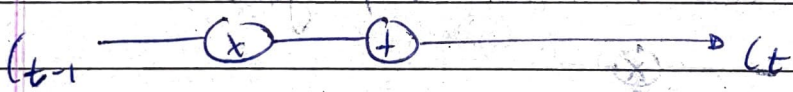
② Store

relevant new infos into ~~new~~ cell state

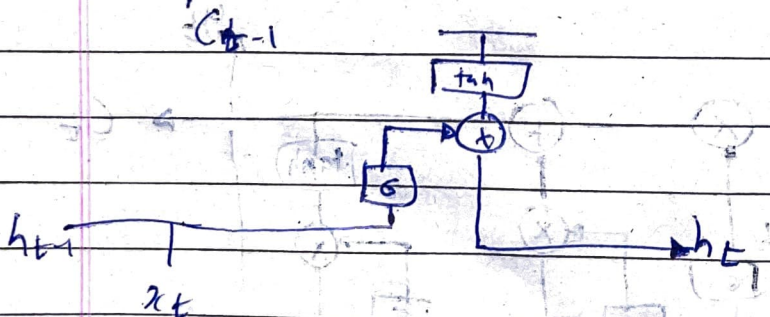


③ Update

selectively cell state



④ Output



→ maintaining a separate cell state allows uninterrupted gradient flow

Key concepts

1- maintain separate cell state from output
2. Use gates to control info flow
   - forget irrelevant
   - store relevant
   - Selectively update cell state
   - output
3 BPTT with ~~out~~ uninterrupted grad flow

Problems

1. Encoding bottleneck
2. Slow, no parallelisation
   BPTT very expensive
3. Not long-memory

Attention model provide learnable memory access