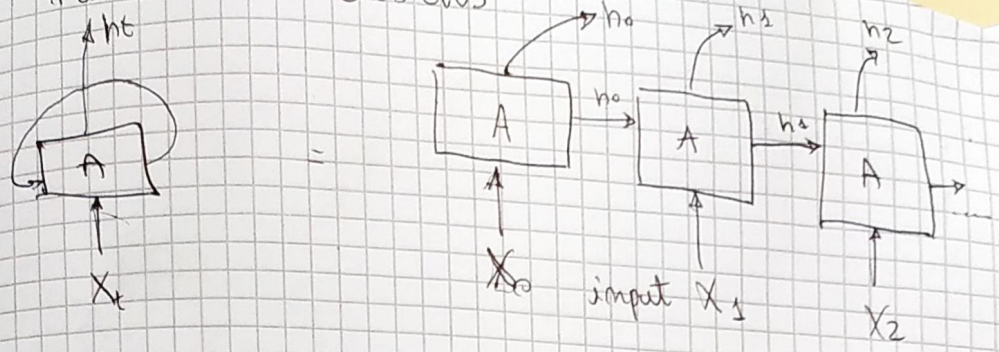
Recurrent Neural Networks



Question:

If no activation functions are used,
(ie network is just multiplication by a matrix A)
what governs the long term behaviour of an RNN?

Answer

Eigenvalues of A

λ is an eigenvalue if $\exists$ a vector s.t.

$$Ax = \lambda x$$

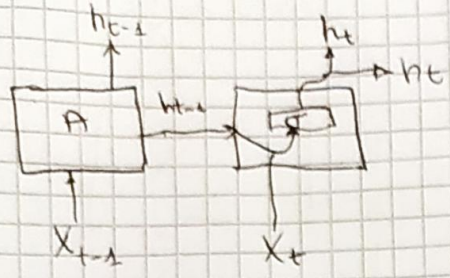
Let λ^A be the largest eigenvalue $|\lambda^A|$ is largest

If $|\lambda^A| > 1$, there is an exploding gradient problem.

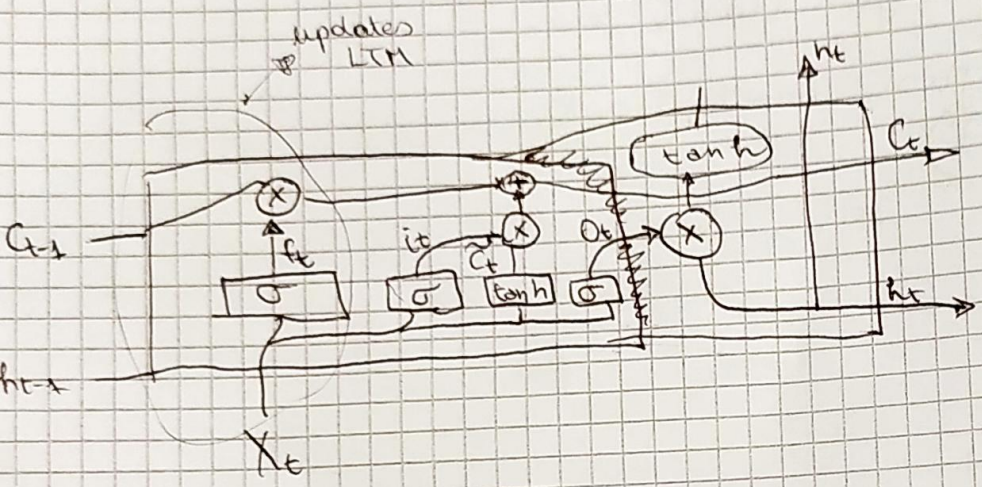
If $|\lambda^A| < 1$, there is a vanishing gradient problem

Solution

Let's design A cleverly. More pseudo-neuroscience.
Need to be able to forget.
Need to be able to remember LONG-TERM memories
Because I grew up in France, I like croissant, ..., baguettes and I speak FRENCH.



$\boxed{\sigma}$ feed forward NN with σ functions



Think of C_t as long-term memory

<u>NOTATION</u>

$\boxed{\sigma}$ = Feed Forward Neural Network with σ activation

$\otimes$ = pointwise multiplication $\begin{bmatrix} 1 \\ 10 \\ -5 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 0{,}5 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 0 \end{bmatrix}$

$\oplus$ = pointwise addition $\begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 4 \end{bmatrix}$

tanh = pointwise tan h $\quad \tanh \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} \tanh(1) \\ \tanh(2) \\ \tanh(3) \end{bmatrix}$

The forget gate layer f_t

bias vector

$$f_t = \sigma(W_f [h_{t-1}, X_t] + b_f)$$

~~Example~~ IMPORTANT: all components of f_t are in $(0,1)$

Analogy: Think of f_t as a pipe

if 1 : very imp. : remember

if 0 : forget

$$\begin{bmatrix} 10 \\ -2 \\ 5 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ 0{,}5 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \\ 2{,}5 \end{bmatrix}$$

The INPUT GATE i_t

$$i_t = \sigma(W_i [h_{t-1}, X_t] + b_i)$$

initialize LTM to 0

New Candidate layer

$$\tilde{C}_t = \tanh(W_c [h_{t-1}, X_t] + b_c)$$

has coords between $(-1, 1)$

New cell state (long term memory)

$$C_t = (f_t \otimes C_{t-1}) \oplus (i_t \otimes \tilde{C}_t)$$

Note

It's important to use tanh and $\otimes$ since we want vectors C_t to take any possible values

<u>Example</u>

Forget BOB'S gender, add ALICE'S gender

The output h_t

$$O_t = \sigma(W_o [h_{t-1}, X_t] + b_o)$$

$$h_t = O_t \otimes \tanh(C_t)$$

Tips for the real world

The overfitting problem

Enrico Fermi was skeptical of models with four parameters

"I remember my friend Johnny Von Neumann used to say, with 4 parameters I can fit an elephant, and with 5 I can make him wiggle his trunk."

MNIST $\approx$ 24000 parameters

We do not want our neural network to memorize the data

It should generalize to larger datasets.

Way to overcome overfitting

1. Train on a small sample of dataset
2. Validate on the dataset
3. Get more data!
   (maybe artificially enlarge data)
4. Dropout: At each training round Keep each node with probability p
   Idea: give all nodes a chance to contribute to learning
5. Regularization: Perturb Error function slightly

L2 Reg: $C = C_0 + \frac{\lambda}{2n} \sum_w w^2$ n = size of training set

λ is hyper parameter, controls tradeoff between minimizing cost error function and having small weights

L1 Reg: $C = C_0 + \frac{\lambda}{2n} \sum_w |w|$