CompSci, March 7, 2023

RNNs are specialized for processing sequences of data.

Each stage in an RNN contains:
- new input
- manipulate state
- reuses weights
- gives a new output.

Think of an RNN as a series of measurements.

- Simple example

$$m\frac{d^2x}{dt^2} + \eta\frac{dx}{dt} + x(t) = F(t)$$

initial conditions

$$x_0 = x(t_0) \quad \wedge \quad v_0 = v(t_0)$$

Rewrite as two coupled ODEs

$$v(t) = \frac{dx}{dt}$$

$$m\frac{dv}{dt} + \eta v + x = F$$

Discretize

$$t \to t_i = t_0 + i\,\Delta t$$

$$i = 0, 1, 2, \ldots n$$

$$\Delta t = \frac{t_m - t_0}{m}$$

$$x \Rightarrow x_i' \qquad v \Rightarrow v_i'$$

Euler's method

$$x_{i+1}' = x_i' + \Delta t\, v_i'$$

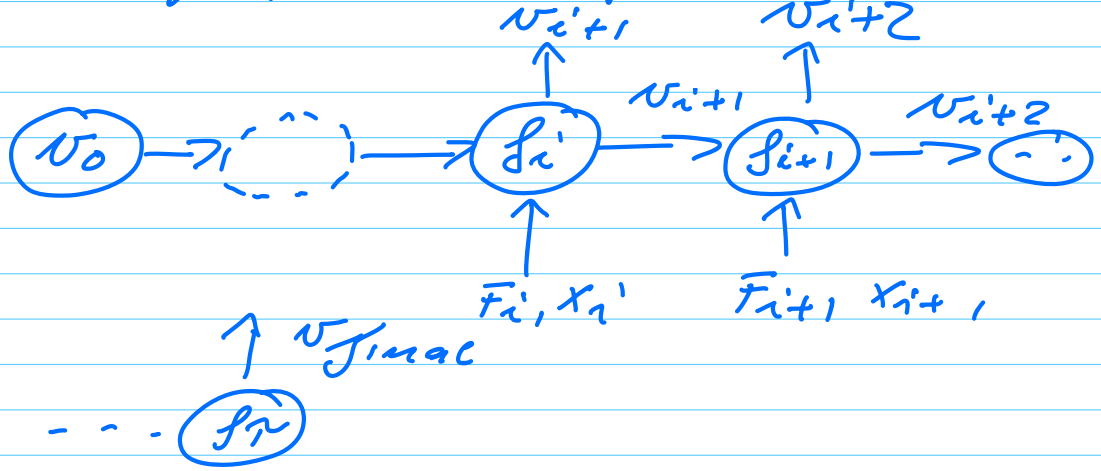
$$v_{i+1}' = v_i' + \Delta t\, a_i'$$

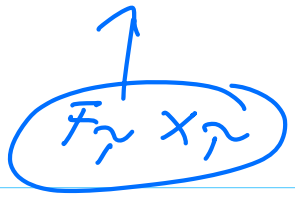
$$\frac{dv}{dt} = a \Rightarrow a_i' = \frac{F_c}{m} - \frac{\mu}{m} v_i' - \frac{1}{m} x_i'$$

$$v_{i+1}' = v_i' + \Delta t\, f(v_i', x_i', F_i')$$

$$v_{i+1}' = f\big(\underbrace{f(v_{i-1}', x_{i-1}', F_{i-1}')}_{f_i'}, v_i', F_i'\big)$$

Diagraphical representation





input from external node.

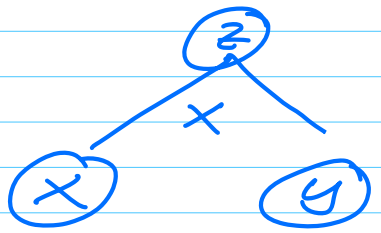
more general

$v_{i+1} \rightarrow s'_{i+1}$

$$s'_{i+1} = h(s'_i, x_i; \theta) = h_i$$

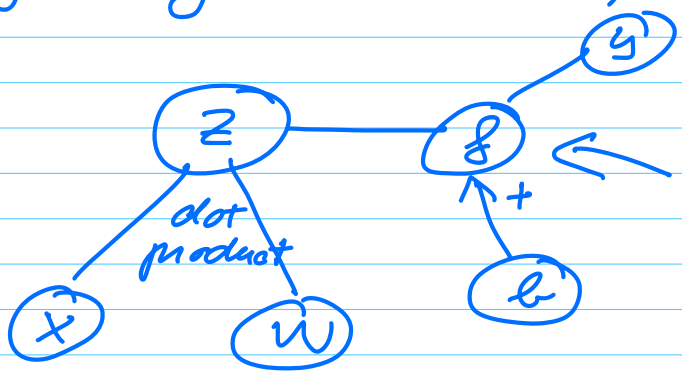
external inputs — parameters to train

Graphical representation

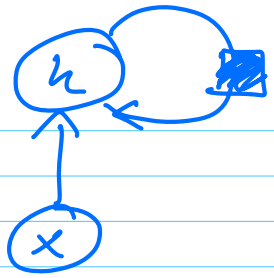


$\Rightarrow z = x \cdot y$

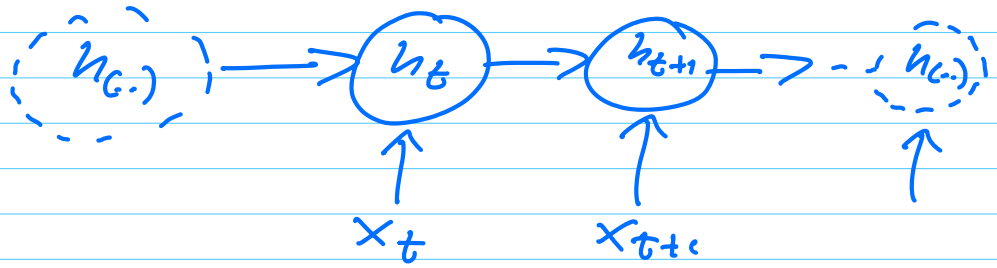
$$y = f(x^T w + b) = f(z)$$

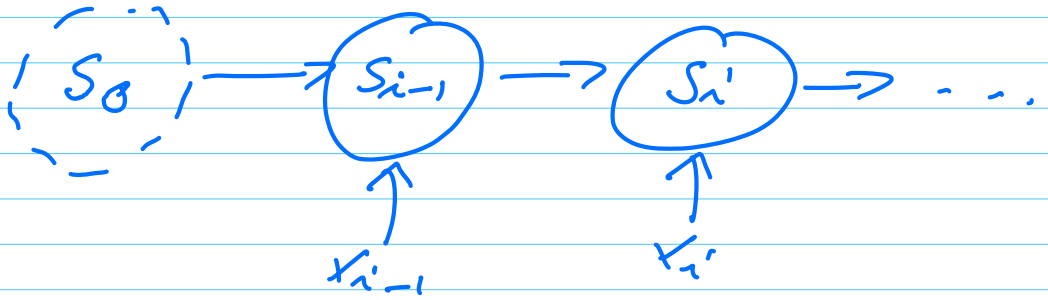


ReLU as example

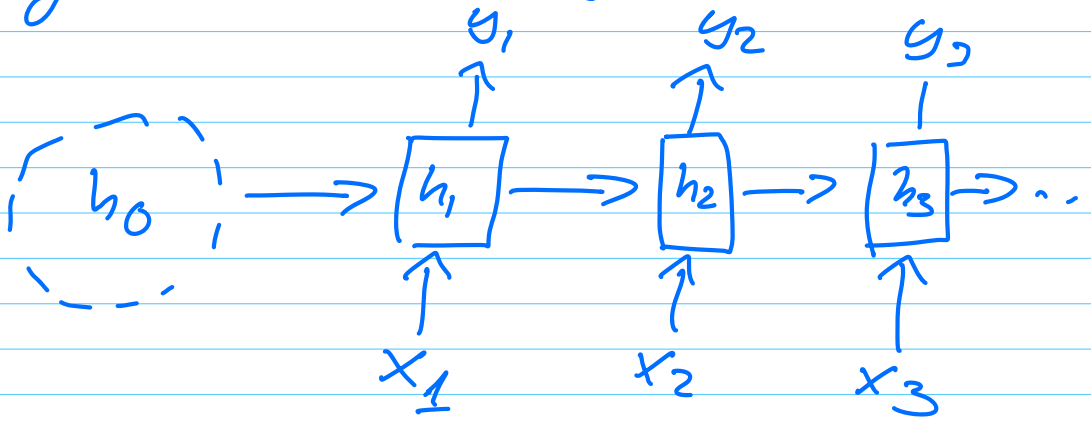


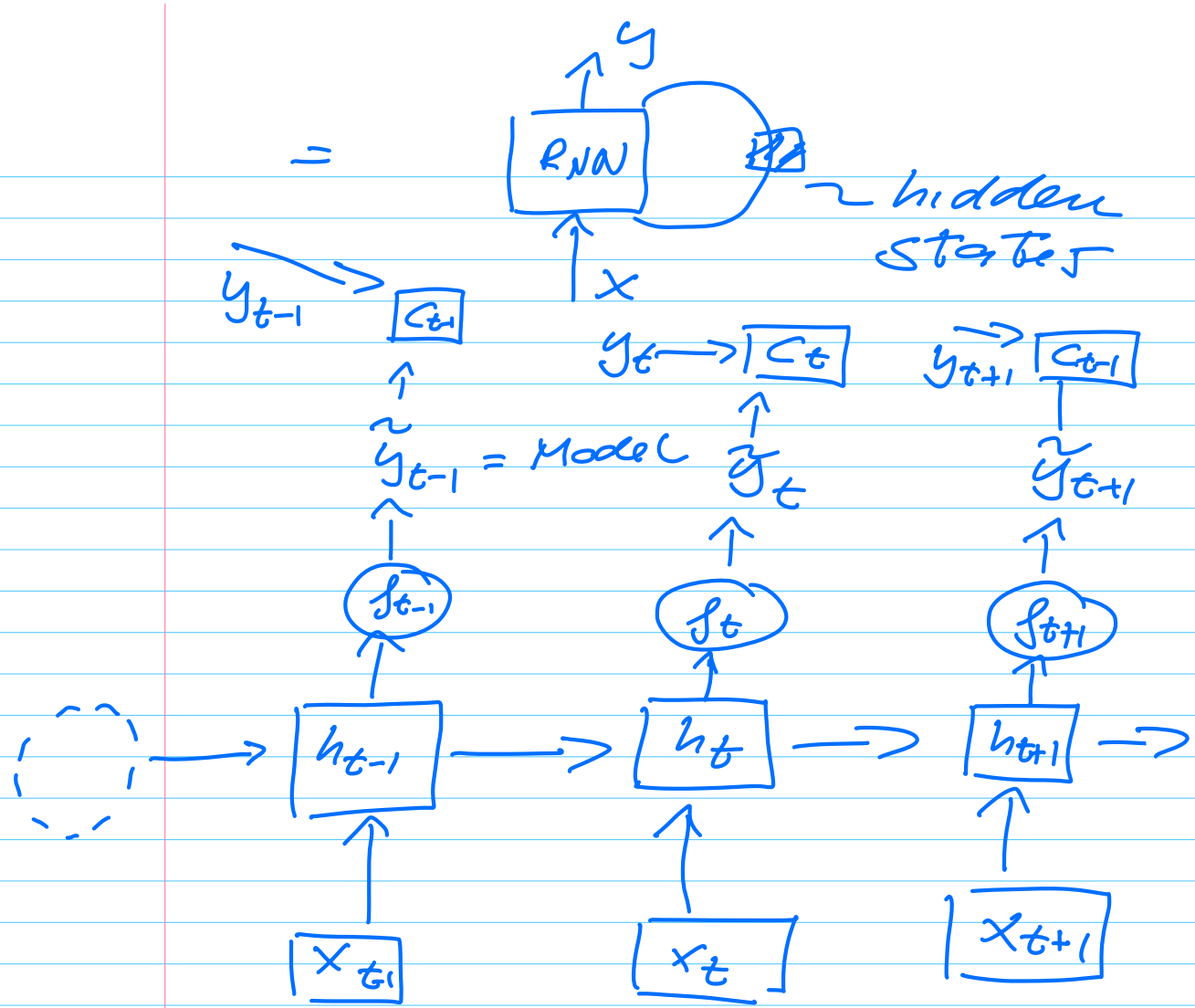
→ open up





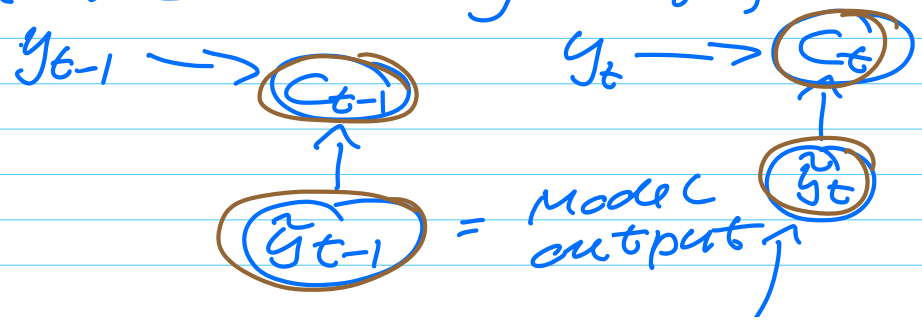
S_i' is given by a specific function h_i'

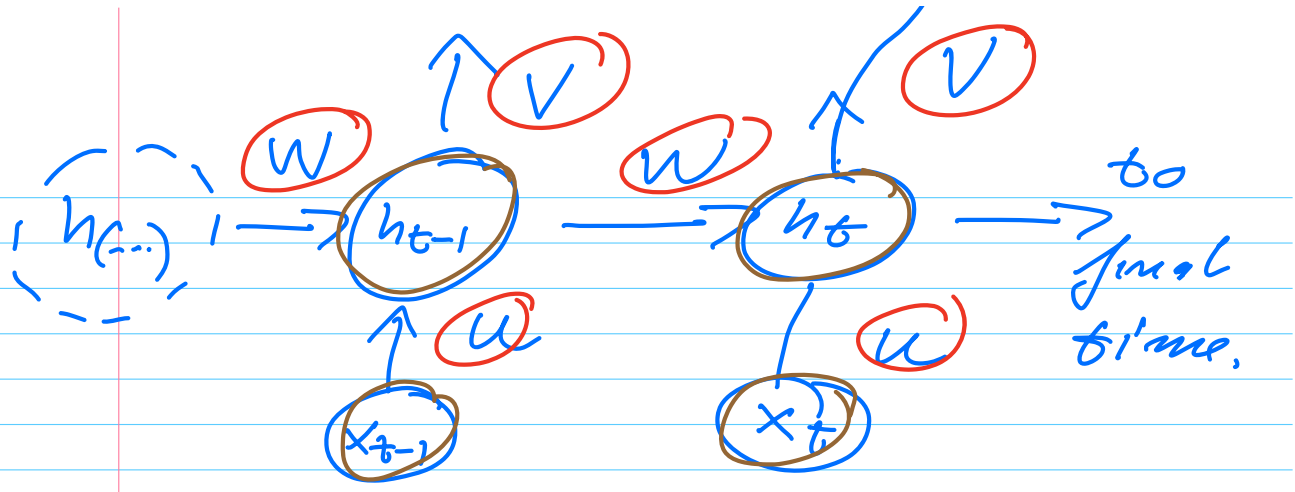




---> (final time)

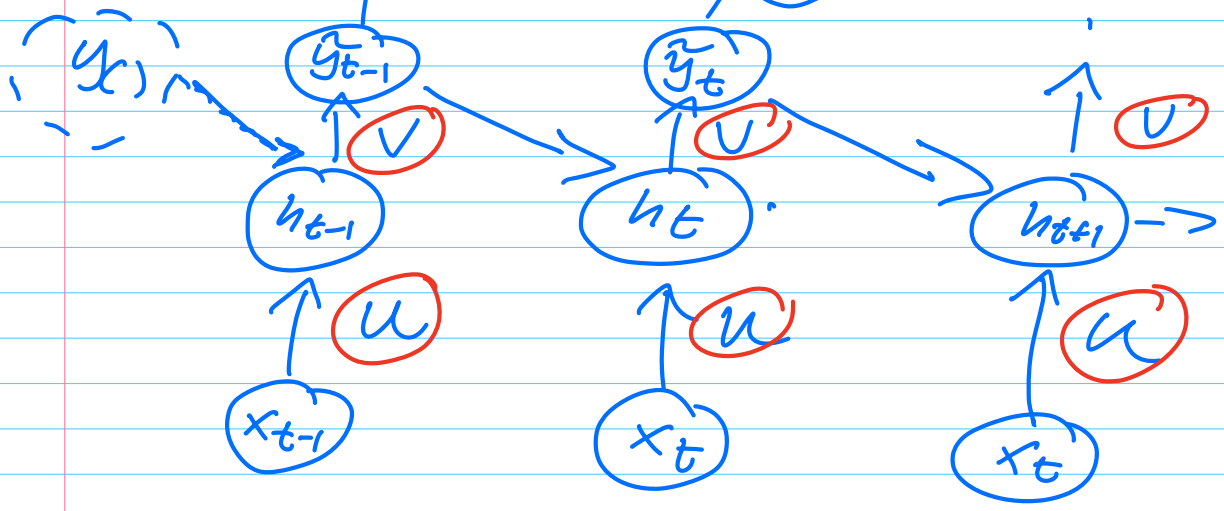
skip f_t for simplicity (absorbed by h_t)





Trained by backpropagation through time.
Impossible to parallelize, CPU and memory expensive

Slightly different architecture



Challenges of long-term dependencies -

$$h_t = W h_{t-1}$$

$h_t = (W)^t h_0$ (same weights)

$= W \cdot W \cdot W \cdots W \cdot h_0$

assume W is diagonalizable,

$W = U D U^T$ $\quad U^T U$

$= U U^T = \mathbb{1}$

eigenpairs of W are λ_i and w_i

expand

$$h_0 = \sum_i \alpha_i w_i$$

$$W \cdot h_0 = \sum_i \alpha_i \lambda_i w_i$$

$$W w_i = \lambda_i w_i$$

$$h_t = \sum_i W^t \alpha_i w_i$$

$$= \sum_i \lambda_i^t \alpha_i w_i$$

assume $\lambda_0 > \lambda_1 > \lambda_2 > \dots > \lambda_n$

$h_t \simeq \lambda_0^t w_0 a_0$

depending on the value of λ_0, we can get vanishing or exploding gradients.

To avoid exploding, a simple trick is to "clip" the gradient $\vec{g}$

if $\|\vec{g}\|_2 \geq \varepsilon$

$$\vec{g} \leftarrow \frac{\varepsilon}{\|\vec{g}\|_2} \vec{g}$$

end if.