# Lecture October 16

Recurrent Neural Networks (RNN)

Define a dynamical system

$$s^{(t)} = f(s^{(t-1)}; \theta)$$

output/state

set of parameters

$$\frac{ds}{dt} = g(s,t)$$

Euler's method

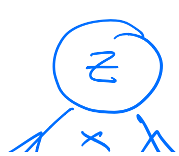
$$s^{(t+1)} = s^{(t)} + h\, g(s^{(t)}, t)$$

stepsize

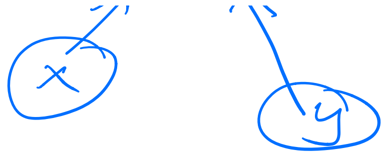
recurrent equation since $s$ at time $t+1$ refers back to the same definition at time $-t-$

## Computational graphs

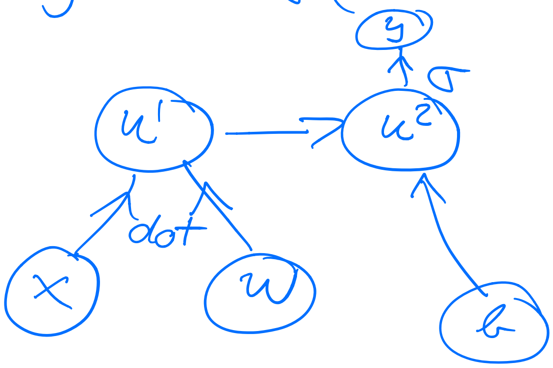
$z = x \cdot y$



(i)

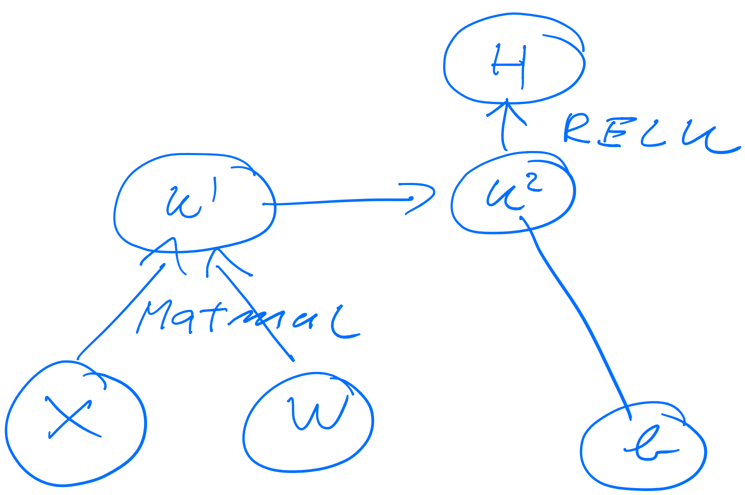



$$y = \sigma(x^T w + b)$$




(ii)

$$H = \max\{0, XW + b\}$$



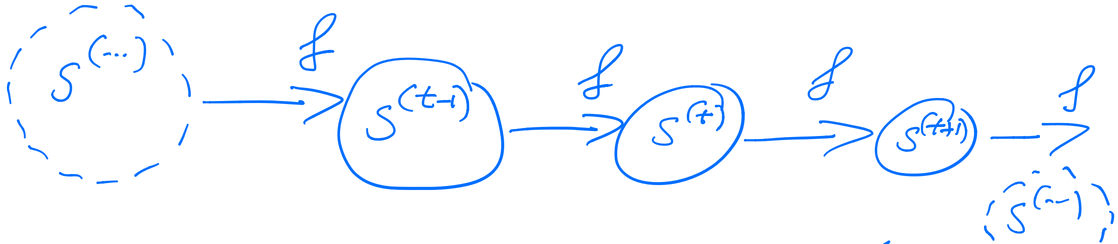

(iii)

Finite # of time steps $\tau$
from $t = 1$ to $t = \tau$

$\tau = 3$

$$s^{(3)} = f(s^{(2)}; \theta)$$

$$= f(f(s^{(1)}; \theta); \theta)$$

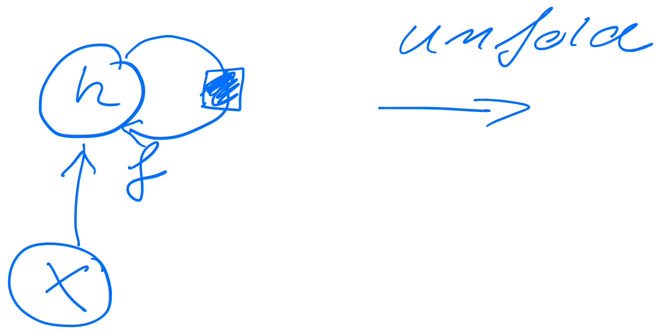


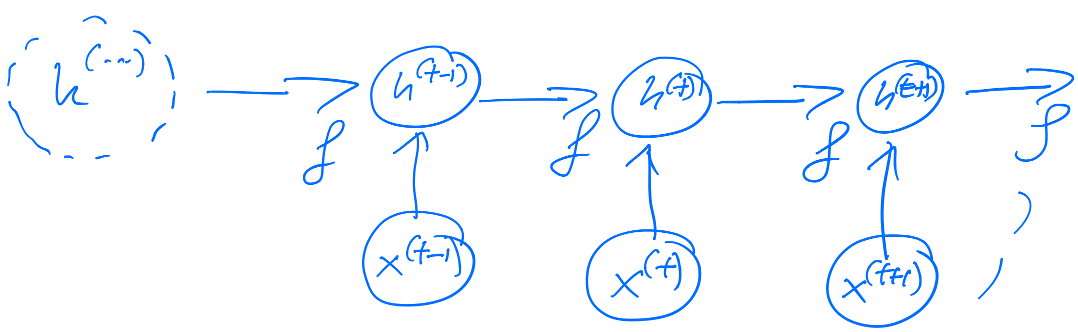

Dynamical system driven by an external signal $x^{(t)}$

$$s^{(t+1)} = f(s^{(t)}, x^{(t+1)}; \theta)$$

Define hidden layer

$$h^{(t+1)} = f(h^{(t)}, x^{(t+1)}; \theta)$$



$h^{(t-1)}$

$$h^{(t)} = f(x^{(t)}U + s^{(t-1)}W + b)$$

Example

$$m\frac{d^2s}{dt^2} = -ks + A\cos(t)$$

$$\begin{cases} \dfrac{ds}{dt} = v(s,t) \\[2mm] \dfrac{dv}{dt} = -\dfrac{k}{m}s + \dfrac{A}{m}\cos(t) \end{cases}$$

Euler's method

$$s_{t+1} = s_t + h \cdot v_t$$

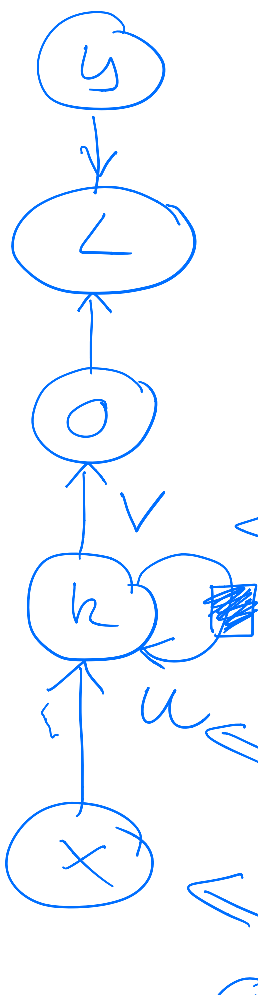
$$v_{t+1} = v_t + h\left(\frac{A}{m}\cos(t) - \frac{k}{m}s_t\right)$$

# Recurrent NN (RNN)

1st model : RNN that produces an output y at each time step and recurrent connections between hidden units

2nd model: RNN that produces an output at each time step and have recurrent connections from the output at one time step to the hidden unit at the next time step.

## 1st model




← output from our model

← cost/loss/risk function

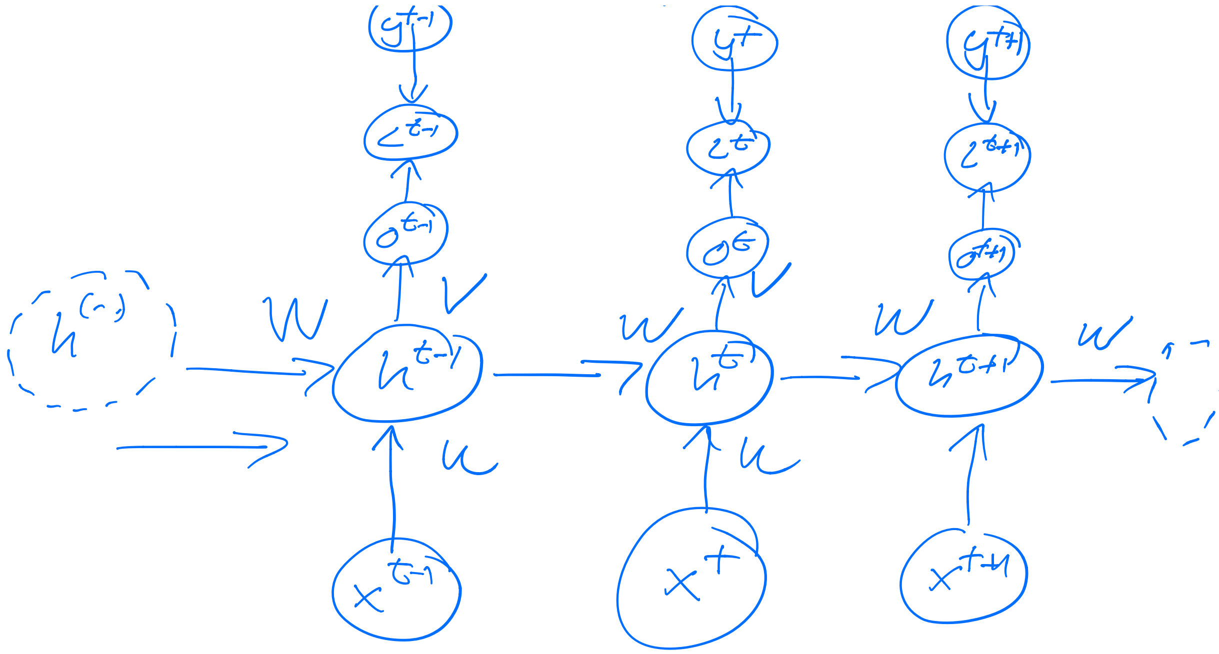
← $y$ = softmax($o$)
$\sigma(o)$, $\tanh(o)$ . . .

← hidden to output connections

← W = hidden-hidden recurrent connections

input-to hidden with weight

input values




$$a^{(t)} = b + W h^{(t-1)} + U x^{(t)}$$

$$h^{(t)} = \tanh(a^{(t)}) \quad (\sigma(a^{(t)}), \ \mathrm{ReLU}(a^{(t)}) \ldots)$$

$$y^{(t)} = \mathrm{softmax}(o^{(t)}) \, / \, \sigma(o^{(t)})$$

$$o^{(t)} = c + V h^{(t)}$$

with $y^{(t)}$ + target$^{(t)}$ we can calculate the Loss function $L$.

Need : $\nabla_h L$ , $\nabla_o L$, $\nabla_c L$

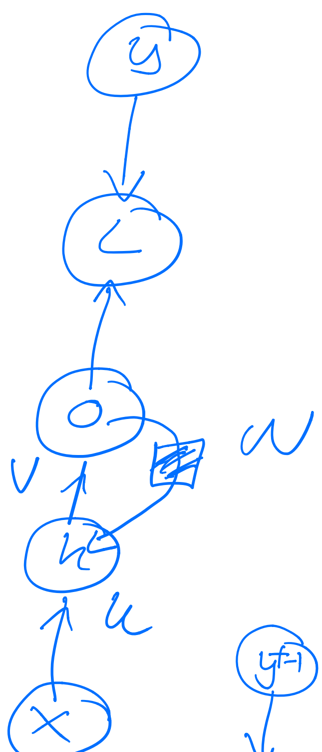
$\nabla_b L$, $\nabla_V L$ , $\nabla_W L$ , $\nabla_U L$

$x^{(t)}$ does not have any parameters.

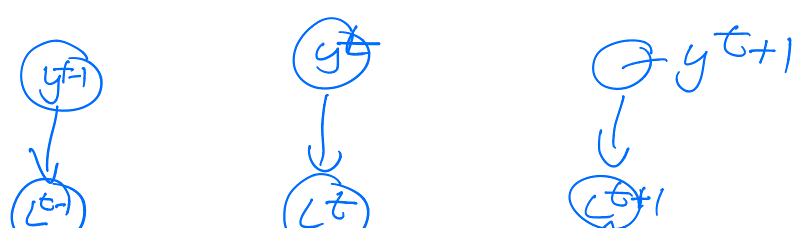
- Training
  - Feed forward from left to right
  - Back prop in time from the right to the left (BPTT)

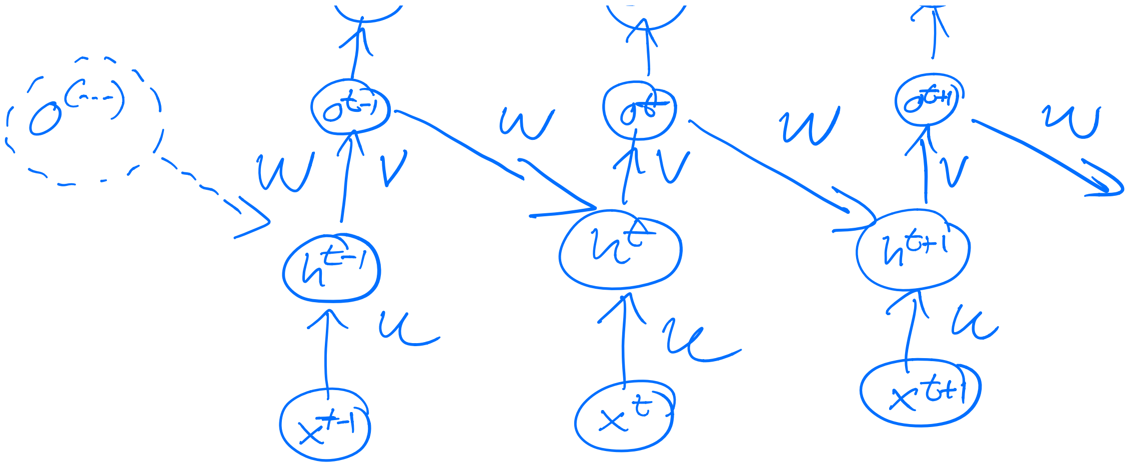
Expensive to train and difficult to parallelize.

## 2nd Model




Open up





$o^{(...)}$
$W$
$o^{t-1}$
$V$
$h^{t-1}$
$U$
$x^{t-1}$
$W$
$o^{t}$
$V$
$h^{t}$
$U$
$x^{t}$
$W$
$o^{t+1}$
$V$
$h^{t+1}$
$U$
$x^{t+1}$
$W$