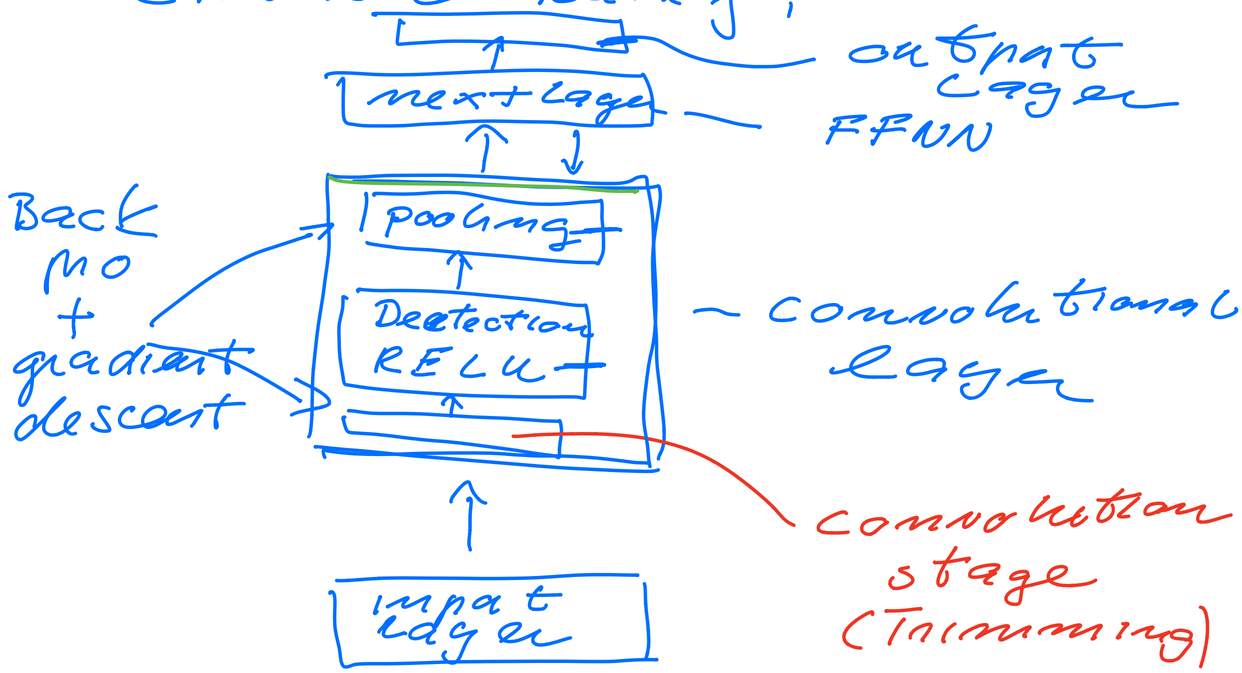
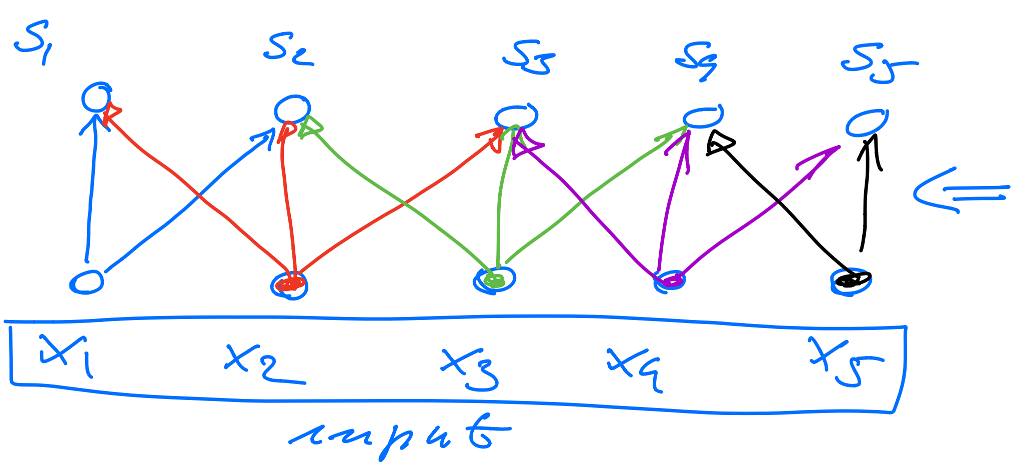
Lecture January 27

CNN in Brief:

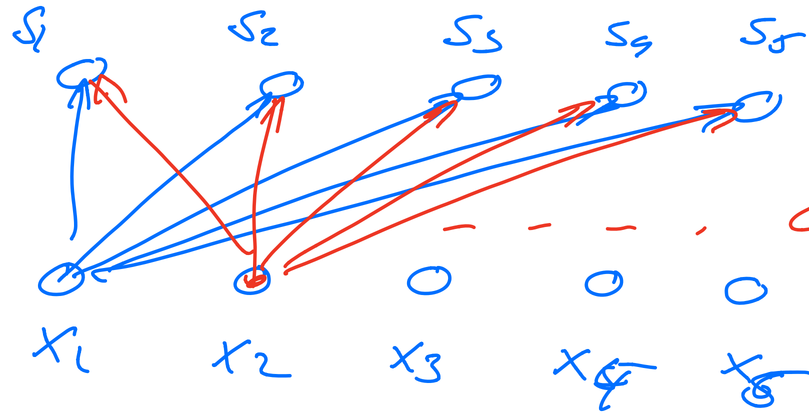


Convolutional stage

Kernel < Input

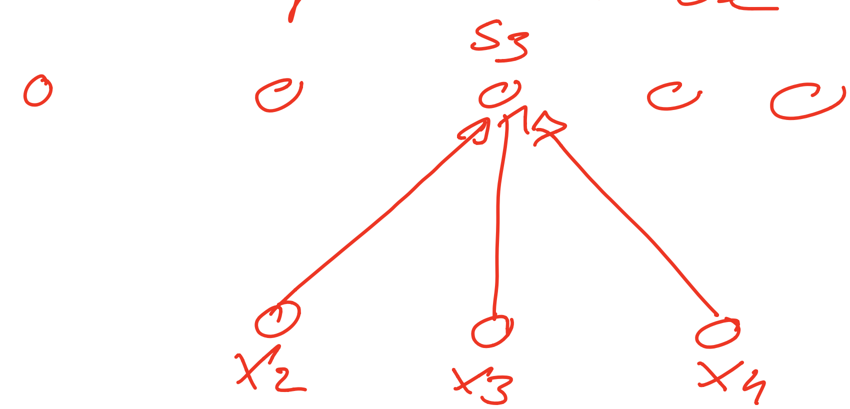


Standard FFNN

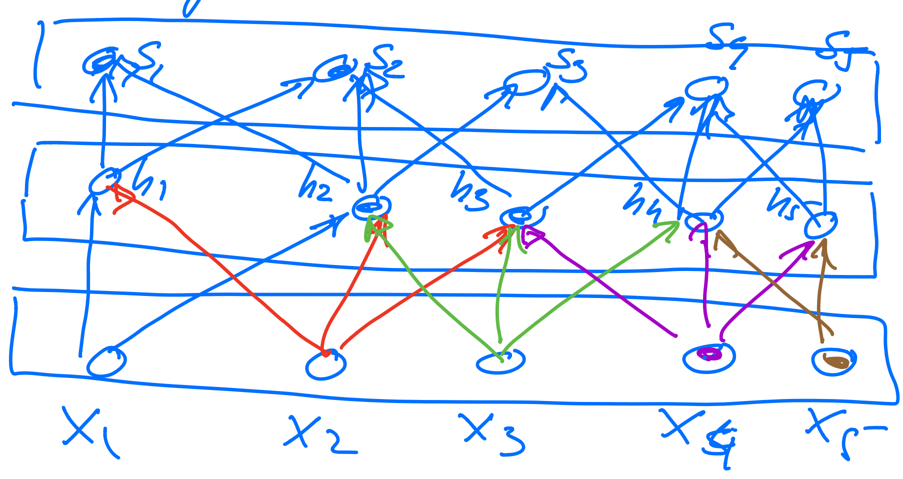


all connected

seen from above



Deeper

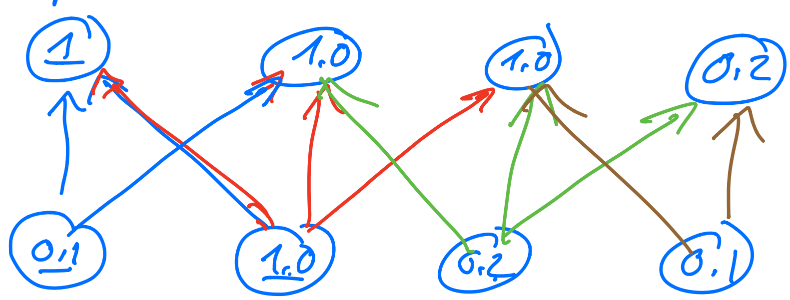


- with inputs to s_i
  RELU (or other activation
  functions) etc.

functions) - stage,

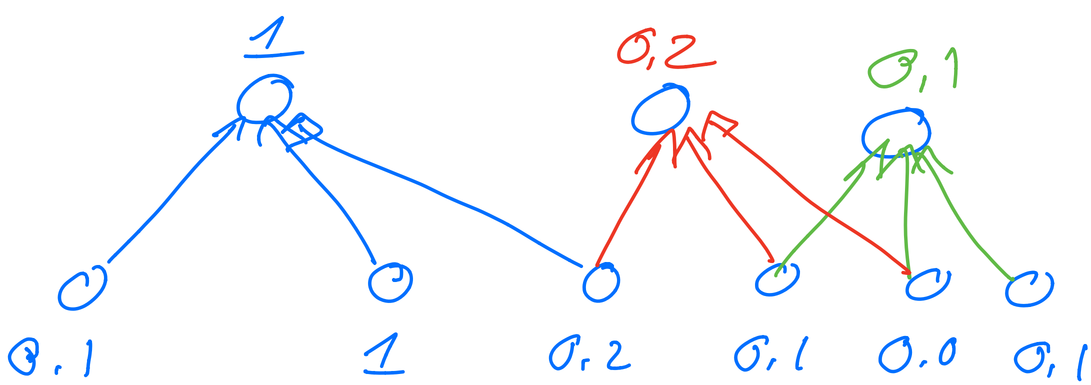
- Pooling stage
  a pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs

- Max pooling



Detector stage with ReLU →

pooling reduces dimensionality



$$H = -\underline{J} \sum_{k=1}^{N} s_k s_{k+1}$$

configurations $= 2^N$

spins $= 4 \Rightarrow 2^{40}$ configs.

10000 and compute

$\underline{H_1}, H_2, \ldots, H_{10000}$ (fixed $J = \underline{1}$)

To use ML to find J ?

Recurrent NN:

Dynamical system:

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

parameters

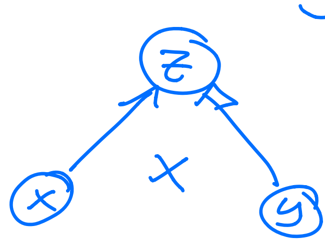
$$\frac{ds}{dt} = f(s, t)$$

Euler's method:

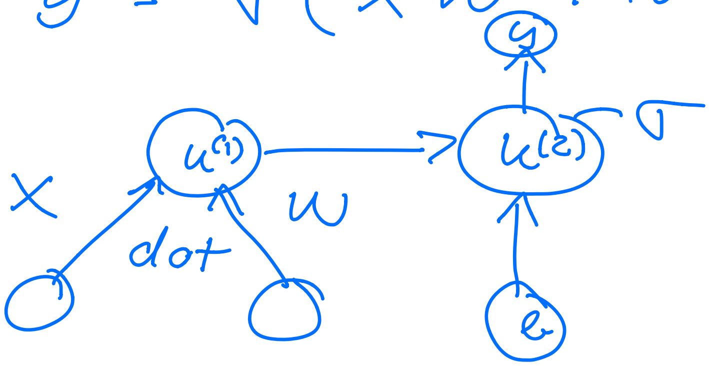
$$s_{t+1} = s_t + \Delta t \, f(s_t, t)$$

Computational graphs:

$$z = x \cdot y$$



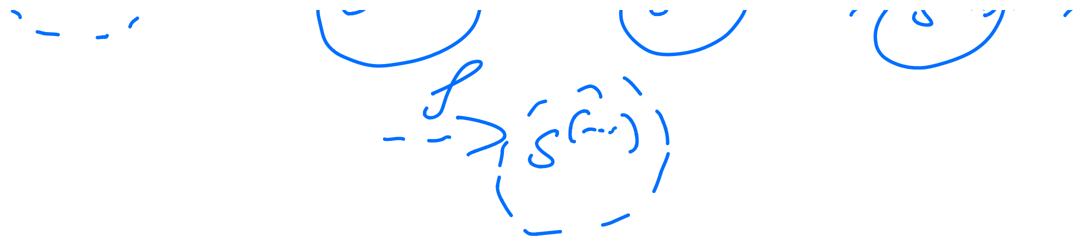
$$y = \sigma(XW + b)$$



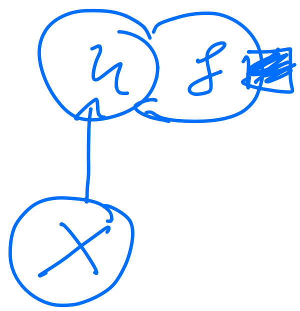
$$\tau = 3$$

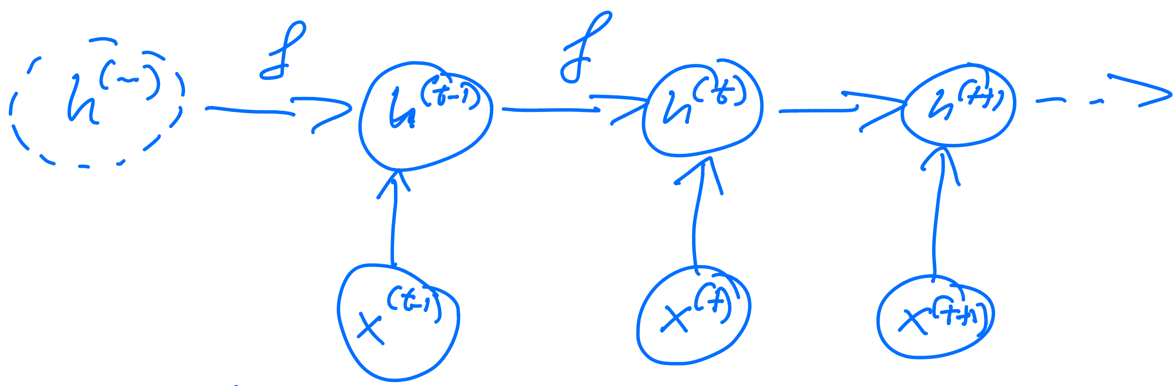
$$s^{(3)} = f(s^{(2)}; \theta^{(3)})$$

$$= f(f(s^{(1)}; \theta^{(1)}); \theta^{(2)})$$

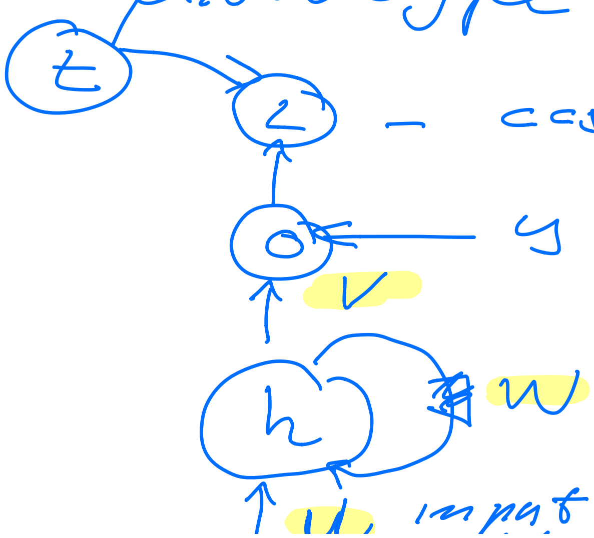


$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$$





prototype RNN (compact)



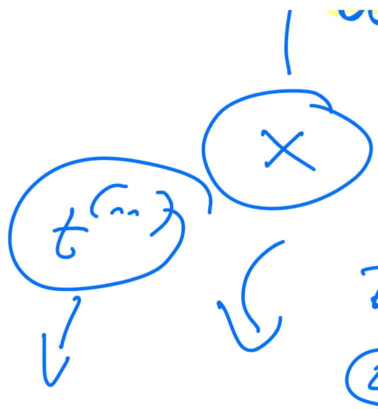
L – cost/loss function

y softmax, ...

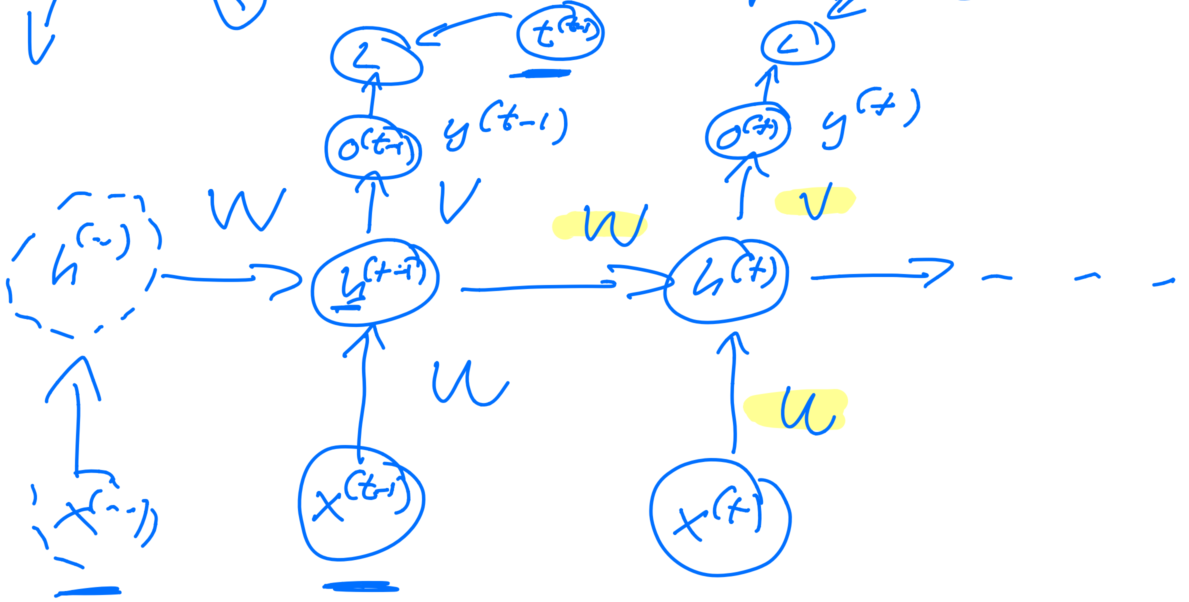
W hidden to hidden recurrence connections

U input to

hidden
layer with weights
input
sequence

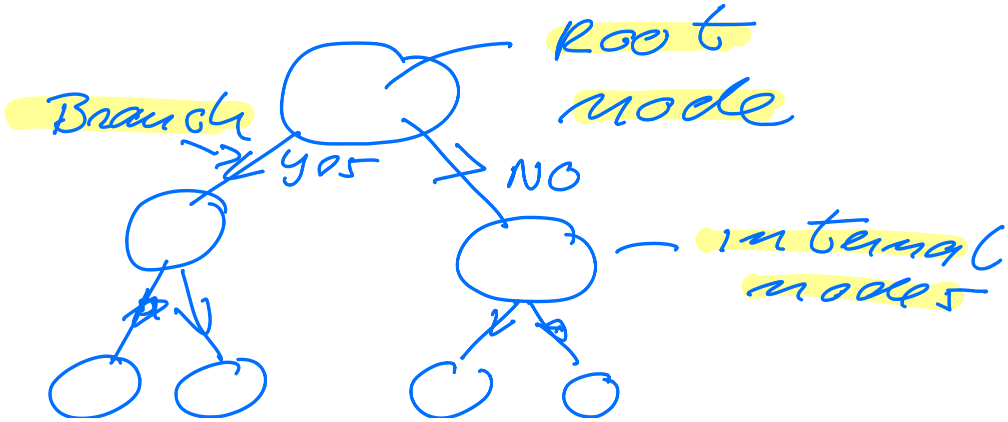


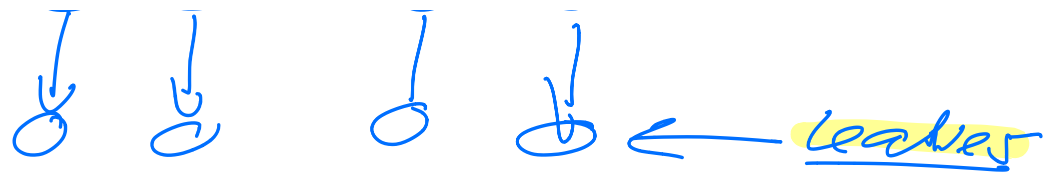
Break up



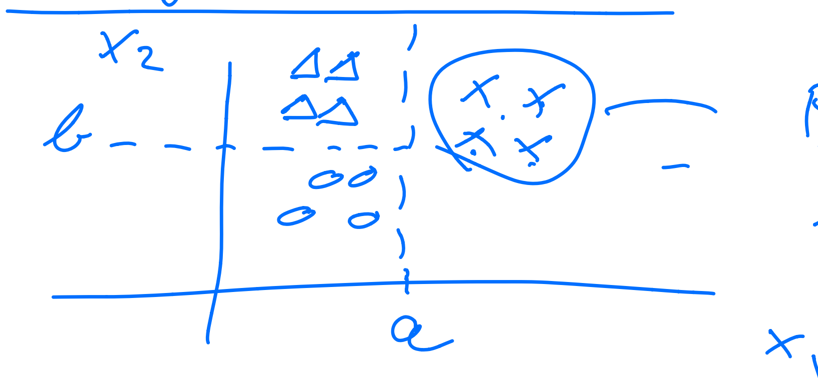
Decision Trees, random forests, bagging, Boosting

– Decision tree





Regression



prediction is mean value in each domain

if $x_1 \geq a$ (x)

if $x_1 < a$ (o, Δ)

if $x_2 \geq b$ (Δ)

else $< b$ (o)