CS626: Speech, NLP and the Web

Delving into Feedforward Network and Backpropagation; Applications of FFNN in NLP

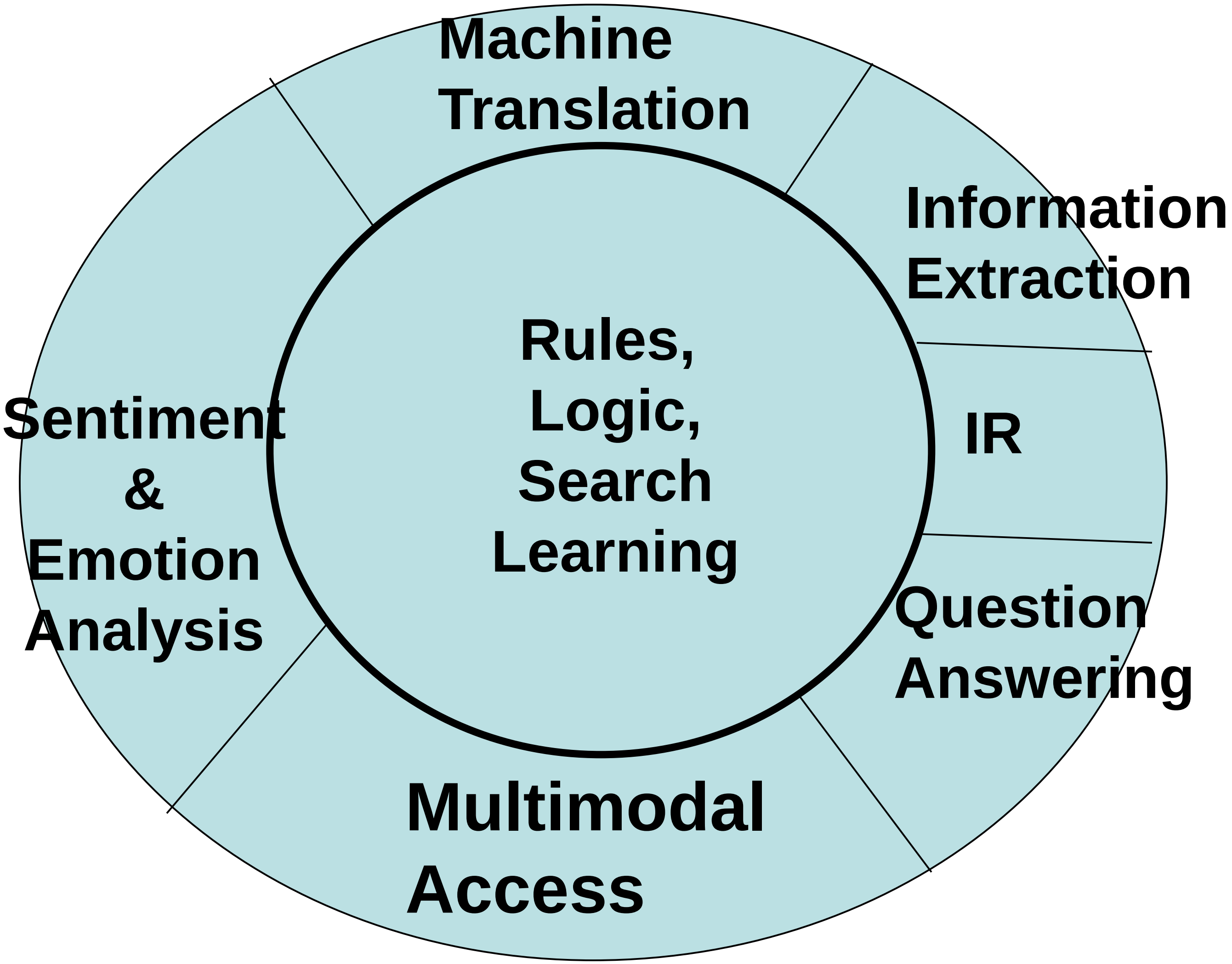
Pushpak Bhattacharyya

Computer Science and Engineering Department

IIT Bombay

Week of 19th October, 2020

NLP Areas



Machine Translation

Information Extraction

IR

Question Answering

Multimodal Access

Sentiment & Emotion Analysis

Rules, Logic, Search Learning

Task *vs.* Technique Matrix

Task (row) vs. Technique (col) Matrix	**Rules Based/Knowledge-Based**	**Classical ML**				**Deep Learning**		
		Perceptron	Logistic Regression	SVM	Graphical Models (HMM, MEMM, CRF)	Dense FF with BP and softmax	RNN-LSTM	CNN
Morphology								
POS								
Chunking								
Parsing								
NER, MWE								
Coref								
WSD								
Machine Translation								
Semantic Role Labeling								
Sentiment								
Question Answering								

Agenda for the week

- Delving deeper into Feedforward N/W an Backpropagation
- Applications of FFNN-BP
- Start of Sequence Processing
- Recurrent N/W

Application of FFNN-BP

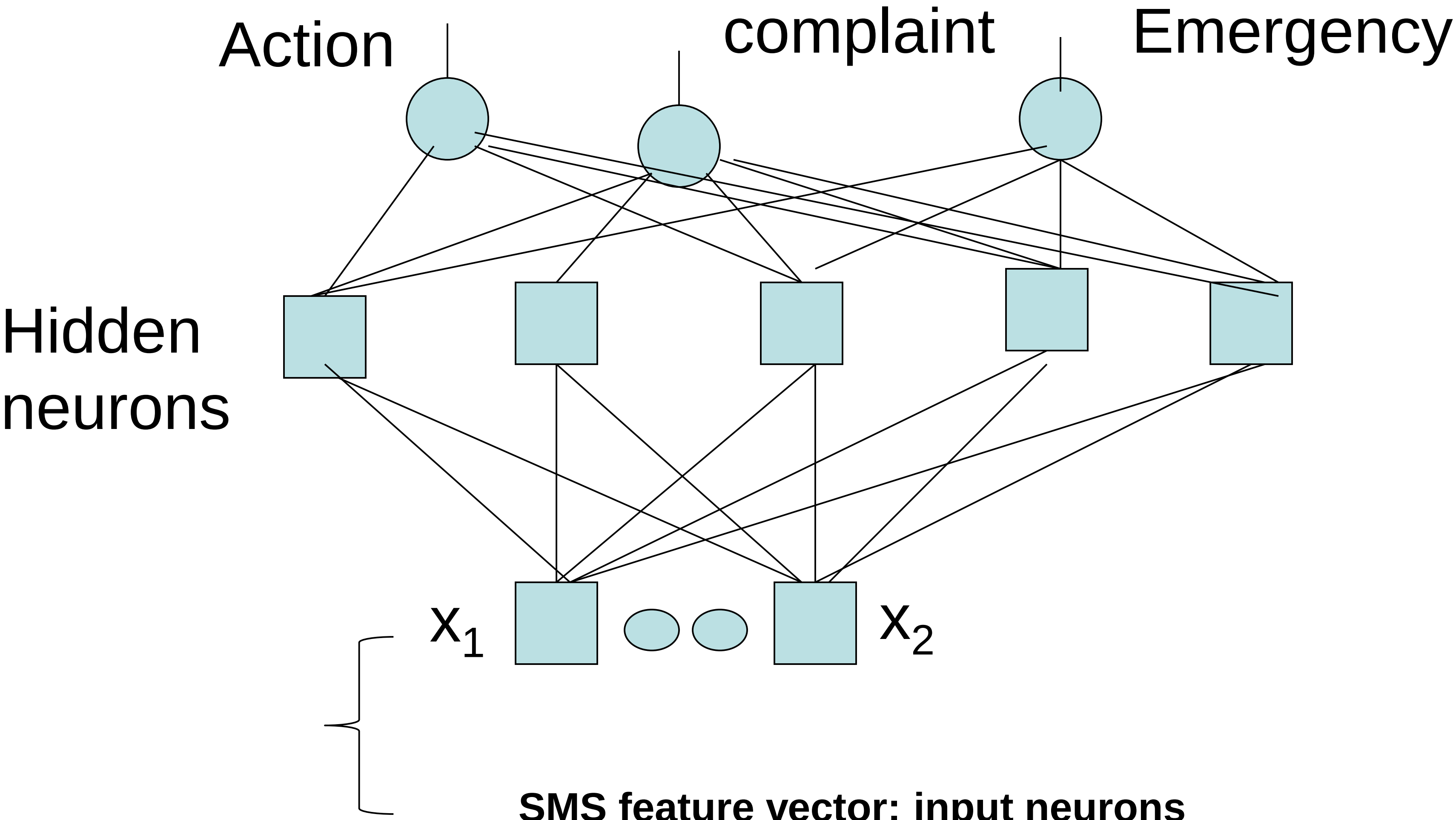
An example SMS complaint

- I have purchased a 80 litre Videocon fridge about 4 months ago when the freeze go to sleep that time compressor give a sound (khat khat khat khat ....) what is possible fault over it is normal I can't understand please help me give me a suitable answer.

Significant words (in red): after stop word removal

- I have purchased a 80 litre Videocon fridge about 4 months ago when the freeze go to sleep that time compressor give a sound (khat khat khat khat ....) what is possible fault over it is normal I can't understand please help me give me a suitable answer.

SMS classification



SMS feature vector: input neurons

Sequence Processing v/s Whole-Text processing

- Preemptive actions: possible in sequence processing
  - E.g., Surveillance
- Micro-information: better handled in seq processing
- Global information: better handled in whole text processing
- Chat-bot: inherently seq processing

An application in Medical Domain

Expert System for Skin Diseases Diagnosis

- Bumpiness and scaliness of skin
- Mostly for symptom gathering and for developing diagnosis skills
- Not replacing doctor’s diagnosis

Architecture of the FF NN

- 96-20-10
- 96 input neurons, 20 hidden layer neurons, 10 output neurons
- Inputs: skin disease symptoms and their parameters
  - *Location, distribution, shape, arrangement, pattern, number of lesions, presence of an active norder, amount of scale, elevation of papuls, color, altered pigmentation, itching, pustules, lymphadenopathy, palmer thickening, results of microscopic examination, presence of herald pathc, result of dermatology test called KOH*

Output

- 10 neurons indicative of the diseases:
  - *psoriasis, pityriasis rubra pilaris, lichen planus, pityriasis rosea, tinea versicolor, dermatophytosis, cutaneous T-cell lymphoma, secondery syphilis, chronic contact dermatitis, soberrheic dermatitis*

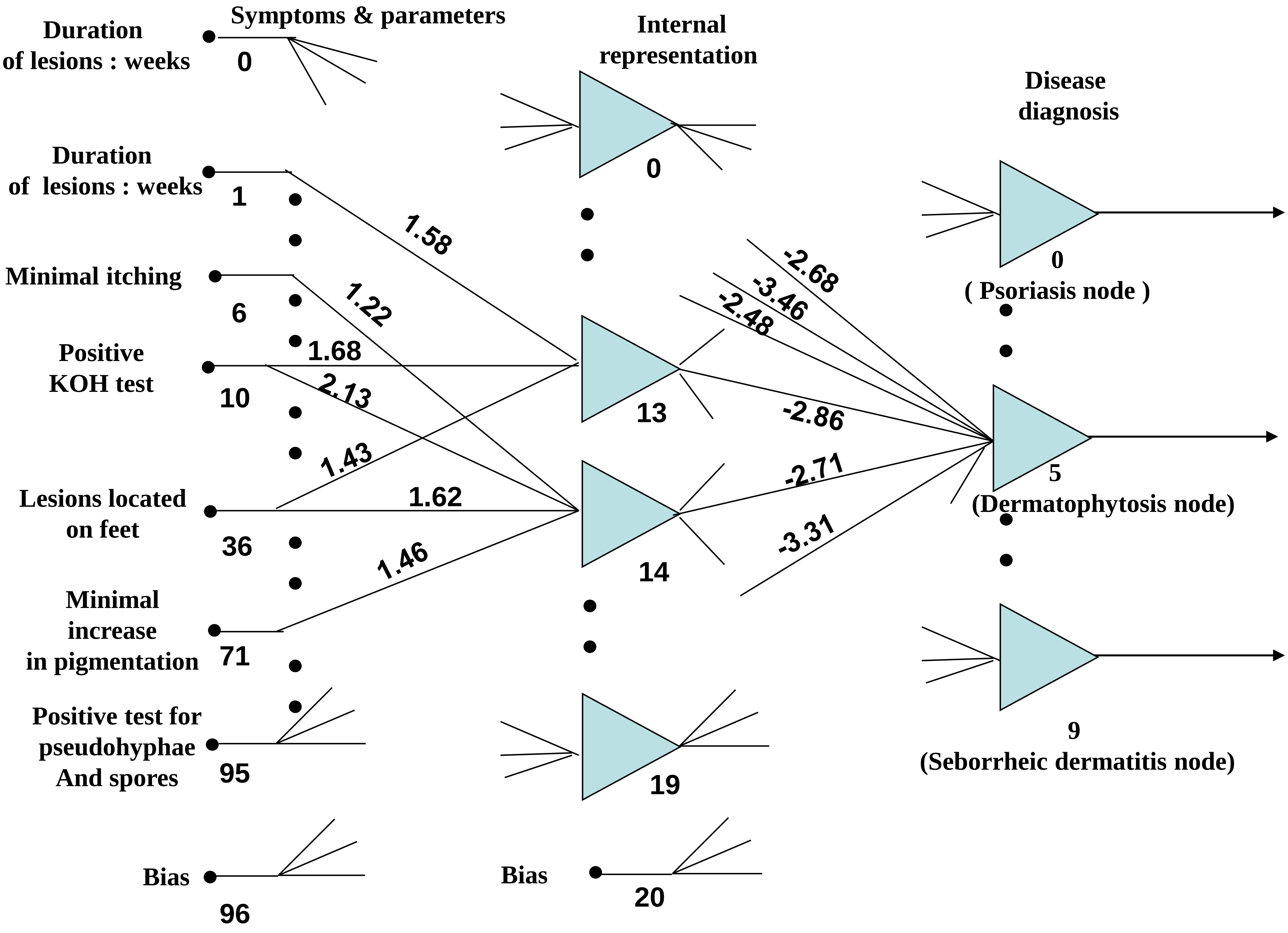


Figure : Explanation of dermatophytosis diagnosis using the DESKNET expert system.

Training data

- Input specs of 10 model diseases from 250 patients
- 0.5 is some specific symptom value is not known
- Trained using standard error backpropagation algorithm

Testing

- Previously unused symptom and disease data of 99 patients
- Result:
- Correct diagnosis achieved for 70% of papulosquamous group skin diseases
- Success rate above 80% for the remaining diseases except for psoriasis
- psoriasis diagnosed correctly only in 30% of the cases
- Psoriasis resembles other diseases within the papulosquamous group of diseases, and is somewhat difficult even for specialists to recognise.

Explanation capability

- Rule based systems reveal the explicit path of reasoning through the textual statements
- Connectionist expert systems reach conclusions through complex, non linear and simultaneous interaction of many units
- Analysing the effect of a single input or a single group of inputs would be difficult and would yield incorrect results

Explanation contd.

- The hidden layer re-represents the data
- Outputs of hidden neurons are neither symtoms nor decisions

Discussion

- Symptoms and parameters contributing to the diagnosis found from the n/w
- Standard deviation, mean and other tests of significance used to arrive at the importance of contributing parameters
- The n/w acts as apprentice to the expert

Hardmax v/s Softmax

- V-verb, N-noun, J-adjective, R-adverb, O-others
- Hardmax
  - – Given an input, if the output is 0 or 1.

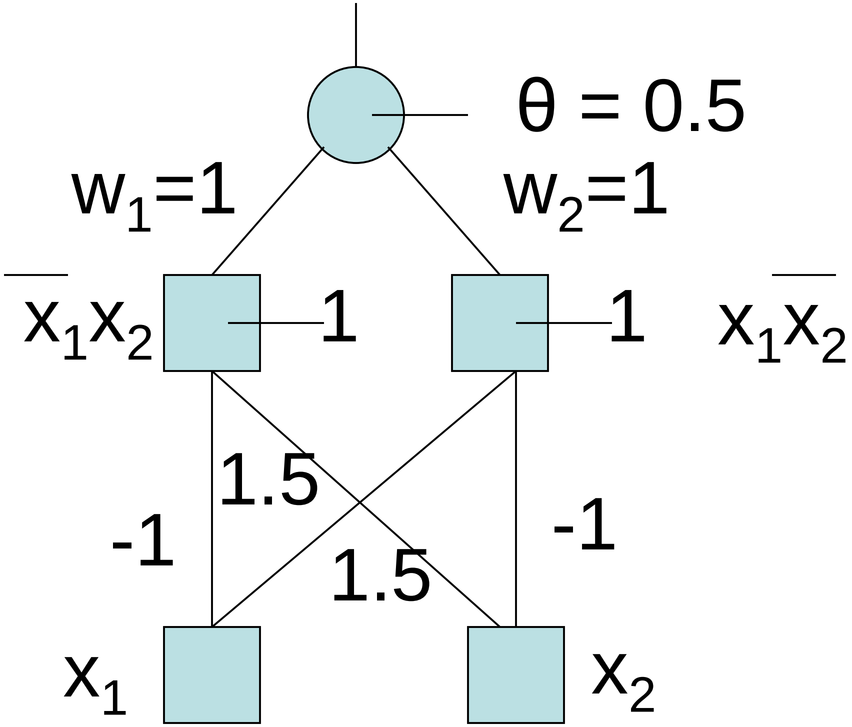
  <V, N, J, R, O> ➔ <0, **1**, 0, 0, 0>

- Softmax
  - – Given an input, if it belongs to multiple label/class, then the output is maximum of all labels/classes.

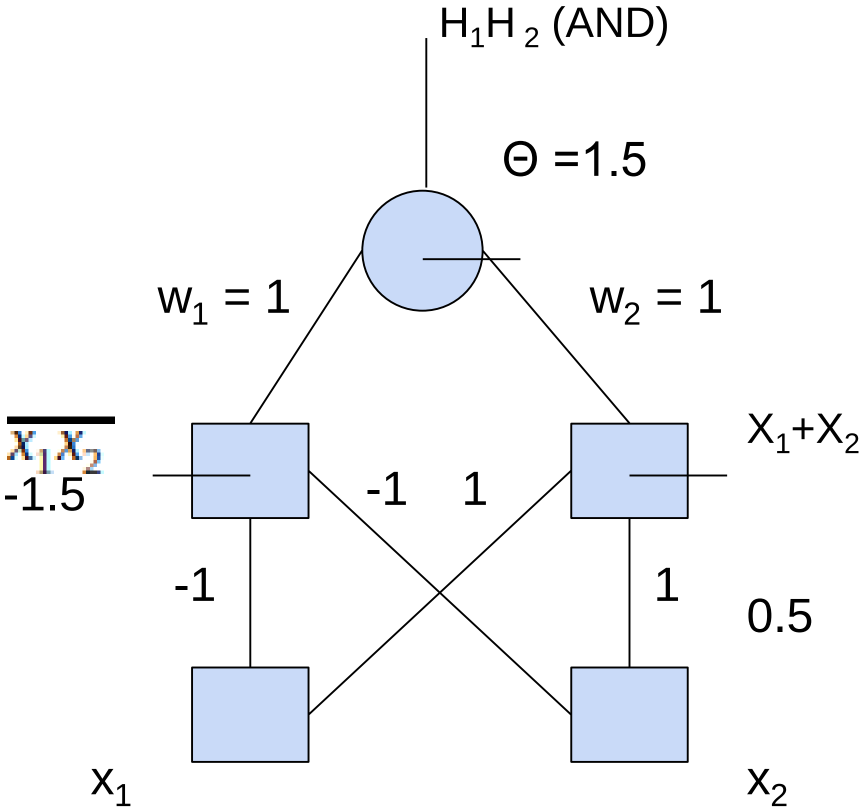
  <V, N, J, R, O> ➔ <0.1, **0.8**, 0.05, 0.02, 0.03>

Feedforward Network and Backpropagation

Example - XOR



Alternative network for XOR



- XOR: not possible using a single perceptron
- Hidden layer gives more computational capability
- Deep neural network: With multiple hidden layers
- Kolmogorov’s theorem of equivalence proves equivalence of multiple layer neural network to a single layer neural network, and each neuron have to correspond to an appropriate functions.

Exercise: Back-propagation

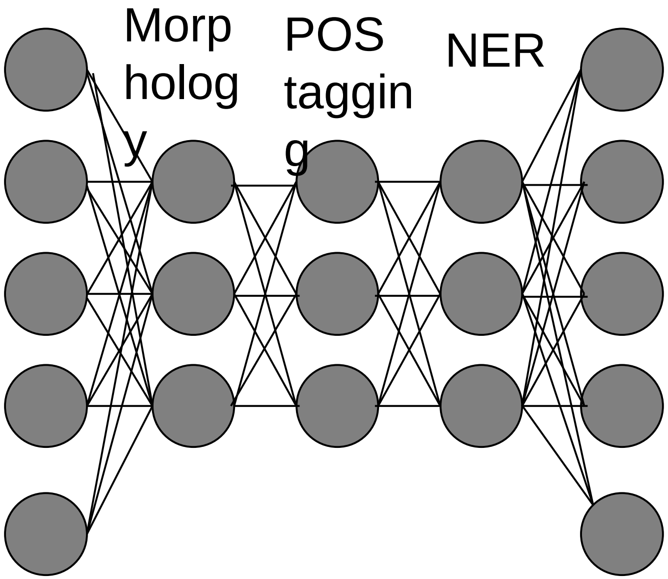
- Implement back-propagation for XOR network

- Observe
  - Check if it converges (error falls below a limit)

  - What is being done at the hidden layer

What a neural network can represent in NLP: Indicative diagram

- Each layer of the neural network possibly represents different NLP stages!!



Batch learning versus Incremental learning

- Batch learning is updating the parameters after ONE PASS over the whole dataset
- Incremental learning updates parameters after seeing each PATTERN
- An epoch is ONE PASS over the entire dataset
  - Take XOR: data set is $V_1=(<0,0>, 0)$, $V_2=(<0,1>, 1)$, $V_3=(<1,0>, 1)$, $V_4=(<1,1>, 0)$
  - If the weight values are changed after each of Vi, then this is incremental learning
  - If the weight values are changed after one pass over all V_is, then it is bathc learning

Can we use PTA for training FFN?

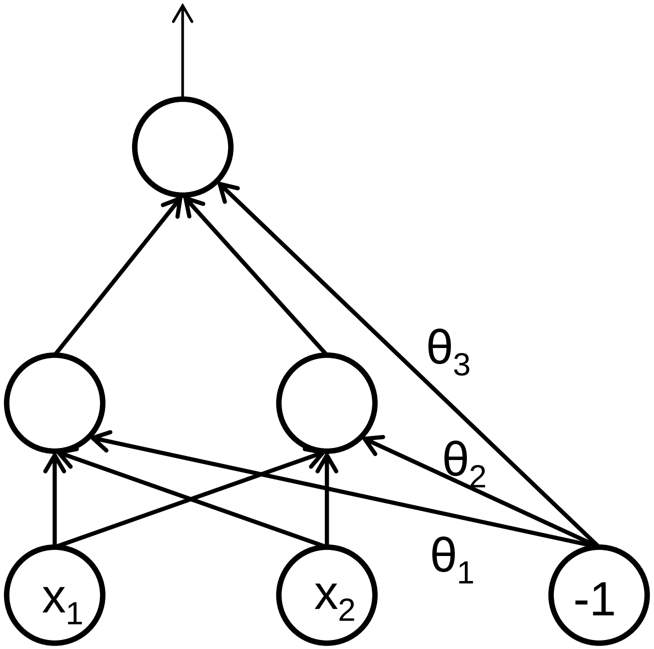
0, 0	0
0, 1	1
1, 0	1
1, 1	0

⇒

-1, 0, 0	0
-1, 0, 1	1
-1, 1, 0	1
-1, 1, 1	0

⇒

1, 0, 0	0
-1, 0, 1	1
-1, 1, 0	1
1, -1, -1	0



No, else the individual neurons are solving XOR, which is impossible.
Also, for the hidden layer neurons we do nothave the i/o behaviour.

Gradient Descent Technique

- Let E be the error at the output layer

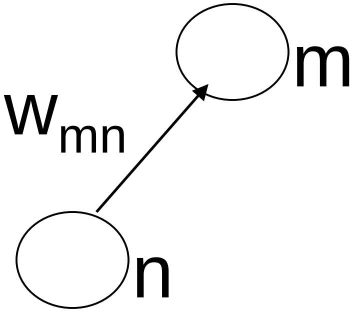
$$E = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{n}(t_i - o_i)_j^2$$

- t_i = target output; o_i = observed output

- i is the index going over n neurons in the outermost layer
- j is the index going over the p patterns (1 to p)
- Ex: XOR:– p=4 and n=1

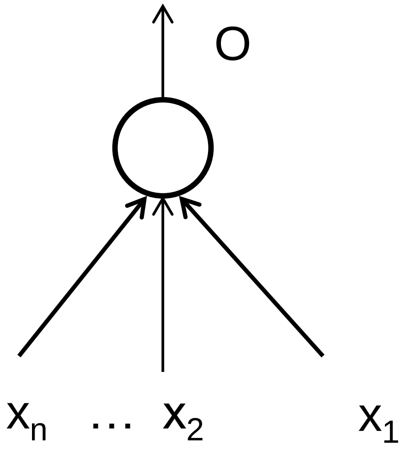
Weights in a FF NN

- w_{mn} is the weight of the connection from the nth neuron to the mth neuron
- E *vs* $\overline{W}$ surface is a complex surface in the space defined by the weights w_{ij}
- $-\frac{\delta E}{\delta w_{mn}}$ gives the direction in which a movement of the operating point in the w_{mn} co-ordinate space will result in maximum decrease in error



$$\Delta w_{mn} \propto -\frac{\delta E}{\delta w_{mn}}$$

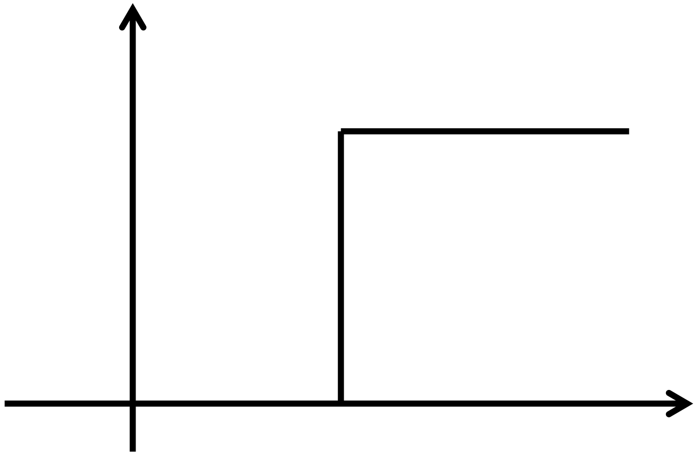
Step function v/s Sigmoid function



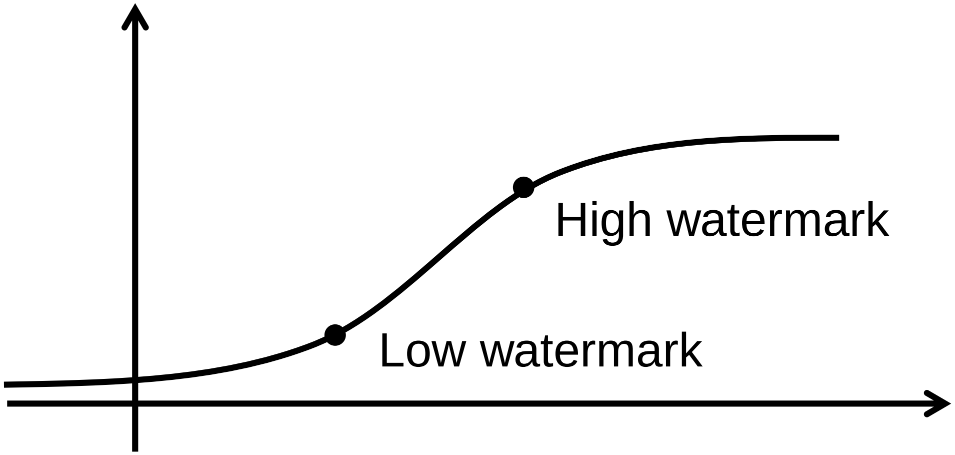
$$\begin{aligned} O &= f\left(\sum w_i x_i\right) \\ &= f(net) \end{aligned}$$

So partial derivative of O w.r.t. net is

$$\frac{\delta O}{\delta net}$$



Non-differentiable



Differentiable

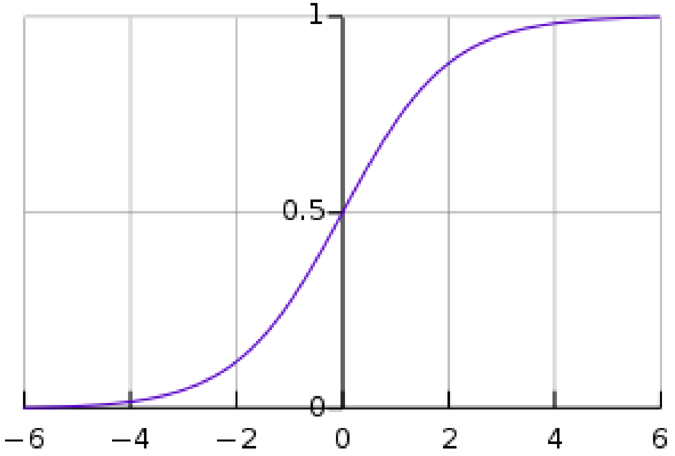
Sigmoid function

$$y = \frac{1}{1+e^{-x}}$$

$$y = \frac{1}{1+e}$$

$$\frac{dy}{dx} = y(1-y)$$

Sigmoid function



$$f(x) = \frac{1}{1+e^{-x}}$$

$$\begin{aligned}
f(x) &= \frac{1}{1+e^{-x}} \\
\frac{df(x)}{dx} &= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) \\
&= \frac{e^{-x}}{(1+e^{-x})^{-2}} \\
&= \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) \\
&= f(x).(1 - f(x))
\end{aligned}$$

Loss function

Total sum squared error

$$E = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{n}(t_i - o_i)_j^2$$

- T_i is the target output
- O_i is the observed output
- i and j are indices over n neurons and p patterns respectively

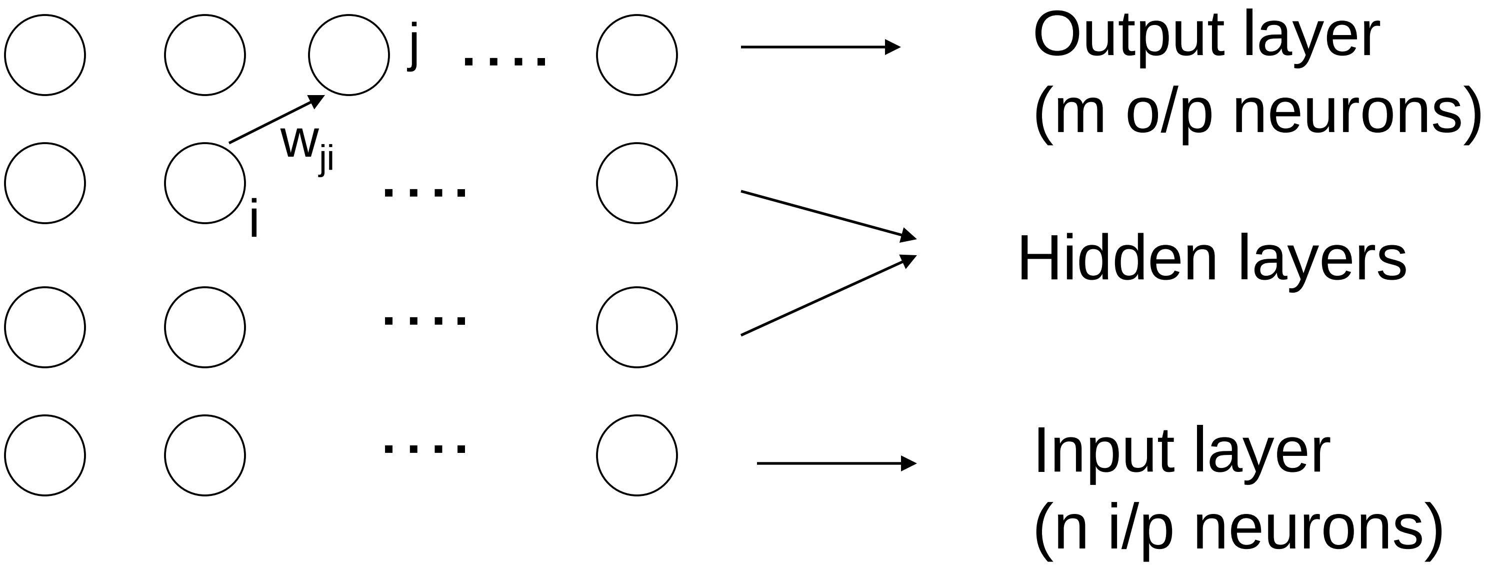
Cross-entropy

$$E = -(1/N) * \sum_{i=1}^{N} y_i \, log(\bar{y}_i)$$

- y_i is the target output
- $\bar{y}_i$ is the observed output
- N is number of training samples

Cross-entropy is used more in NLP than total sum squared error

Backpropagation algorithm



- Fully connected feed forward network
- Pure FF network (no jumping of connections over layers)

Gradient Descent Equations

$$\Delta w_{ji} = -\eta \frac{\delta E}{\delta w_{ji}} (\eta = \text{learning rate}, 0 \leq \eta \leq 1)$$

$$\frac{\delta E}{\delta w_{ji}} = \frac{\delta E}{\delta net_j} \times \frac{\delta net_j}{\delta w_{ji}} (net_j = \text{input at the j}^{th} \text{ neuron})$$

$$\frac{\delta E}{\delta net_j} = -\delta j$$

$$\Delta w_{ji} = \eta \delta j \frac{\delta net_j}{\delta w_{ji}} = \eta \delta j o_i$$

Backpropagation – for outermost layer

$$\delta j = -\frac{\delta E}{\delta net_j} = -\frac{\delta E}{\delta o_j} \times \frac{\delta o_j}{\delta net_j} (net_j = \text{input at the j}^{th}\text{ layer})$$

$$E = \frac{1}{2}\sum_{p=1}^{m}(t_p - o_p)^2$$

$$\text{Hence, } \delta j = -(-(t_j - o_j)o_j(1 - o_j))$$

$$\Delta w_{ji} = \eta(t_j - o_j)o_j(1 - o_j)o_i$$

Observations from Δw_{ji}

$$\Delta w_{ji} = \eta(t_j - o_j)o_j(1 - o_j)o_i$$

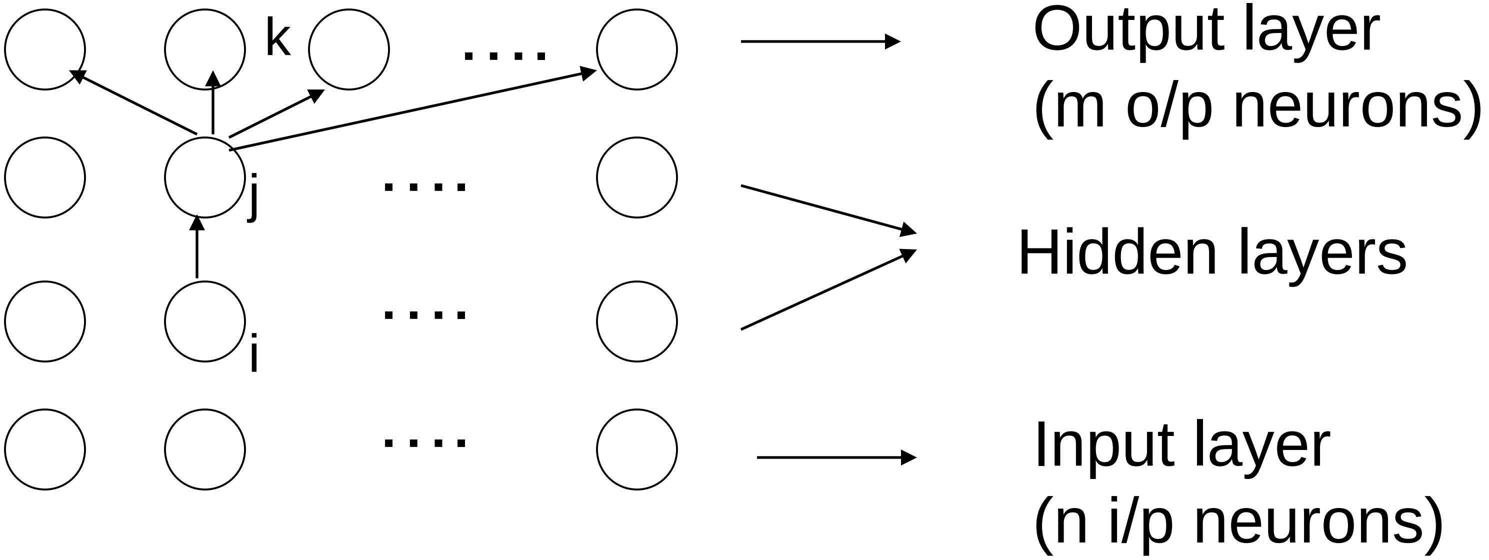
$$\Delta w_{ji} \to 0 \quad \text{if,}$$

$$1.\ O_j \to t_j \quad \text{and/or}$$

$$\left.\begin{array}{l} 2.\ O_j \to 1 \quad \text{and/or} \\ 3.\ O_j \to 0 \quad \text{and/or} \end{array}\right\} \text{Saturation behaviour}$$

$$4.\ O_i \to 0 \qquad \left.\vphantom{O_i}\right\} \text{Credit/Blame assignment}$$

Backpropagation for hidden layers



δ_k is propagated backwards to find value of δ_j

Backpropagation – for hidden layers

$$\Delta w_{ji} = \eta \delta j o_i$$

$$\delta j = -\frac{\delta E}{\delta net_j} = -\frac{\delta E}{\delta o_j} \times \frac{\delta o_j}{\delta net_j}$$

$$= -\frac{\delta E}{\delta o_j} \times o_j (1 - o_j)$$

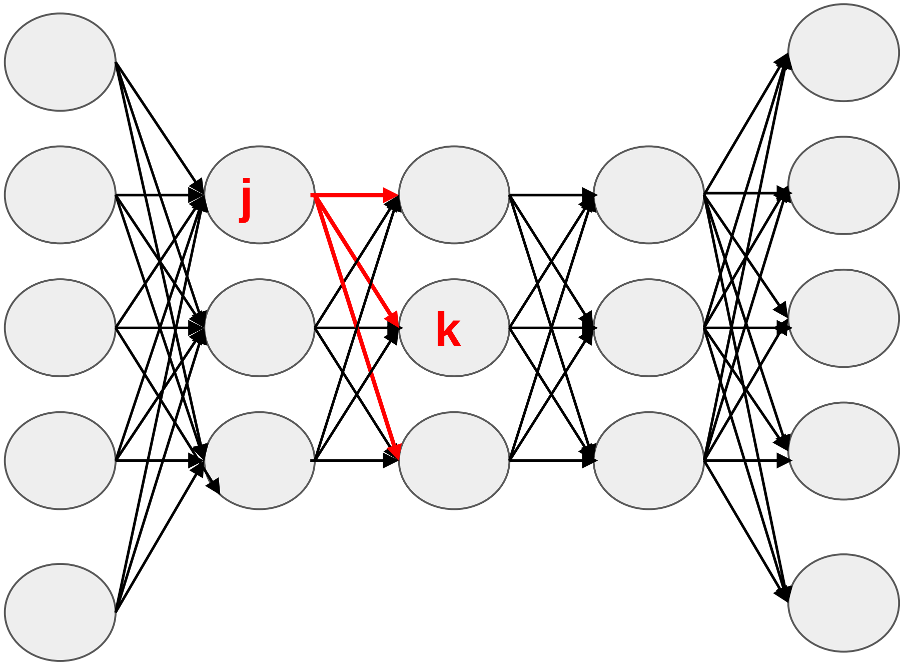
This recursion can give rise to vanishing and exploding Gradient problem

$$= -\sum_{k \in \text{next layer}} \left(\frac{\delta E}{\delta net_k} \times \frac{\delta net_k}{\delta o_j}\right) \times o_j (1 - o_j)$$

$$\text{Hence, } \delta_j = -\sum_{k \in \text{next layer}} (-\delta_k \times w_{kj}) \times o_j (1 - o_j)$$

$$= \sum_{k \in \text{next layer}} (w_{kj} \delta_k) o_j (1 - o_j)$$

Back-propagation- for hidden layers: Impact on net input on a neuron



- O_j affects the net input coming to all the neurons in next layer

General Backpropagation Rule

- General weight updating rule:

$$\Delta w_{ji} = \eta \delta j o_i$$

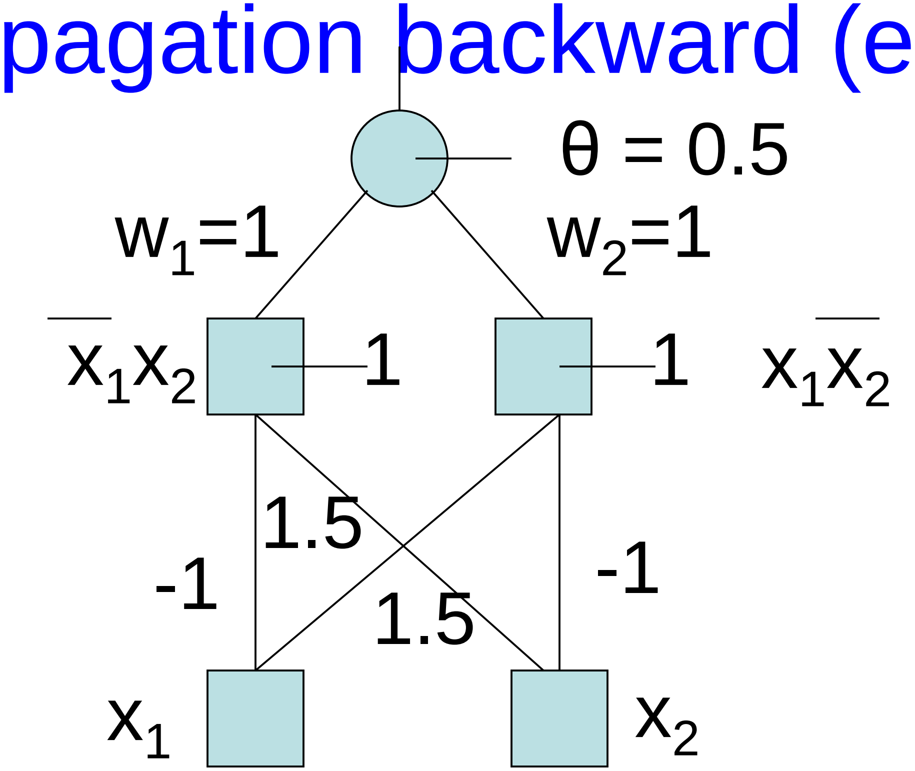
- Where

$$\delta_j = (t_j - o_j) o_j (1 - o_j) \quad \text{for outermost layer}$$

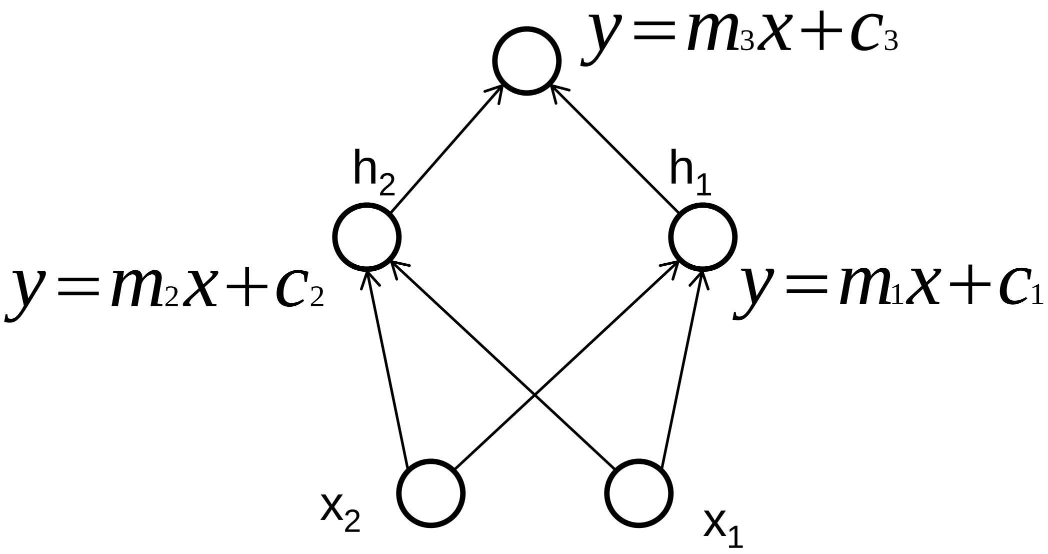
$$= \sum_{k \in \text{next layer}} (w_{kj} \delta_k) o_j (1 - o_j) o_i \quad \text{for hidden layers}$$

How does it work?

- Input propagation forward and error propagation backward (e.g. XOR)



Can Linear Neurons Work?



$$h_1=m_1(w_1x_1+w_2x_2)+c_1$$

$$h_1=m_1(w_1x_1+w_2x_2)+c_1$$

$$Out=(w_5h_1+w_6h_2)+c_3$$
$$=k_1x_1+k_2x_2+k_3$$

Note: The whole structure shown in earlier slide is reducible to a single neuron with given behavior

$$Out = k_1 x_1 + k_2 x_2 + k_3$$

Claim: A neuron with linear I-O behavior can’t compute X-OR.

Proof: Considering all possible cases:

[assuming 0.1 and 0.9 as the lower and upper thresholds]

For (0,0), Zero class:

$$m(w_1.0 + w_2.0 - \theta) + c < 0.1$$
$$\Rightarrow c - m.\theta < 0.1$$

For (0,1), One class:

$$m(w_1.1 + w_2.0 - \theta) + c > 0.9$$
$$\Rightarrow m.w_1 - m.\theta + c > 0.9$$

For (1,0), One class: $m.w_2 - m.\theta + c > 0.9$

For (1,1), Zero class: $m.w_1 - m_2.\theta + c < 0.1$

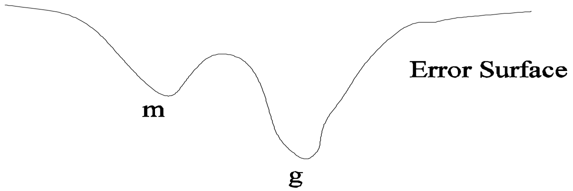
These equations are inconsistent. Hence X-OR can’t be computed.

Observations:

1. A linear neuron can’t compute X-OR.
2. A multilayer FFN with linear neurons is collapsible to a single linear neuron, hence **no a additional power due to hidden layer.**
3. Non-linearity is essential for power.

Local Minima

Due to the Greedy nature of BP, it can get stuck in local minimum *m* and will never be able to reach the global minimum *g* as the error can only decrease by weight change.



m- local minima, g- global minima

Figure- Getting Stuck in local minimum

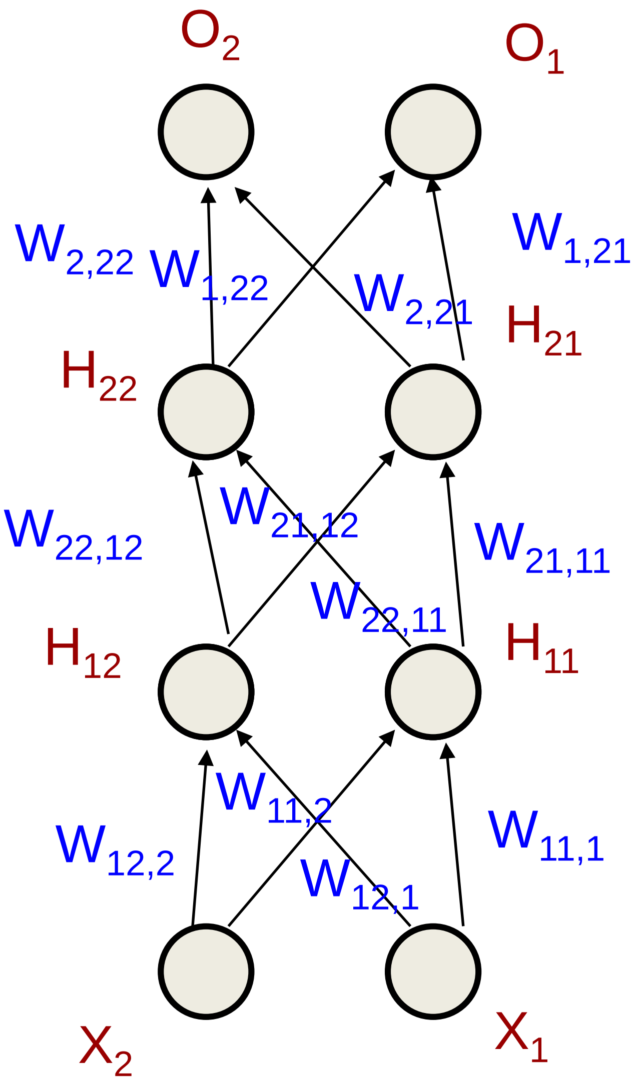
Momentum factor

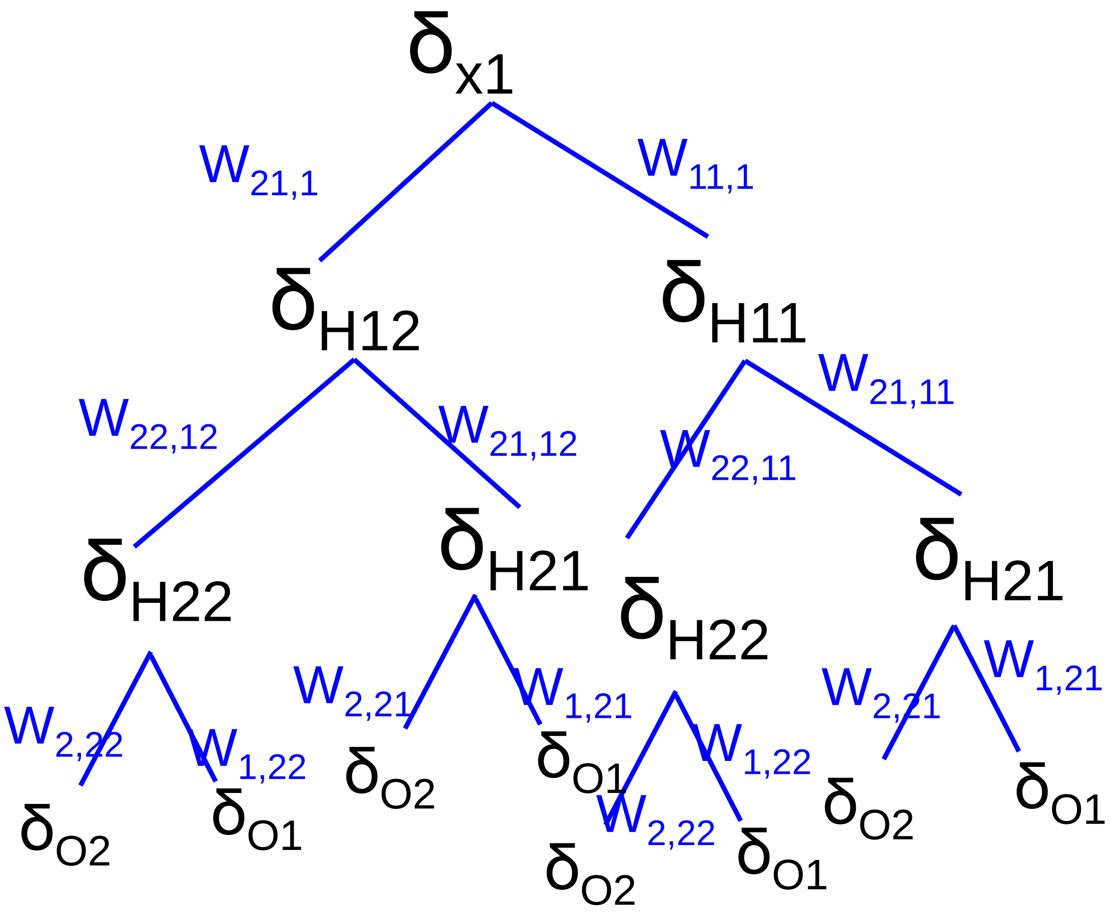
1. Introduce momentum factor.

$$(\Delta w_{ji})_{nth-iteration} = \eta\delta_j O_i + \beta(\Delta w_{ji})_{(n-1)th-iteration}$$

- Accelerates the movement out of the trough.
- Dampens oscillation inside the trough.
- Choosing β : If β is large, we may jump over the minimum.

Vanishing/Exploding Gradient





$$\delta_{x1}=W_{11,1}\delta_{H11}+W_{21,1}\delta_{H12}$$

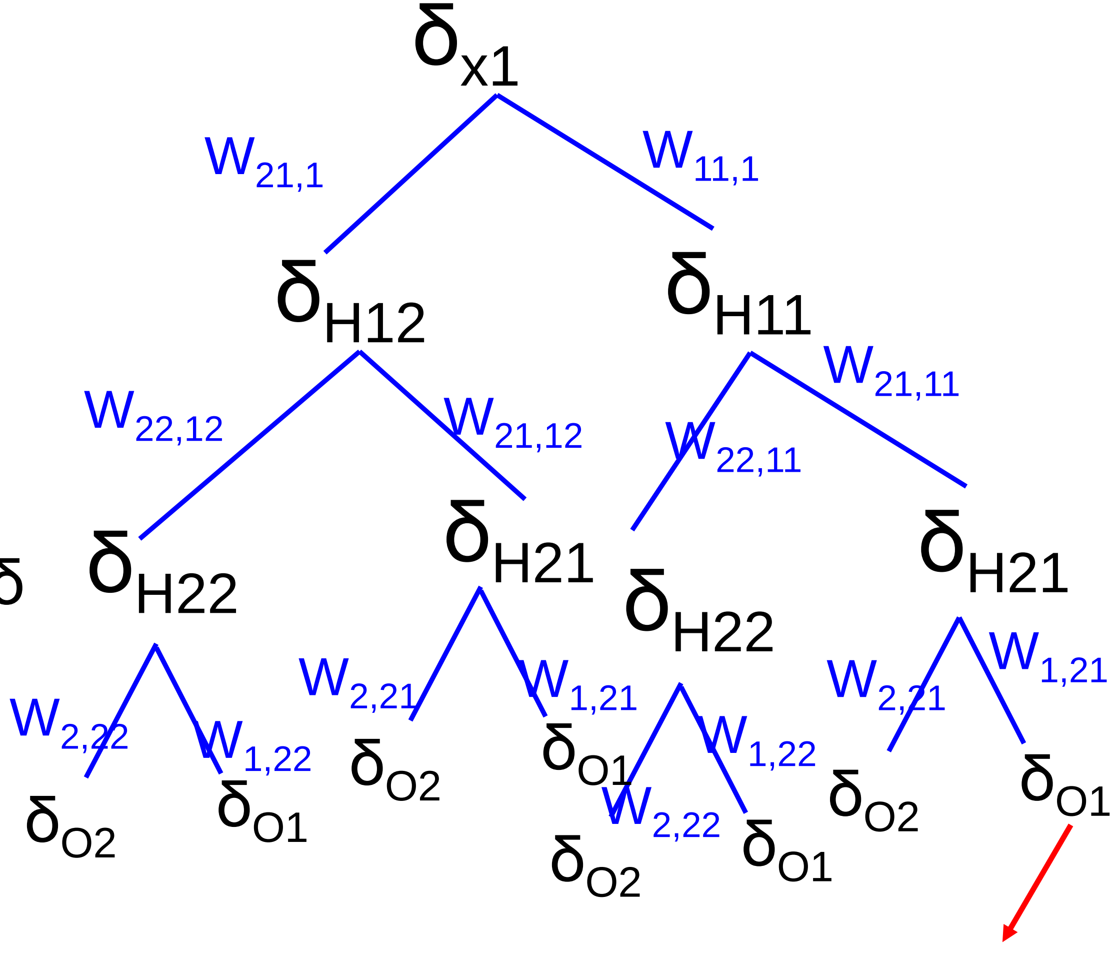
Vanishing/Exploding Gradient

$\delta_{x1}=W_{11,1}\delta_{H11}+W_{21,1}\delta_{H12}$ [2 terms]

$=W_{11,1}(W_{21,11}\delta_{H21}+W_{22,11}\delta_{H22})+W_{21,1}(W_{21,12}\delta_{H21}+W_{22,12}\delta_{H22})$ [4 terms]

$= W_{11,1}W_{21,11}\delta_{H21}+W_{11,1}W_{22,21}\delta_{H22}+W_{21,1}W_{21,12}\delta_{H21}+W_{21,1}W_{22,12}\delta_{H22}$

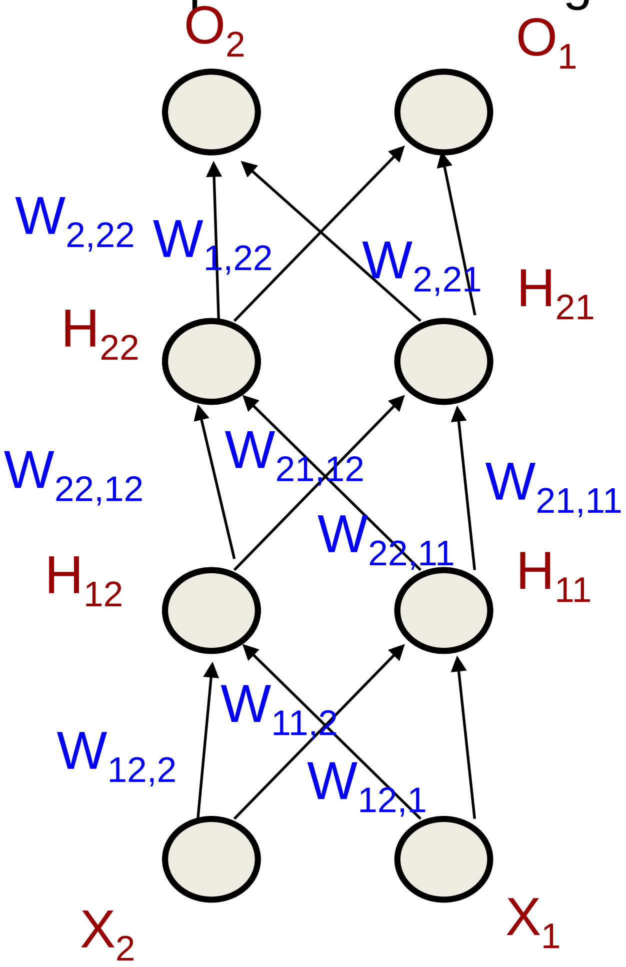
= (4 terms with δ_{o1}) + (4 terms with δ_{o2}; one term shown for the leftmost leaf's weight)

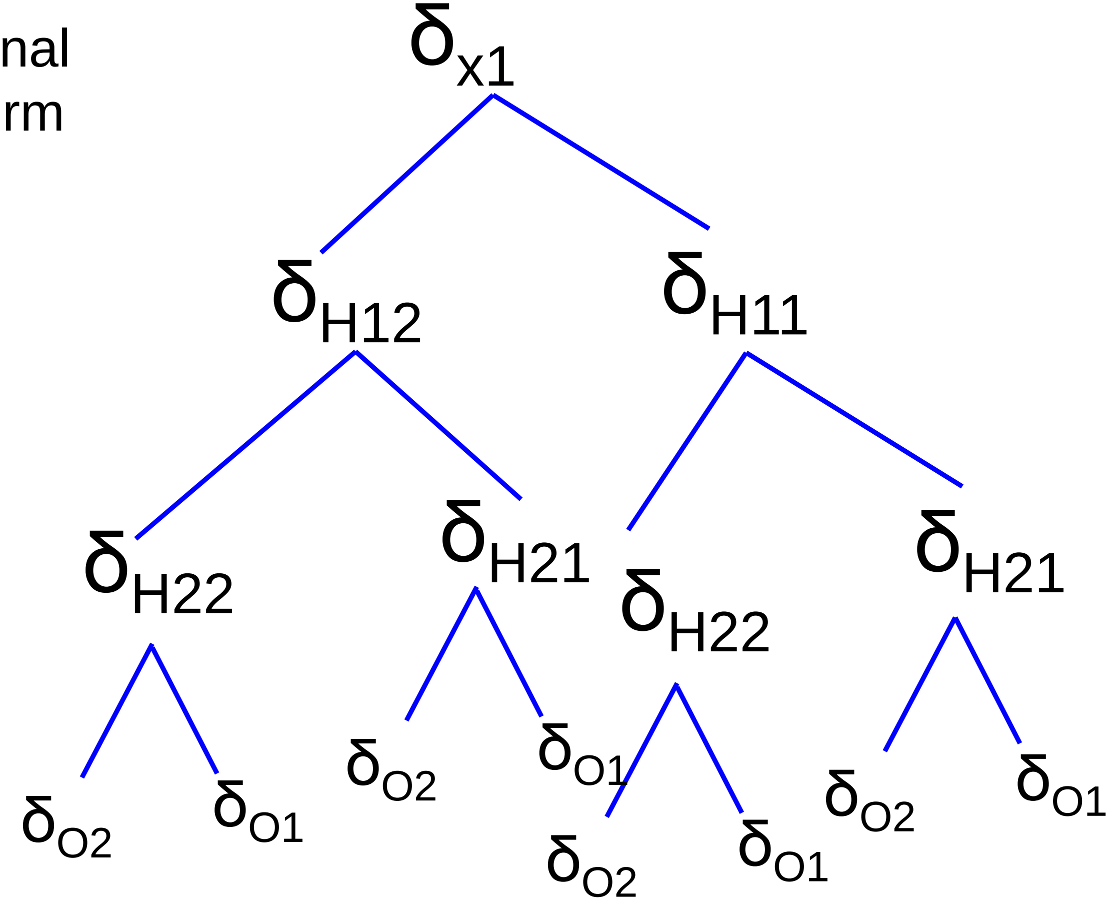


$W_{11,1}W_{21,11}W_{1,21}$

Vanishing/Exploding Gradient

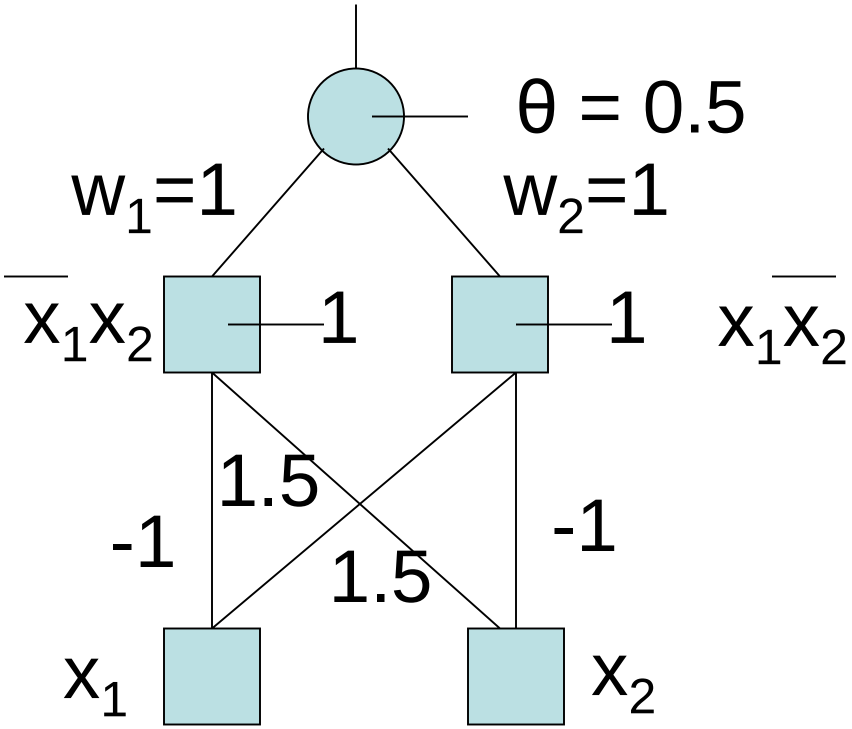
With ‘*B*’ as branching factor and ‘*L*’ as number of levels, There will be B^L terms in the final Expansion of δ_{x1}. Also each term Will be product of L weights





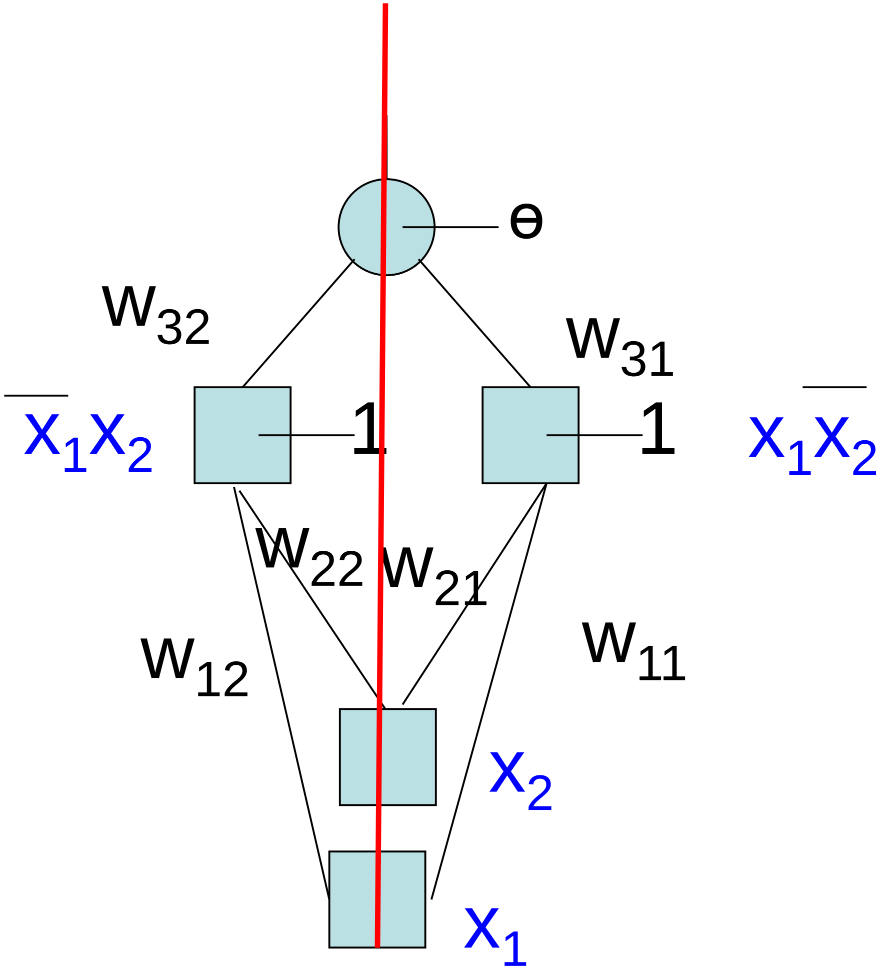
Symmetry breaking

- If mapping demands different weights, but we start with the same weights everywhere, then BP will never converge.



XOR n/w: if we s started with identical weight everywhere, BP will not converge

Symmetry breaking: understanding with proper diagram



Symmetry
About
The red
Line should
Be broken

Exercise

- Find the weakest condition for symmetry breaking. It is not the case that only when ALL weights are equal, the network faces the symmetry problem.

Backpropagation Application in NLP

Depenency Parsing

Dependency parsing

- Process
  - Determine all parts of speech tags (Identify verbs, nouns etc.)
  - If there are multiple verbs, identify main verb
  - Root arrow: Main verb
  - Recursively identify heads and modifiers in the subparts of clauses and phrases of the sentence
- Transition based dependency parsing
  - Find heads and modifiers from a sentence as a classification problem
    - At every position identify if it is a head or a modifier
    - If modifier, which head it is associated with

Exercise: Dependency parsing using FFNN

- Implement feed forward neural network for dependency parsing.
    - Each step in dependency parsing is a classification problem.
        - First classification decision is with respect to finding the main verb.
        - Then at every step we decide if a word is a head or a modifier; if modifier, then modifier for which word.

Recurrent Neural Network

Acknowledgement:

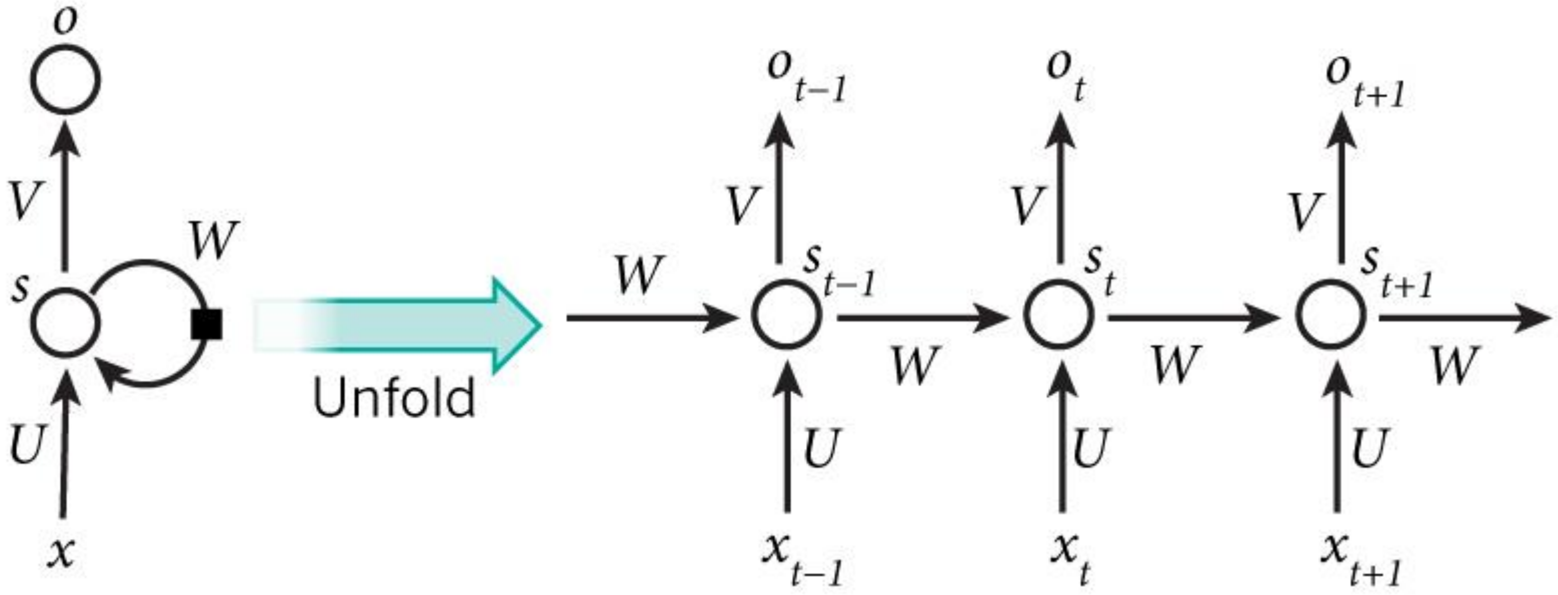
1. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

By Denny Britz

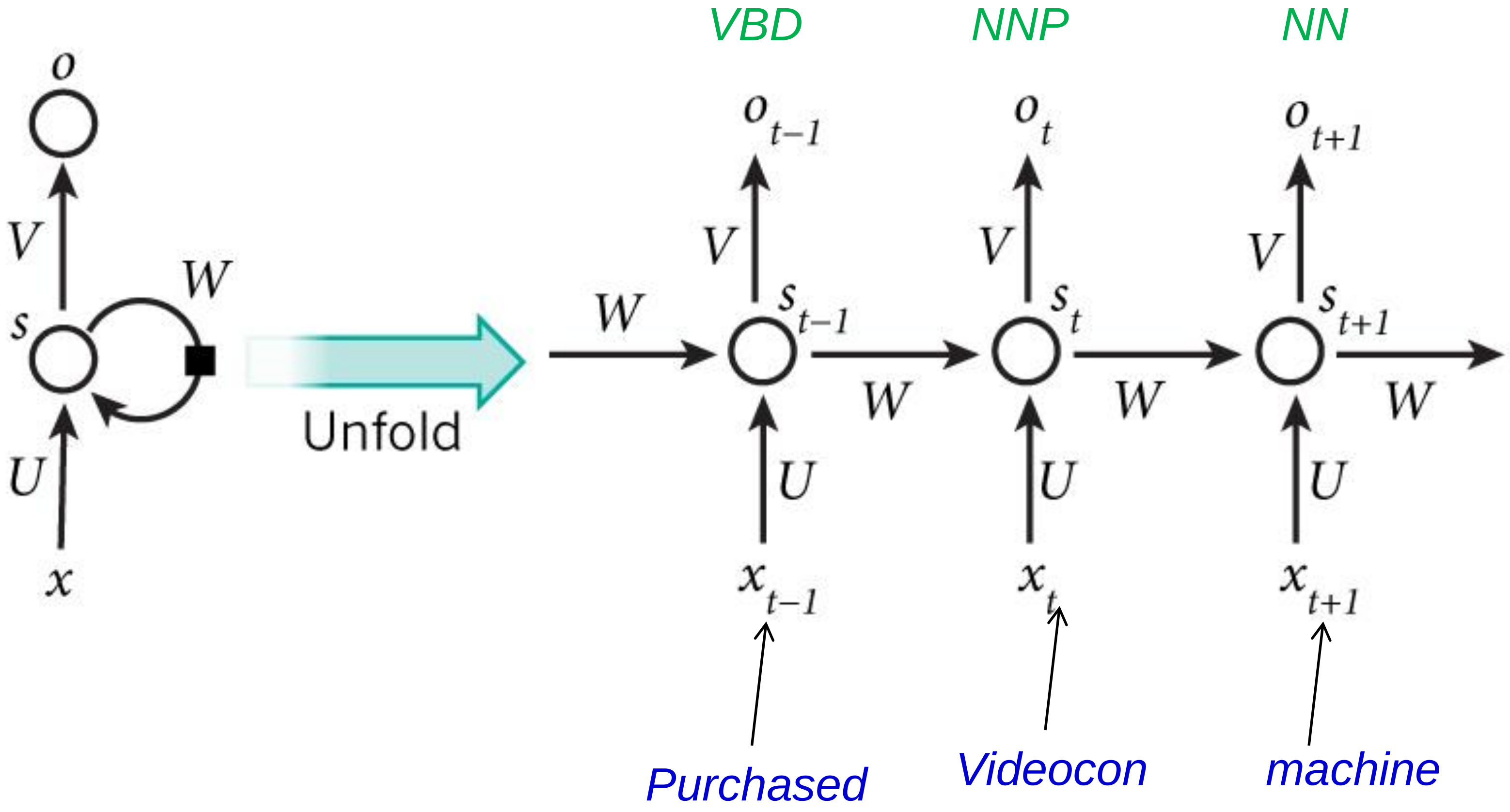
2. Introduction to RNN by Jeffrey Hinton

http://www.cs.toronto.edu/~hinton/csc2535/lectures.html

Sequence processing m/c



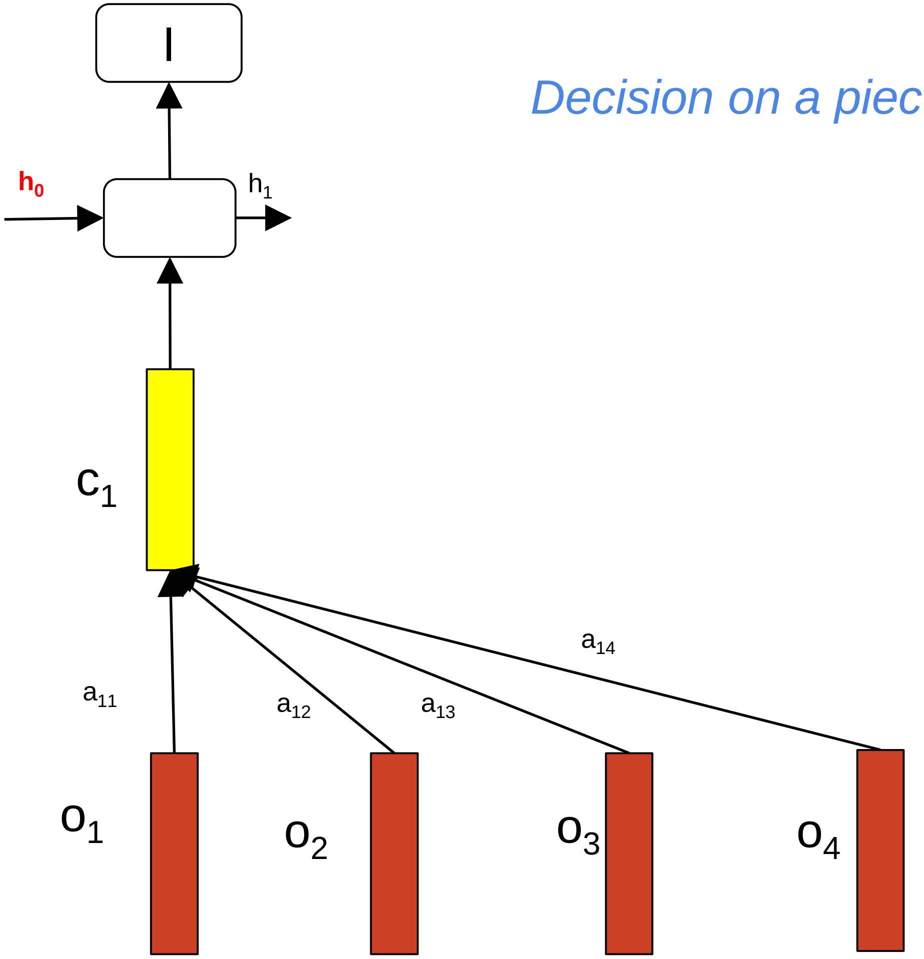
E.g. POS Tagging



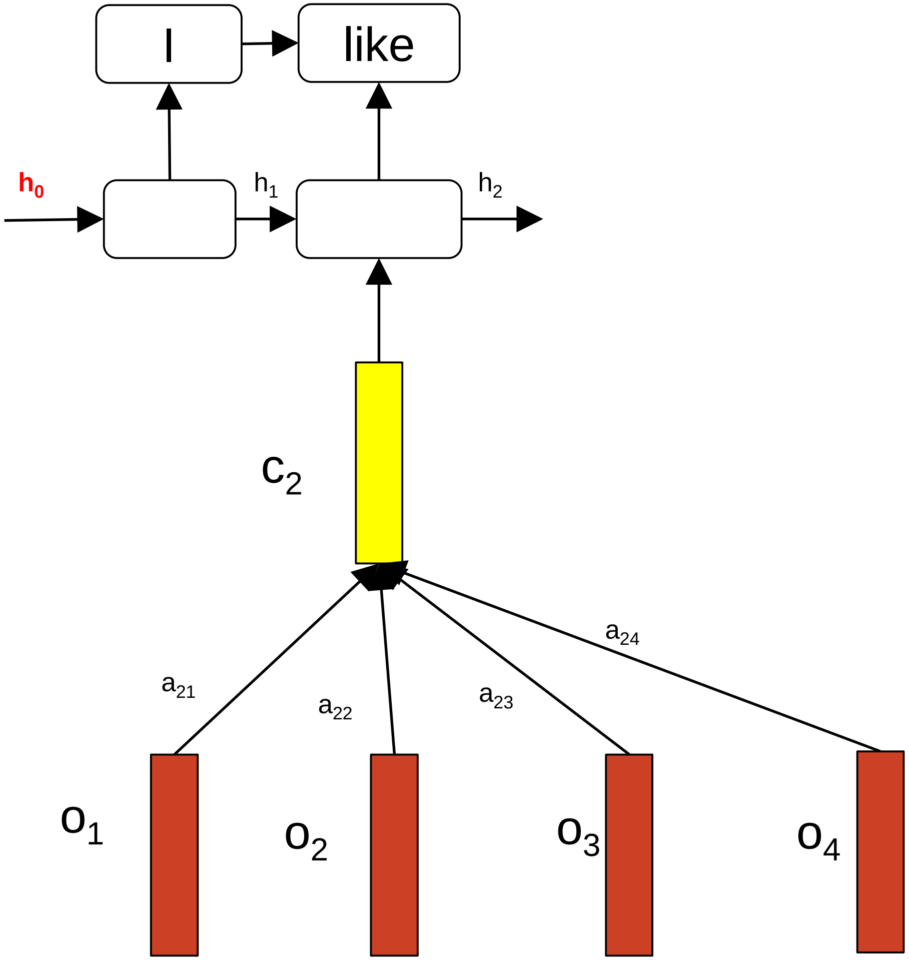
VBD *NNP* *NN*

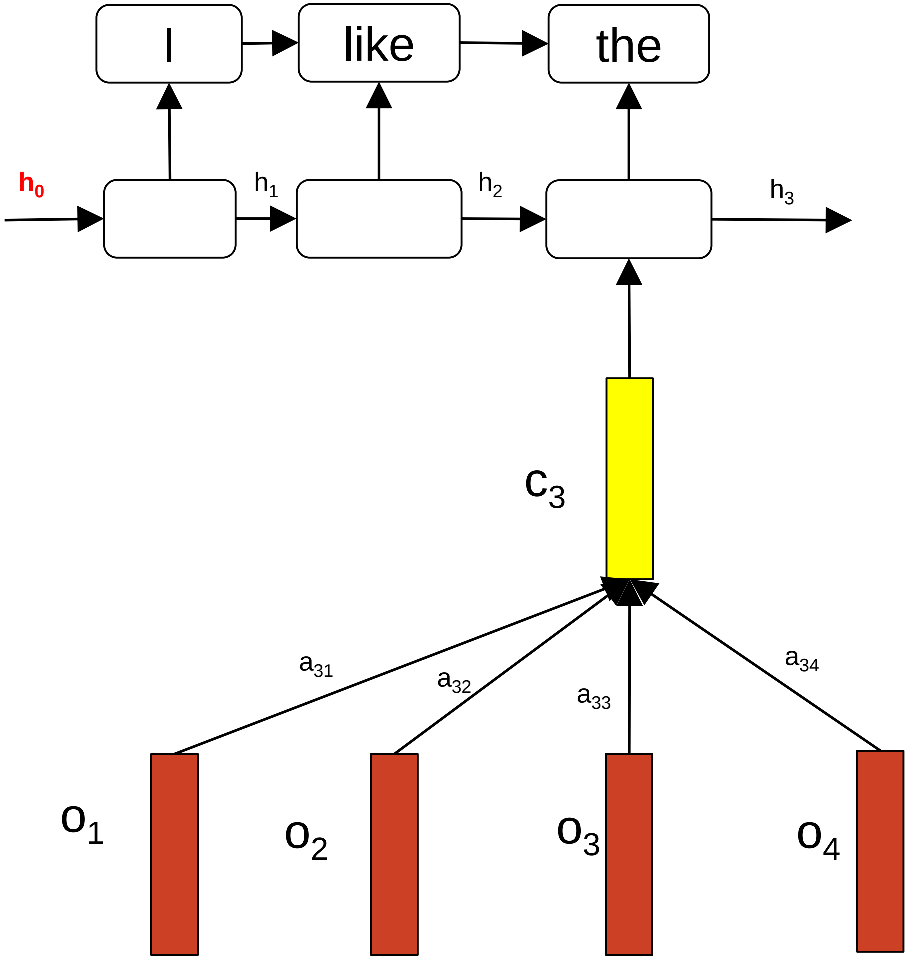
Purchased *Videocon* *machine*

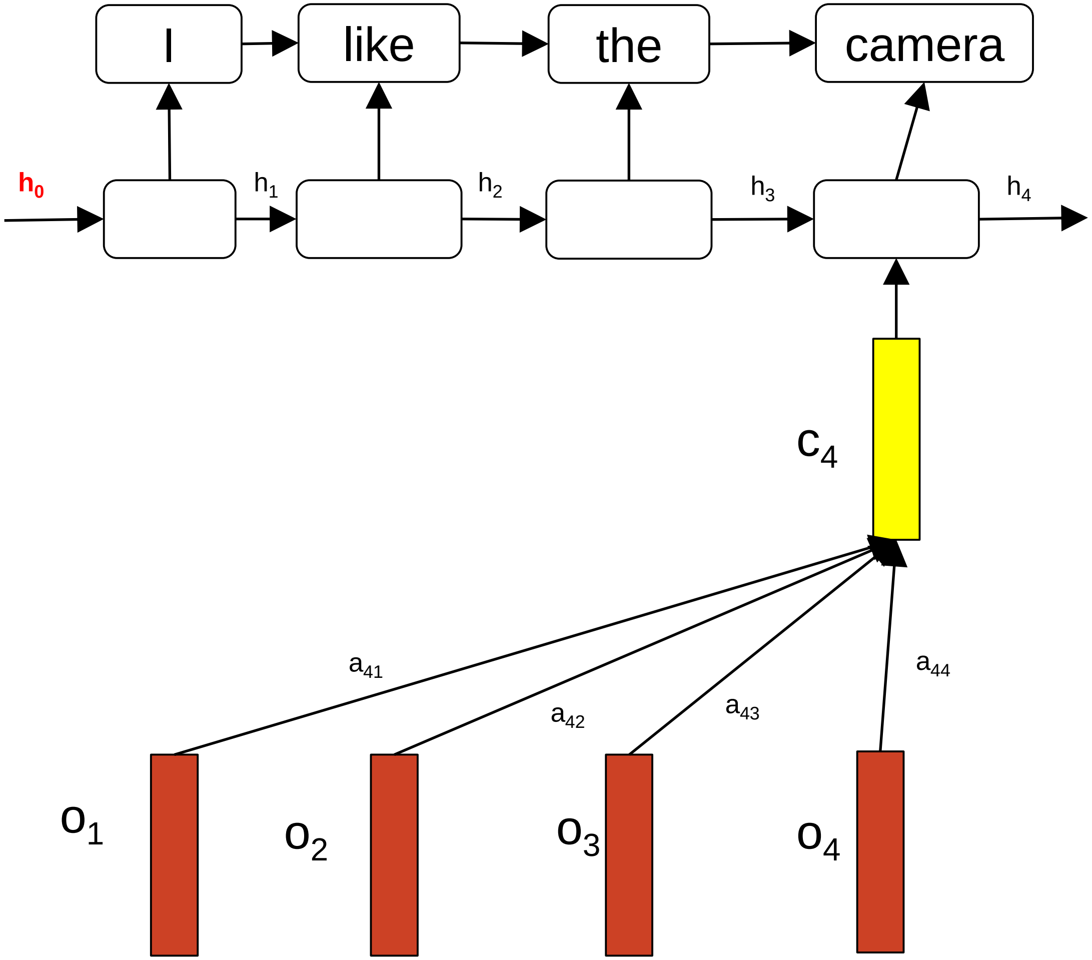
E.g. Sentiment Analysis

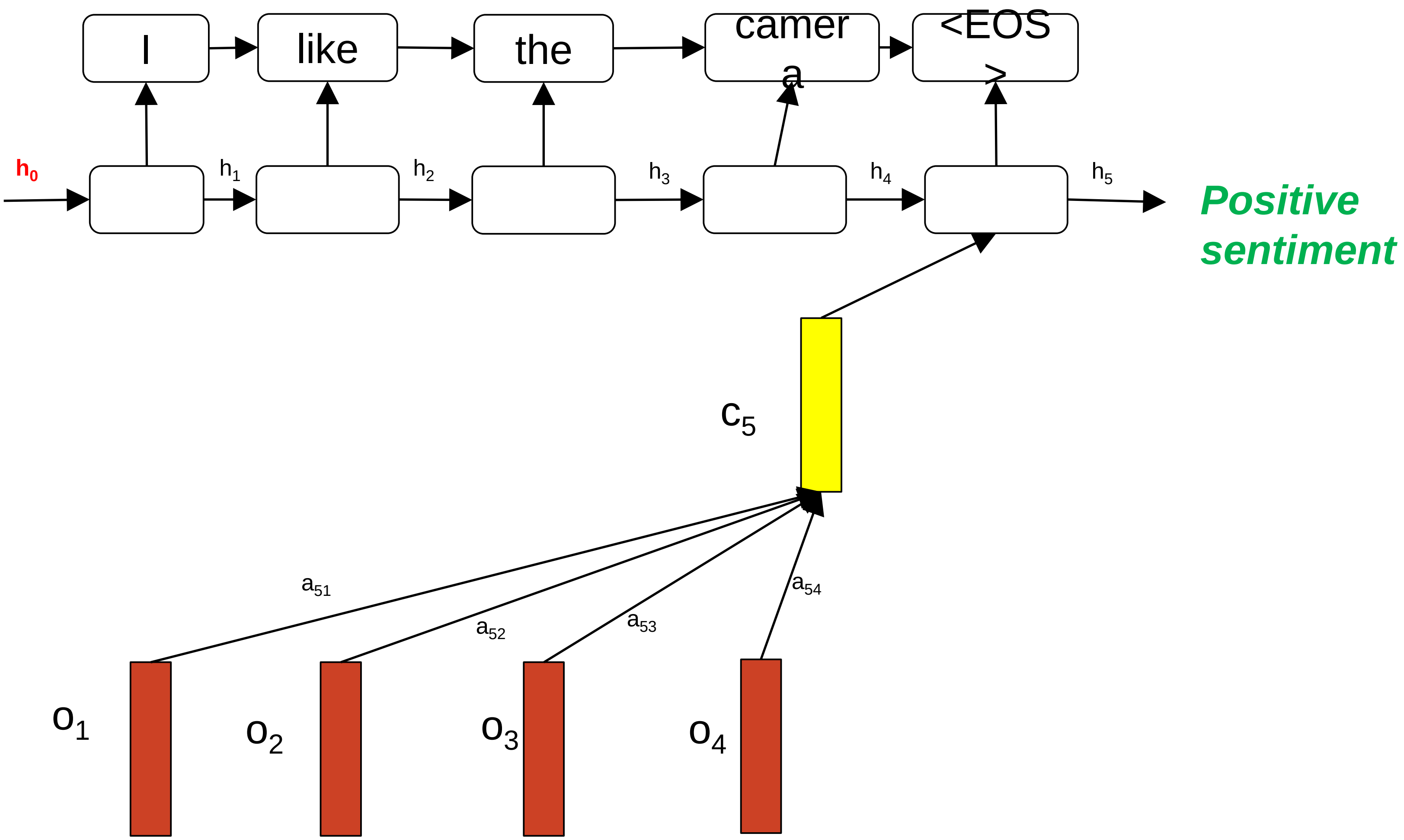


Decision on a piece of text

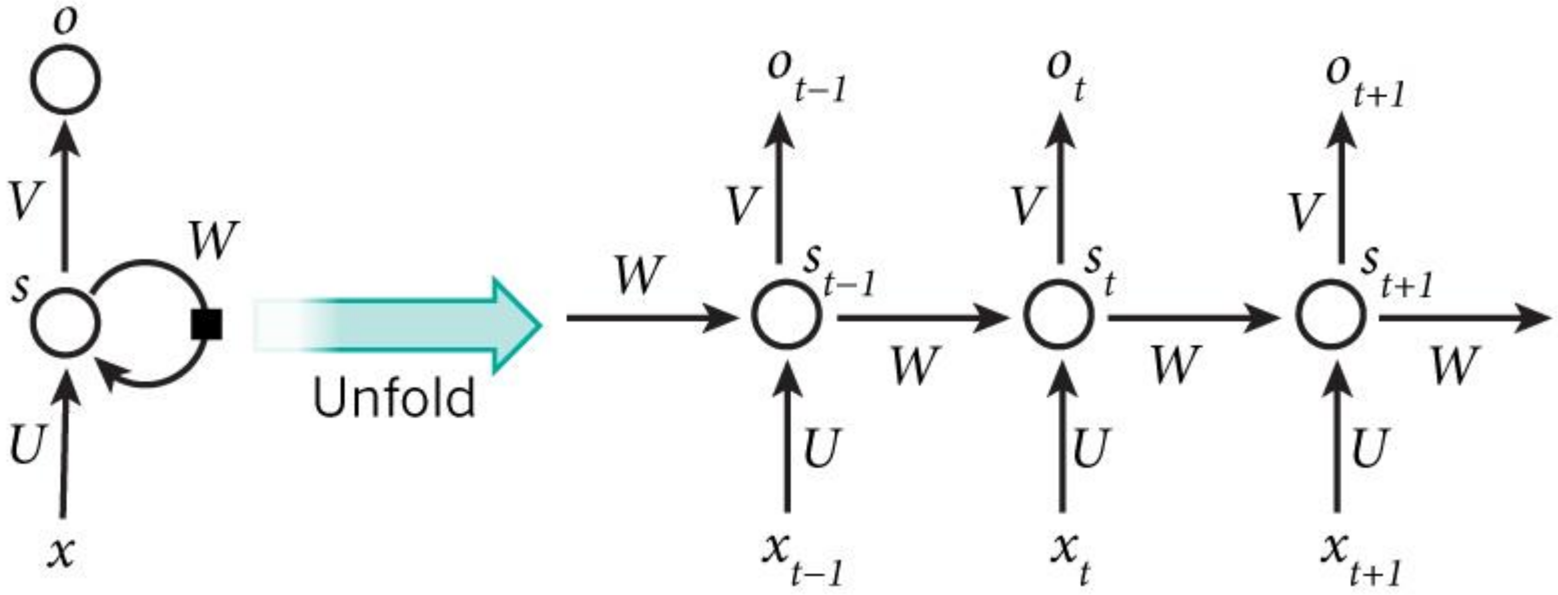

I
like
h_0
h_1
h_2
c_2
a_{24}
a_{21}
a_{22}
a_{23}
o_1
o_2
o_3
o_4


I
like
the
h0
h1
h2
h3
c3
a31
a32
a33
a34
o1
o2
o3
o4


I
like
the
camera
h_0
h_1
h_2
h_3
h_4
c_4
a_{41}
a_{42}
a_{43}
a_{44}
o_1
o_2
o_3
o_4


I
like
the
camer a
<EOS >
h0
h1
h2
h3
h4
h5
Positive sentiment
c5
a51
a52
a53
a54
o1
o2
o3
o4

Back to RNN model



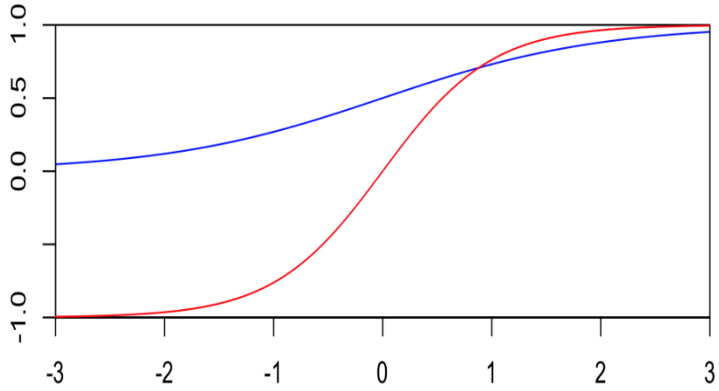
Notation: input and state

- x_t is the input at time step *t*. For example, could be a one-hot vector corresponding to the second word of a sentence.
- s_t is the hidden state at time step *t*. It is the "memory" of the network.
- $s_t = f(U.x_t + Ws_{t-1})$ ***U*** **and** ***W*** **matrices are learnt**
- *f* is a function of the input and the previous state
- Usually *tanh* or *ReLU* (approximated by *softplus)*

Tanh, *ReLU* (rectifier linear unit) and Softplus

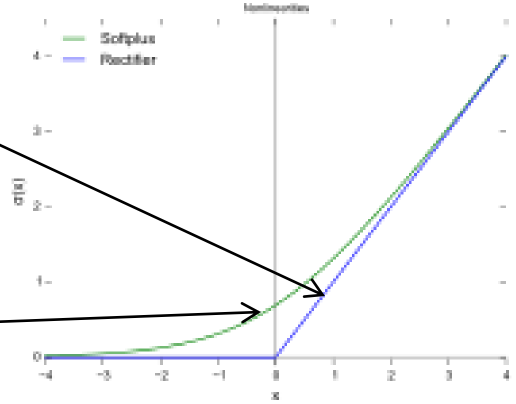
$$\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$\tanh =$



$$f(x) = \max(0, x)$$

$$g(x) = \ln(1 + e^{x})$$



Notation: output

- o_t is the output at step t

- For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary

- $o_t = softmax(V.s_t)$

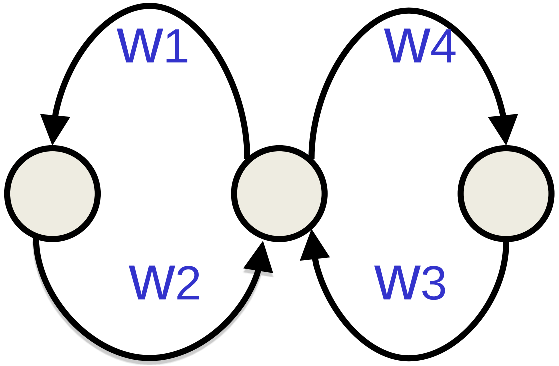
Operation of RNN

- RNN shares the same parameters (*U*, *V*, *W*) across all steps

- Only the input changes

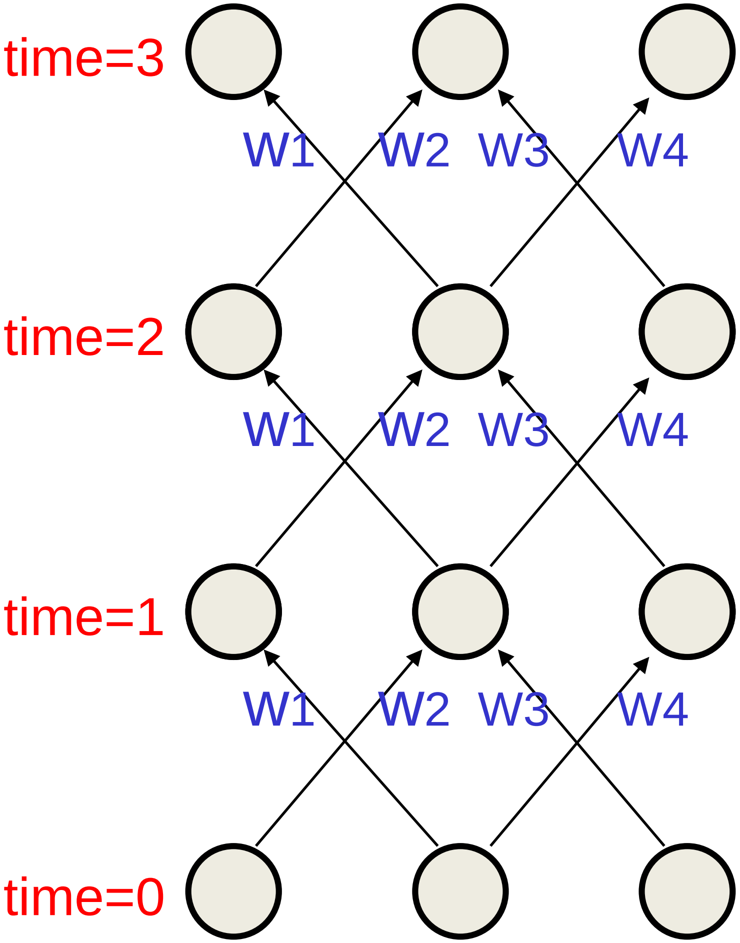
- Sometimes the output at each time step is not needed: e.g., in sentiment analysis
- Main point: the hidden states !!

The equivalence between feedforward nets and recurrent nets



Assume that there is a time delay of 1 in using each connection.

The recurrent net is just a layered net that keeps reusing the same weights.


time=3
W1
W2
W3
W4
time=2
W1
W2
W3
W4
time=1
W1
W2
W3
W4
time=0