CS626: Speech, NLP and the Web

EM, SMT alignment, BP

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

Week of 1st November, 2021

Derivation of E and M steps for 2 coin problem (1/2)- M step

Take partial derivative of $E_{Z|X,\theta}(.)$ (prev. slide) wrt p, p_1, p_2 and equate to 0.

$$p = \frac{\sum_{i=1}^{N} E(z_i)}{N}$$

$$p_1 = \frac{\sum_{i=1}^{N} E(z_i)x_i}{\sum_{i=1}^{N} E(z_i)}$$

(Uses the fact that E(CX) =C.E(X), where C is a constant)

$$p_2 = \frac{M - \sum_{i=1}^{N} E(z_i)x_i}{N - \sum_{i=1}^{N} E(z_i)}; M = \#Heads, N = \#tosses$$

(Uses the fact that E(C+X) =C+E(X), where C is a constant)

Derivation of E and M steps for 2 coin problem (2/2)- E step

$E(z_i|x_i) = 1.P(z_i=1|x_i) + 0.P(z_i=0|x_i)$

$= P(z_i=1|x_i)$

$$P(z_i = 1 | x_i) = \frac{P(z_i = 1, x_i)}{P(x_i)}$$

$$= \frac{pp_1^{x_i}(1-p_1)^{1-x_i}}{P(x_i, z_i = 1) + P(x_i, z_i = 0)}$$

$$= \frac{pp_1^{x_i}(1-p_1)^{1-x_i}}{pp_1^{x_i}(1-p_1)^{1-x_i} + (1-p)p_2^{x_i}(1-p_2)^{1-x_i}}$$

Generalization into N “throws” using M “things” each having L outcomes

From

Pushpak Bhattacharyya, *Machine Translation*, CRC Press, 2015

Multiple outcomes from multiple entities

- “Throw” of “something” where that something has more than 2 outcomes, e.g., throw of multiple dice
- The observation sequence has a sequence of 1 to 6s
- But we do not know which observation came from which dice
- Gives rise to a multinomial that is extremely useful in NLP ML.

Observation Sequence

- N ‘throws’, 1 of L outcomes from each throw, 1 of the M ‘things’ (called ‘sources’) chosen
- $\Sigma_{k=1,L} x_{ik}=1$, since each x_{ik} is either 1 or 0 and one and only one of them is1.
- D (data):

$$<x_{11}/x_{12}/\ldots x_{1L}>, <x_{21}/x_{22}/\ldots x_{2L}>, \ldots$$
$$<x_{N1}/x_{N2}/\ldots x_{NL}>$$

Hidden Variable

- Hidden variable for M sources
- $\Sigma_{j=1,M} z_{ij} = 1$, since each z_{ij} is either 1 or 0 and one and only one of them is 1.
- Z:

$$<z_{11}/z_{12}/\ldots z_{1M}>, <z_{21}/z_{22}/\ldots z_{2M}>, \ldots$$
$$<z_{N1}/z_{N2}/\ldots z_{NM}>$$

Parameters

- Parameter set θ:
  - π_j: probability of choosing source j
  - p_{jk}: probability of observing k^{th} outcome from the j^{th} source

Likelihood Expression and Constraints

multinouli expression

$$L = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\pi_j \prod_{k=1}^{L} p_{jk}^{x_{ik}} \right]^{z_{ij}}$$

$$LL = \sum_{i=1}^{N} \sum_{j=1}^{M} z_{ij} \left[\log \pi_j + \sum_{k=1}^{L} x_{ik} \log p_{jk} \right]$$

$$E_{Z|X}(LL) = \sum_{i=1}^{N} \sum_{j=1}^{M} E(z_{ij} \mid x_{ik}) \left[\log \pi_j + \sum_{k=1}^{L} x_{ik} \log p_{jk} \right]$$

Maximize subject to

$$\sum_{j=1}^{M} \pi_j = 1$$

$$\sum_{k=1}^{L} p_{jk} = 1, \; j = 1, \ldots M$$

Introduce Lagrangian

$$Q =$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M} E(z_{ij} \mid x_{ik})\left[\log \pi_j + \sum_{k=1}^{L} x_{ik} \log p_{jk}\right]$$

$$-\alpha\left(\sum_{j=1}^{M} \pi_j - 1\right) - \sum_{j=1}^{M} \beta_j \left(\sum_{k=1}^{J} p_{jk} - 1\right)$$

Finding ΠTs

$$\frac{\partial Q}{\partial \pi_j} = \sum_{i=1}^{N} \frac{E(z_{ij})}{\pi_j} - \alpha = 0$$

$$\Rightarrow \alpha.\pi_j = \sum_{i=1}^{N} E(z_{ij}); \Rightarrow \sum_{j=1}^{M} (\alpha.\pi_j) = \sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij})$$

$$\Rightarrow \alpha = \sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij}), \because \sum_{j=1}^{M} \pi_j = 1$$

$$\Rightarrow \pi_j = \frac{\sum_{i=1}^{N} E(z_{ij})}{\sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij})} = \frac{\sum_{i=1}^{N} E(z_{ij})}{\sum_{i=1}^{N} \sum_{j=1}^{M} E(z_{ij})}$$

$$now, \sum_{j=1}^{M} E(z_{ij}) = \sum_{j=1}^{M} [1.P(z_{ij} = 1) + 0.P(z_{ij} = 0)]$$

$$= \sum_{j=1}^{M} \pi_j = 1$$

$$\Rightarrow \pi_j = \frac{\sum_{i=1}^{N} E(z_{ij})}{\sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij})} = \frac{\sum_{i=1}^{N} E(z_{ij})}{N}$$

Finding P_{ij}s

$$\frac{\partial Q}{\partial p_{jk}} = \sum_{i=1}^{N} \frac{E(z_{ij})}{p_{jk}} - \beta_j = 0$$

$$\Rightarrow \beta_j . p_{jk} = \sum_{i=1}^{N} E(z_{ij}).x_{ik}; \Rightarrow \sum_{k=1}^{L} (\beta_j . p_{jk}) = \sum_{k=1}^{L} \sum_{i=1}^{N} E(z_{ij}).x_{ik}$$

$$\Rightarrow \beta_j = \sum_{k=1}^{L} \sum_{i=1}^{N} E(z_{ij}) x_{ik}, \because \sum_{k=1}^{L} p_{jk} = 1$$

$$\Rightarrow p_{jk} = \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{k=1}^{L} \sum_{i=1}^{N} E(z_{ij}) x_{ik}} = \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{i=1}^{N} E(z_{ij}) . \sum_{k=1}^{L} x_{ik}}$$

$$Now, \sum_{k=1}^{L} x_{ik} = 1$$

$$\Rightarrow p_{jk} = \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{i=1}^{N} E(z_{ij})}$$

Expectation of hidden variable: E step

$E(z_{ij}|x_{ik}) = 1.P(z_{ij}=1|x_{ik}) + 0.P(z_{ij}=0|x_{ik})$

$= P(z_{ij}=1|x_{ik})$

$$P(z_{ij}=1|x_{ik}) = \frac{P(z_{ij}=1, x_{ik})}{P(x_{ik})}$$

$$= \frac{\pi_j \prod_{k=1}^{L} p_{jk}^{x_{ik}}}{\sum_{j=1}^{M} P(x_{ik}, z_{ij})}$$

$$= \frac{\pi_j \prod_{k=1}^{L} p_{jk}^{x_{ik}}}{\sum_{j=1}^{M} [\pi_j \prod_{k=1}^{L} p_{jk}^{x_{ik}}]}$$

M-step

M-Step:

$$\pi_j = \frac{\sum_{i=1}^{N} E(z_{ij})}{\sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij})}$$

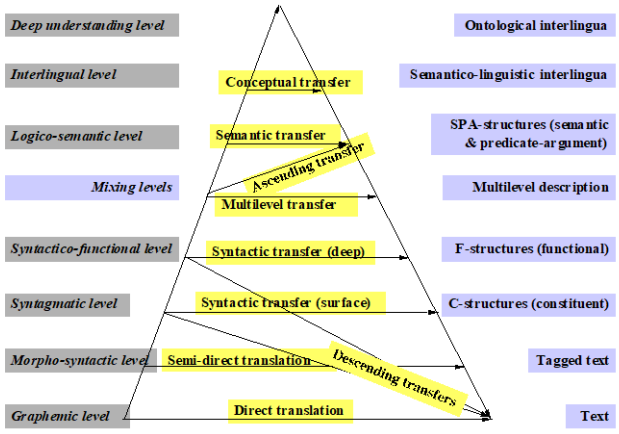
$$p_{jk} = \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{i=1}^{N} E(z_{ij})}$$

E-step

E-Step:

$$E(z_{ij}) = \frac{\pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})}{\sum_{j=1}^{M} \pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})}$$

Statistical Machine Translation



Czeck-English data

- [nesu] “I carry”
- [ponese] “He will carry”
- [nese] “He carries”
- [nesou] “They carry”
- [yedu] “I drive”
- [plavou] “They swim”

To translate …

- I will carry.
- They drive.
- He swims.
- They will drive.

Hindi-English data

- [DhotA huM] “I carry”
- [DhoegA] “He will carry”
- [DhotA hAi] “He carries”
- [Dhote hAi] “They carry”
- [chalAtA huM] “I drive”
- [tErte hEM] “They swim”

Bangla-English data

- [bai] “I carry”
- [baibe] “He will carry”
- [bay] “He carries”
- [bay] “They carry”
- [chAlAi] “I drive”
- [sAMtrAy] “They swim”

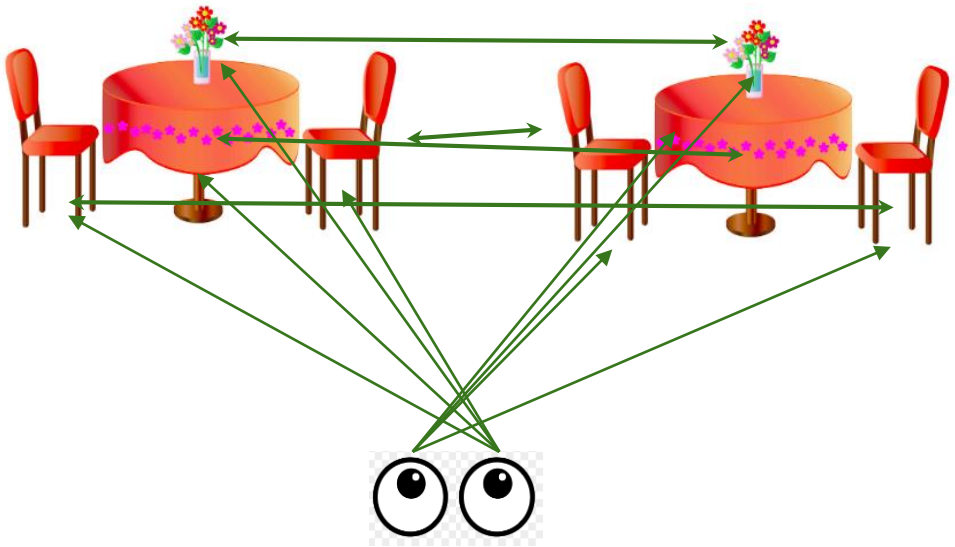
To translate … (repeated)

- I will carry.
- They drive.
- He swims.
- They will drive.

Alignment

How to build part alignment from whole alignment

- Two images are in alignment: images on the two retina
- Need to find alignment of parts of it



Fundamental and ubiquitous

- Spell checking
- Translation
- Transliteration
- Speech to text
- Text to speeh

EM for word alignment from sentence alignment: example

English	French
(1) three rabbits	(1) trois lapins
a b	w x
(2) rabbits of Grenoble	(2) lapins de Grenoble
b c d	x y z

Exercise: Think of how the knowledge that three/trois is a number noun, Grenoble is a place, rabbit/lapins is a noun of animacy can help reduce data need and also make computation more efficient.

VIMP: The parameters and hidden variables in the rabbit-lapins example

- We are interested in probabilities of English words to French word mappings in E-F lexicon; these are the PARAMETERS

- The HIDDEN variables are the indicator variables capturing alignments in the parallel sentences; expected values of these indicators variables give the EXPECTED COUNT of word-word alignments per sentence pair

Initial Probabilities:
each cell denotes *t(a ←→ w), t(a ←→ x) etc.*

	a	b	c	d
w	1/4	1/4	1/4	1/4
x	1/4	1/4	1/4	1/4
y	1/4	1/4	1/4	1/4
z	1/4	1/4	1/4	1/4

Example of expected count

C[w←→a; (a b)←→(w x)] {this is the expected count, corresponding to the expected value of the indicator variable which is 1 if w and a are indeed aligned, else 0}

$$= \frac{t(w \leftrightarrow a)}{t(w \leftrightarrow a)+t(w \leftrightarrow b)} \times \#(a \text{ in } \text{`a b'}) \times \#(w \text{ in } \text{`w x'})$$

$$= \frac{1/4}{1/4+1/4} \times 1 \times 1 = 1/2$$

“counts”

a b ←→ ***w x***	a	b	c	d
w	1/2	1/2	0	0
x	1/2	1/2	0	0
y	0	0	0	0
z	0	0	0	0

b c d ←→ ***x y z***	a	b	c	d
w	0	0	0	0
x	0	1/3	1/3	1/3
y	0	1/3	1/3	1/3
z	0	1/3	1/3	1/3

Revised probability: example

$$t_{revised}(a \leftrightarrow w)$$

$$= \frac{1/2}{(1/2+1/2+0+0)_{(a\ b)\leftrightarrow(w\ x)} + (0+0+0+0)_{(b\ c\ d)\leftrightarrow(x\ y\ z)}}$$

Revised probabilities table

	a	b	c	d
w	1/2	1/2	0	0
x	1/4	5/12	1/6	1/6
y	0	1/3	1/3	1/3
z	0	1/3	1/3	1/3

“revised counts”

a b ←→ ***w x***	a	b	c	d
w	1/2	3/8	0	0
x	1/2	5/8	0	0
y	0	0	0	0
z	0	0	0	0

b c d ←→ ***x y z***	a	b	c	d
w	0	0	0	0
x	0	5/9	1/3	1/3
y	0	2/9	1/3	1/3
z	0	2/9	1/3	1/3

Re-Revised probabilities table

	a	b	c	d
w	1/2	1/2	0	0
x	3/16	**85/144**	1/9	1/9
y	0	1/3	1/3	1/3
z	0	1/3	1/3	1/3

Continue until convergence; notice that (b,x) binding gets progressively stronger; b=rabbits, x=lapins

Derivation of EM based Alignment Expressions

$$V_E = \text{vocalbulary of language } L_1 \text{ (Say English)}$$

$$V_F = \text{vocabulary of language } L_2 \text{ (Say Hindi)}$$

E^1 *what is in a name ?*
नाम में क्या है?

F^1 *naam meM kya hai ?*
name in what is ?

E^2 *That which we call rose, by any other name will smell as sweet.*
जिसे हम गुलाब कहते हैं, और भी किसी नाम से उसकी कुशबू समान मीठा होगी

F^2 *Jise hum gulab kahte hai, aur bhi kisi naam se uski khushbu samaan mitha hogii*
That which we rose say , any other name by its smell as sweet
That which we call rose, by any other name will smell as sweet.

Vocabulary mapping

Vocabulary

V_E	V_F
what , is , in, a , name , that, which, we , call ,rose, by, any, other, will, smell, as, sweet	*naam, meM, kya, hai, jise, ham, gulab, kahte, aur, bhi, kisi, bhi, uski, khushbu, saman, mitha, hogii*

Key Notations

English vocabulary : V_E
French vocabulary : V_F
No. of observations / sentence pairs : S
Data D which consists of S observations looks like,

$$e^1{}_1, e^1{}_2, \ldots, e^1{}_{l^1} \Leftrightarrow f^1{}_1, f^1{}_2, \ldots, f^1{}_{m^1}$$

$$e^2{}_1, e^2{}_2, \ldots, e^2{}_{l^2} \Leftrightarrow f^2{}_1, f^2{}_2, \ldots, f^2{}_{m^2}$$

.....

$$e^s{}_1, e^s{}_2, \ldots, e^s{}_{l^s} \Leftrightarrow f^s{}_1, f^s{}_2, \ldots, f^s{}_{m^s}$$

.....

$$e^S{}_1, e^S{}_2, \ldots, e^S{}_{l^S} \Leftrightarrow f^S{}_1, f^S{}_2, \ldots, f^S{}_{m^S}$$

No. words on English side in s^{th} sentence : l^s
No. words on French side in s^{th} sentence : m^s
$index_E(e^s{}_p)$ =Index of English word $e^s{}_p$in English vocabulary/dictionary
$index_F(f^s{}_q)$ =Index of French word $f^s{}_q$in French vocabulary/dictionary

(Thanks to Sachin Pawar for helping with the maths formulae processing)

Hidden variables and parameters

Hidden Variables (Z) :
Total no. of hidden variables $= \sum_{s=1}^{S} l^s m^s$ where each hidden variable is as follows:
$z_{pq}^s = 1$, if in s^{th} sentence, p^{th} English word is mapped to q^{th} French word.
$z_{pq}^s = 0$, otherwise

Parameters (Θ) :
Total no. of parameters $= |V_E| \times |V_F|$, where each parameter is as follows:
$P_{i,j} =$ Probability that i^{th} word in English vocabulary is mapped to j^{th} word in French vocabulary

Likelihoods

Data Likelihood *L(D; Θ)* :

$$L(D;\Theta) = \prod_{s=1}^{S} \prod_{p=1}^{l^s} \prod_{q=1}^{m^s} \left(P_{index_E(e_p^s), index_F(f_q^s)} \right)^{z_{pq}^s}$$

Data Log-Likelihood LL(D; Θ) :

$$LL(D;\Theta) = \sum_{s=1}^{S} \sum_{p=1}^{l^s} \sum_{q=1}^{m^s} z_{pq}^s \log \left(P_{index_E(e_p^s), index_F(f_q^s)} \right)$$

Expected value of Data Log-Likelihood E(LL(D; Θ)) :

$$E(LL(D;\Theta)) = \sum_{s=1}^{S} \sum_{p=1}^{l^s} \sum_{q=1}^{m^s} E(z_{pq}^s) \log \left(P_{index_E(e_p^s), index_F(f_q^s)} \right)$$

Constraint and Lagrangian

$$\sum_{j=1}^{|V_F|} P_{i,j} = 1 \ , \forall i$$

$$\sum_{s=1}^{S} \sum_{p=1}^{l^s} \sum_{q=1}^{m^s} E(z_{pq}^s) \ log \left(P_{index_E(e_p^s), index_F(f_q^s)} \right) - \sum_{i=1}^{|V_E|} \lambda_i \left(\sum_{j=1}^{|V_F|} P_{i,j} - 1 \right)$$

Differentiating wrt P_{ij}

$$\sum_{s=1}^{S}\sum_{p=1}^{l^s}\sum_{q=1}^{m^s}\delta_{index_E(e_p^s),i}\,\delta_{index_F(f_q^s),j}\left(\frac{E(z_{pq}^s)}{P_{i,j}}\right)-\lambda_i=0$$

$$P_{i,j}=\frac{1}{\lambda_i}\sum_{s=1}^{S}\sum_{p=1}^{l^s}\sum_{q=1}^{m^s}\delta_{index_E(e_p^s),i}\,\delta_{index_F(f_q^s),j}E(z_{pq}^s)$$

$$\sum_{j=1}^{|V_F|}P_{i,j}=1=\sum_{j=1}^{|V_F|}\frac{1}{\lambda_i}\sum_{s=1}^{S}\sum_{p=1}^{l^s}\sum_{q=1}^{m^s}\delta_{index_E(e_p^s),i}\,\delta_{index_F(f_q^s),j}E(z_{pq}^s)$$

Final E and M steps

M-step

$$P_{i,j} = \frac{\sum_{s=1}^{S}\sum_{p=1}^{l^s}\sum_{q=1}^{m^s}\delta_{index_E(e_p^s),i}\,\delta_{index_F(f_q^s),j}E(z_{pq}^s)}{\sum_{j=1}^{|V_F|}\sum_{s=1}^{S}\sum_{p=1}^{l^s}\sum_{q=1}^{m^s}\delta_{index_E(e_p^s),i}\,\delta_{index_F(f_q^s),j}E(z_{pq}^s)}, \forall i,j$$

E-step

$$E(z_{pq}^s) = \frac{P_{index_E(e_p^s),index_F(f_q^s)}}{\sum_{q'=1}^{m^s}P_{index_E(e_p^s),index_F(f_{q'}^s)}}, \forall s,p,q$$

Tools that implement word alignment

- Giza++ which comes with Moses
- Berkeley Aligner

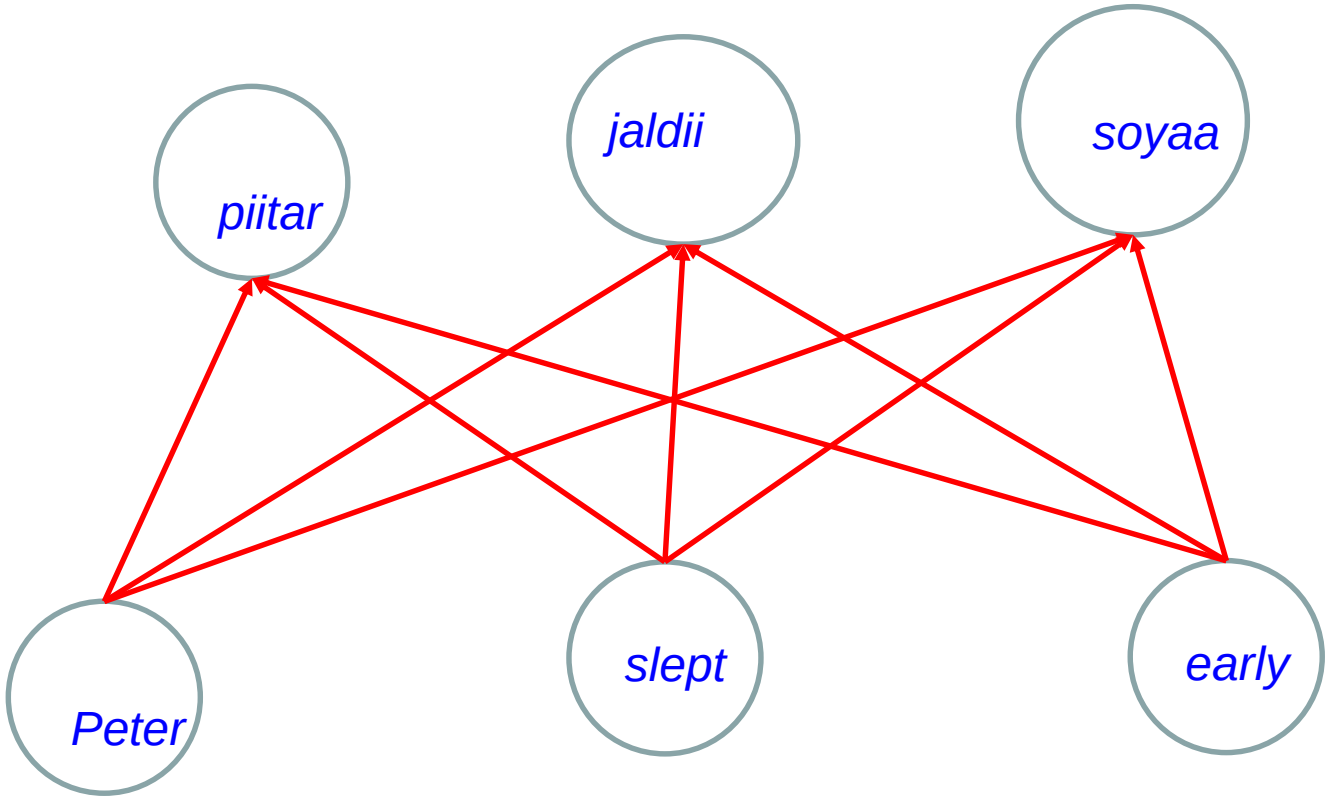
Attention and Alignment

Attention and Alignment

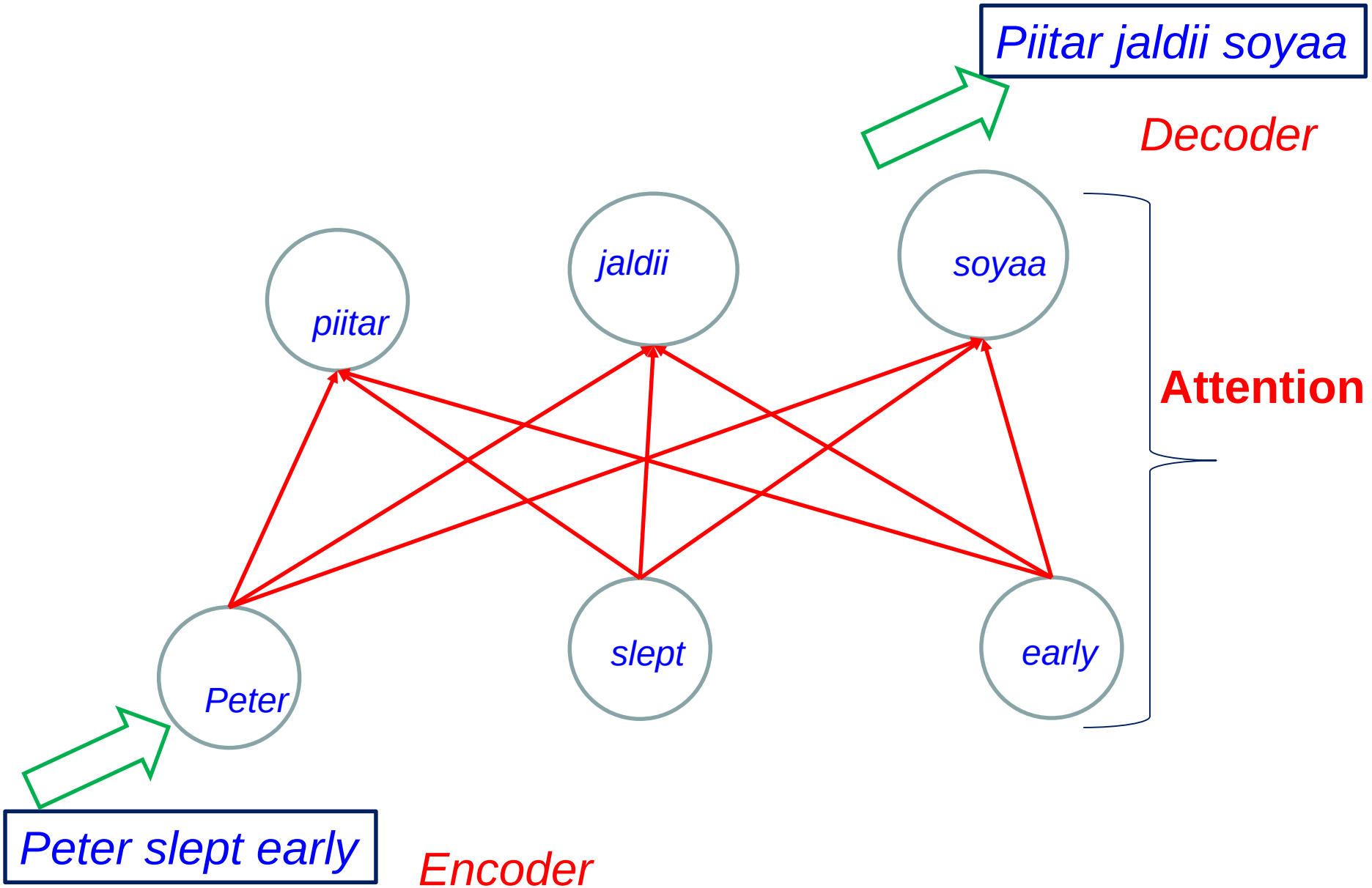
Hindi (col) --> English (row) | \/	PIITAR (पीटर)		JALDII (जल्दी)		SOYAA (सोया)
PETER	1		0		0
SLEPT	0		0		1
EARLY	0		1		0

FFNN for alignment:

Peter slept early → piitar jaldii soyaa

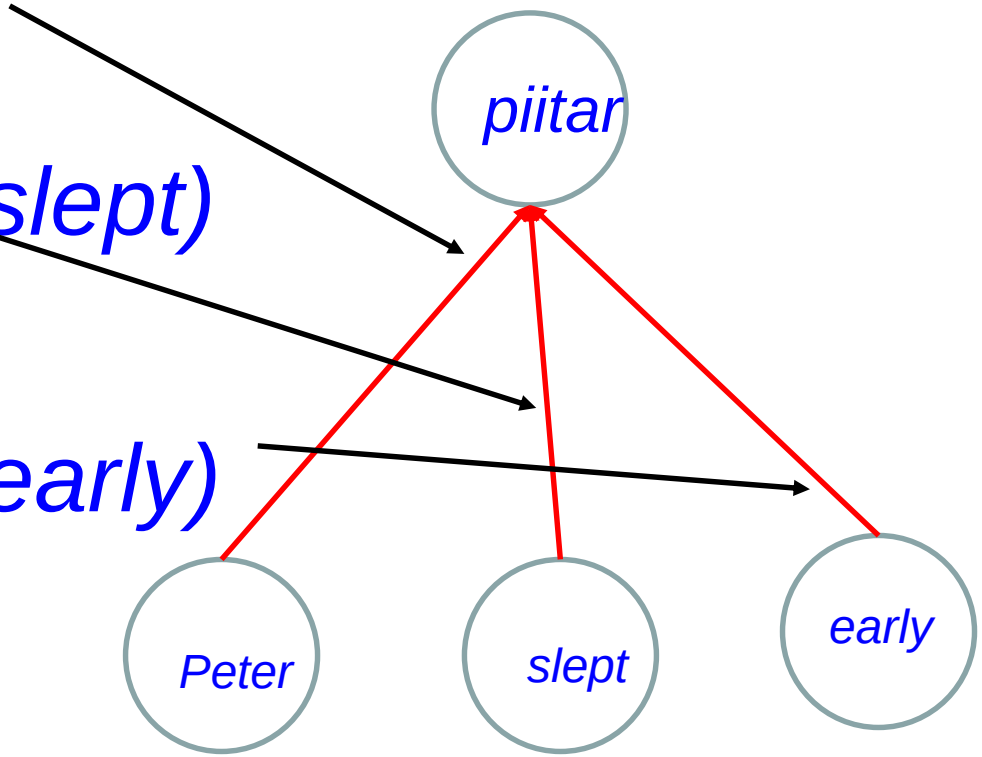


Introduce Attention Layer between Encoder and Decoder



How to learn the weights- attention weights?

- Weight (*piitar, peter)*
- Weight (*piitar, slept)*
- Weight (*piitar, early)*



Attention: Linguistic and Cognitive View

Long distance dependency: WSD

The *bank*

Long distance dependency: WSD

The *bank* that Ram

Long distance dependency: WSD

The *bank* that Ram used to visit

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed due to

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed due to the lockdown

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed due to the lockdown with the Govt

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed due to the lockdown with the Govt. getting worried that

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed due to the lockdown with the Govt. getting worried that crowding of people

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed due to the lockdown with the Govt. getting worried that crowding of people during the

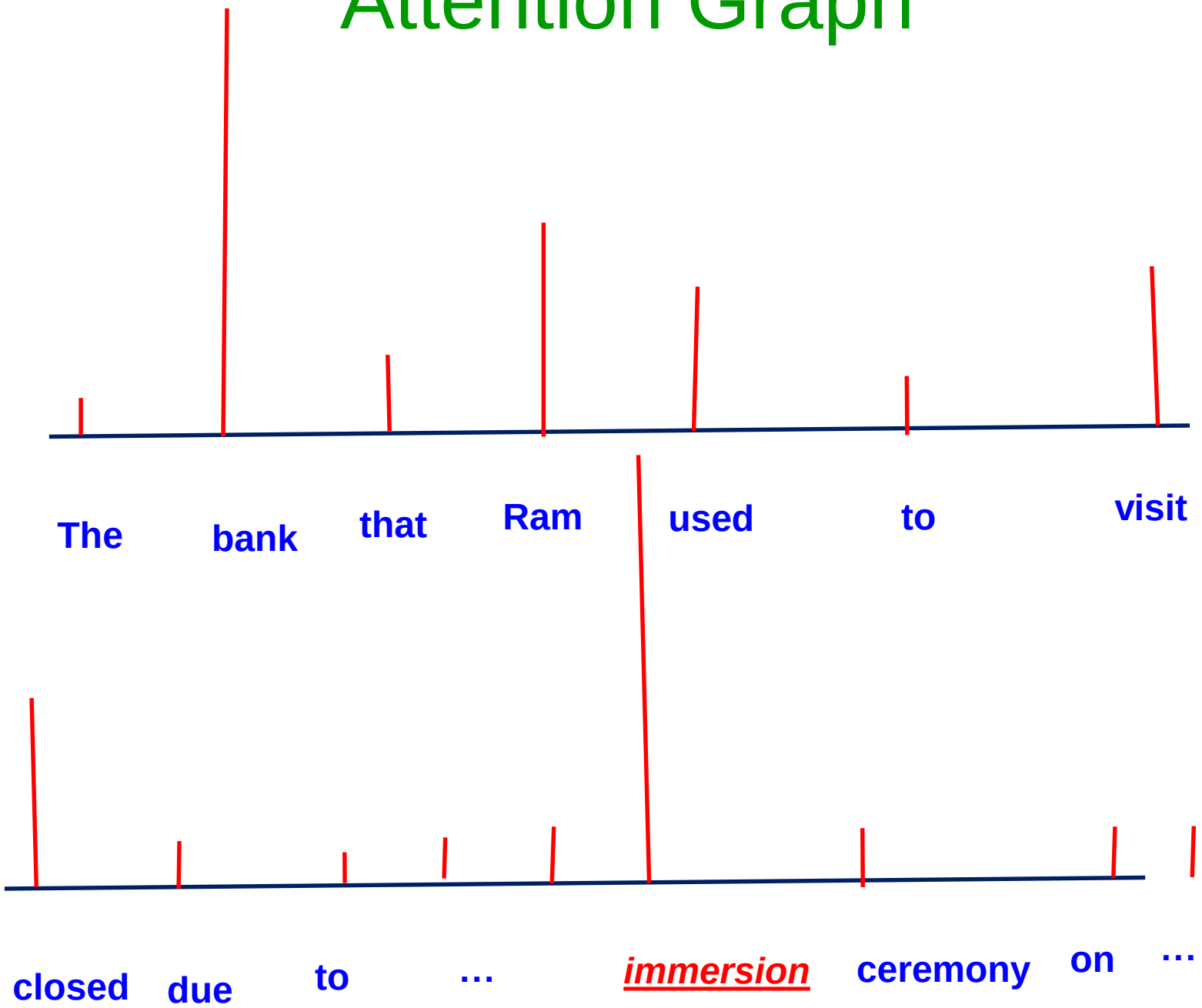
Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed due to the lockdown with the Govt. getting worried that crowding of people during the immersion ceremony

Long distance dependency: WSD

The *bank* that Ram used to visit 30 years before was closed due to the lockdown with the Govt. getting worried that crowding of people during the immersion ceremony on the river will aggravate the situation.

Attention Graph



Different forms of Attention

- Morphological Attention: For predicting the token ‘jayega’, attention should be given to the token ‘Ram’ from the morphological perspective in order to render the correct form of the verb (in gender, person, number etc)
- Shallow Parsing Attention: The previous two tokens e.g might carry enough syntactic context for predicting the correct part of speech at a given position.
- Semantic Attention: From the sentence for example, ‘university’ and ‘higher studies’ are semantically related.

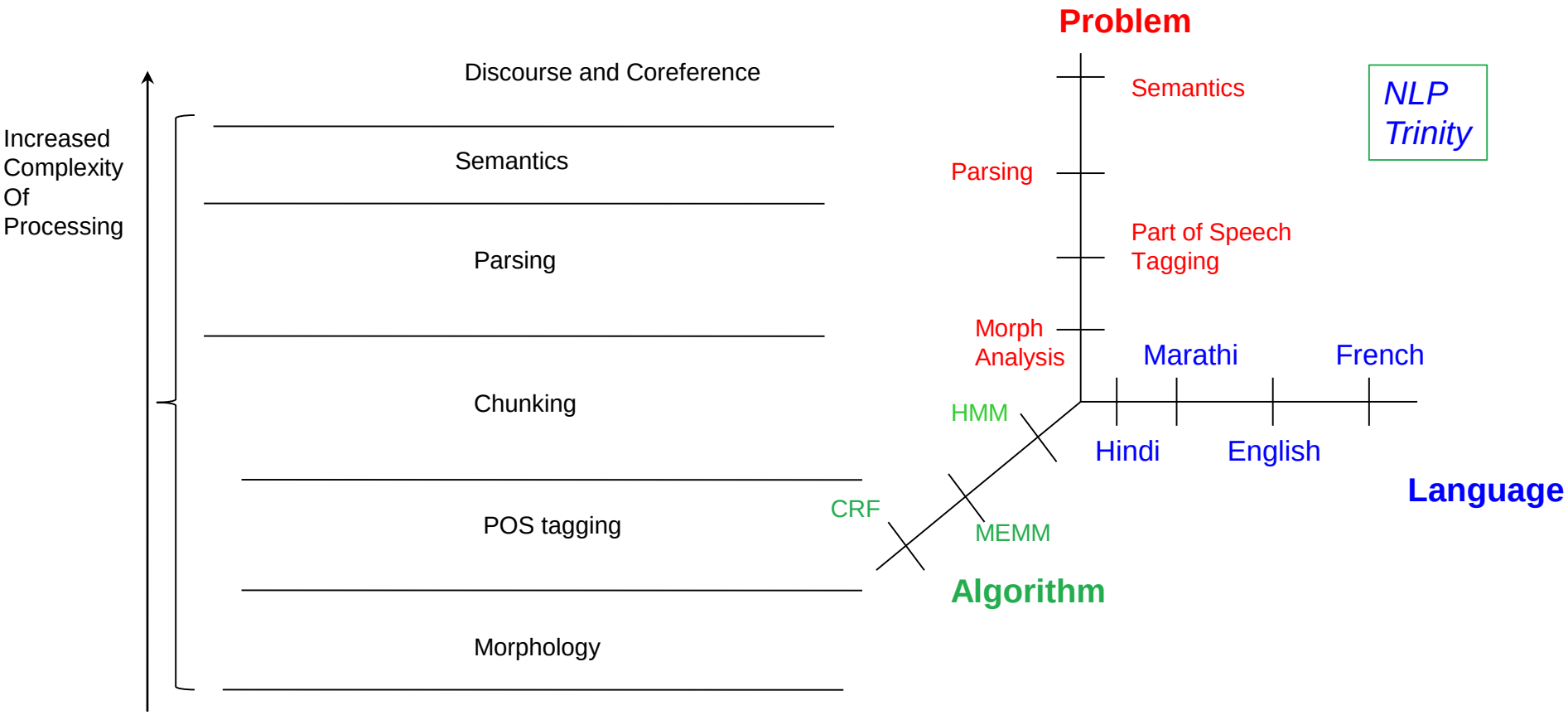
Backward Chaining for learning Attention

As an example, for different forms of jaana: jayega, jayegi, jayenge, jaoge etc

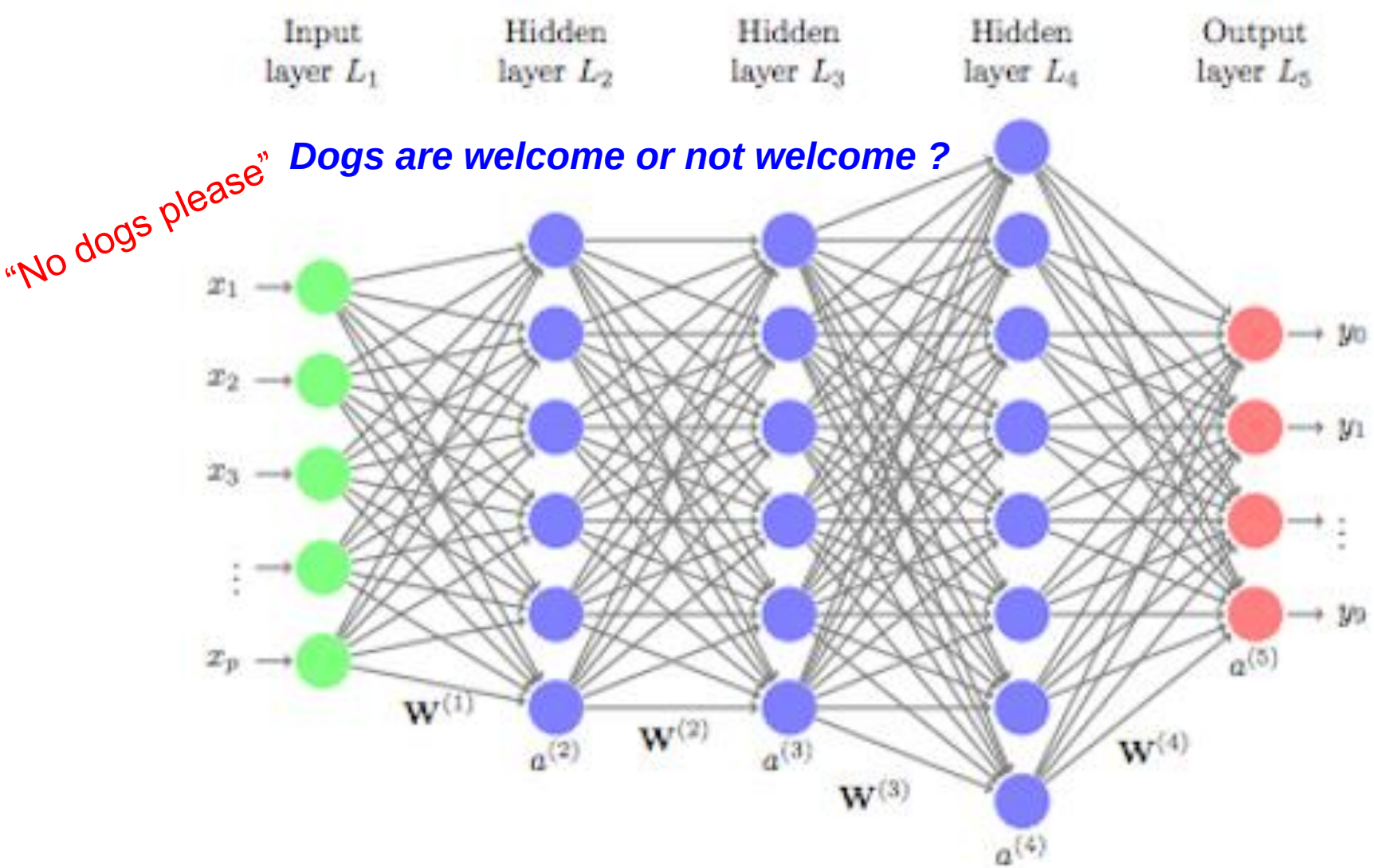
- We have to produce a ranking for all the different forms of ‘jaana’.
- The ranking is based on probabilities, in particular the softmax computation.
- Softmax depends on computing e^{net_input}.
- The net input is computed from the dot product of word vectors...

Feedforward network and Backpropagation

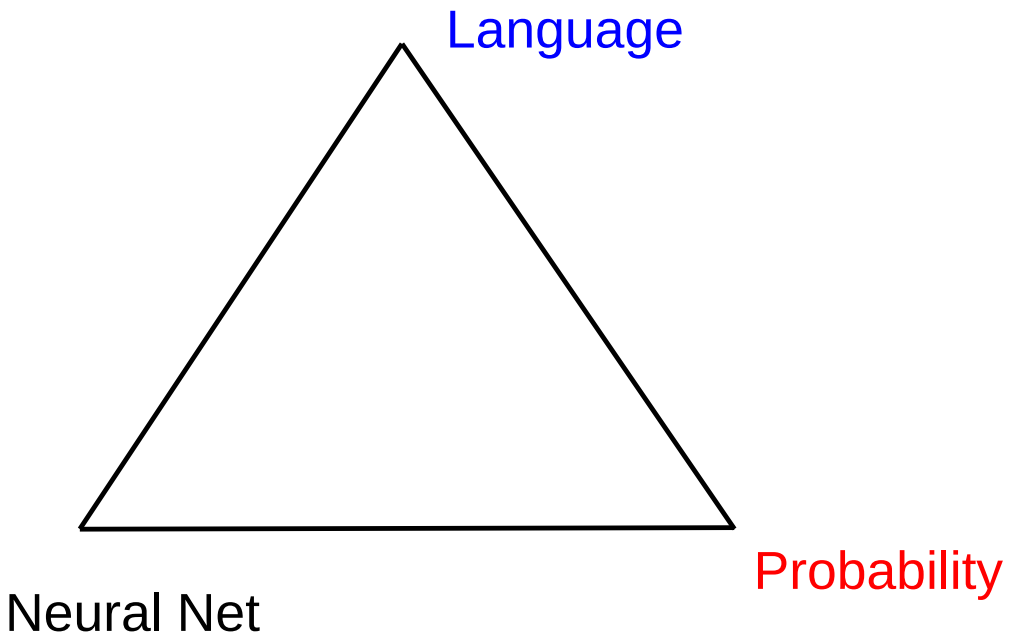
NLP is layered Processing, Multidimensional too



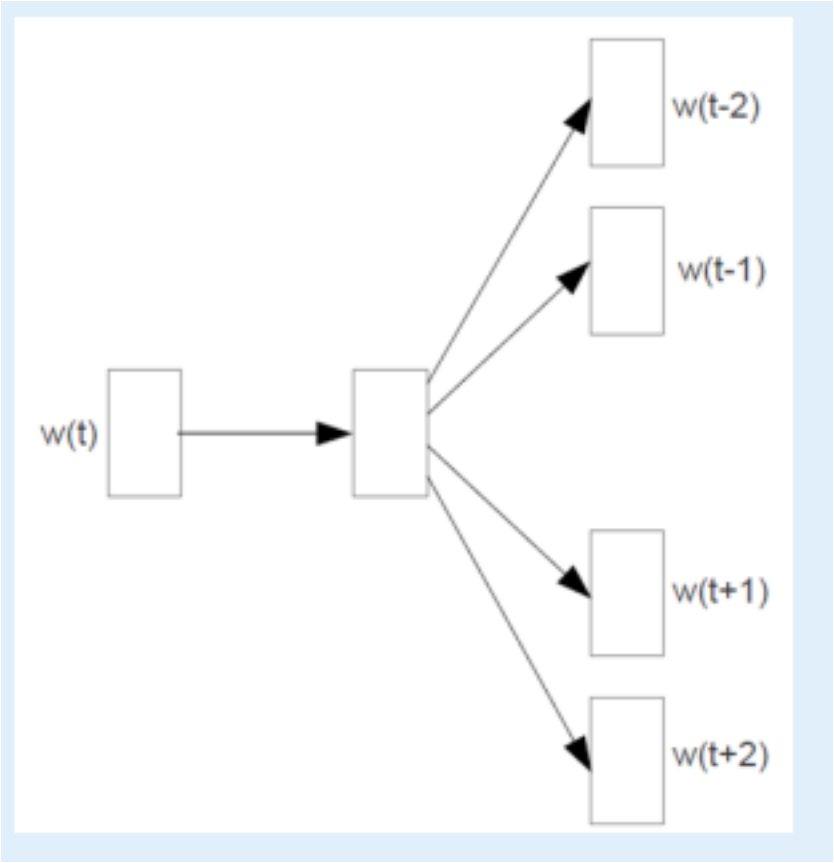
Deep Neural Nets and NLP: layer to layer correspondence



Nature of DL-NLP



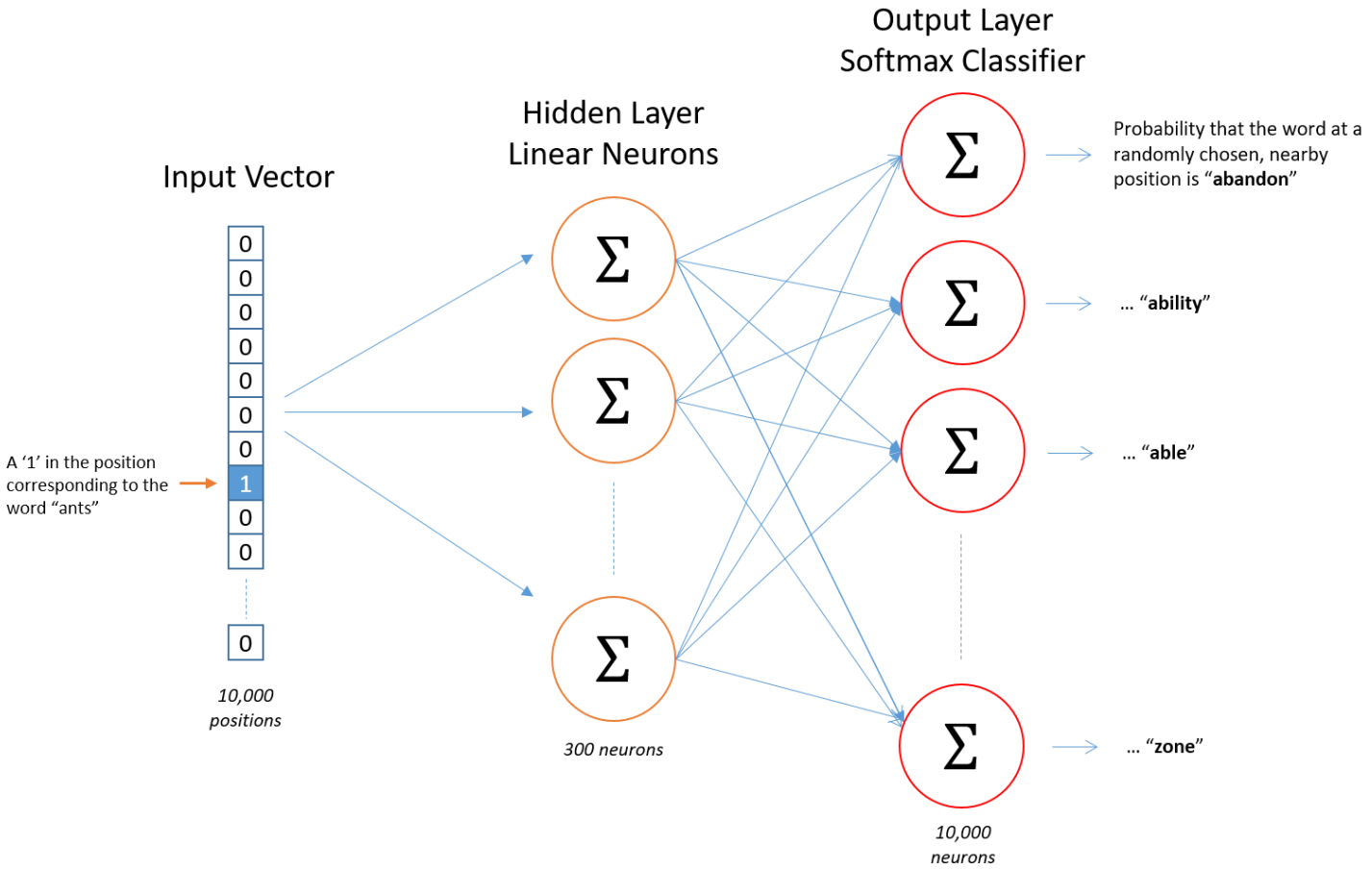
Skip gram- predict context from word



For CBOW:

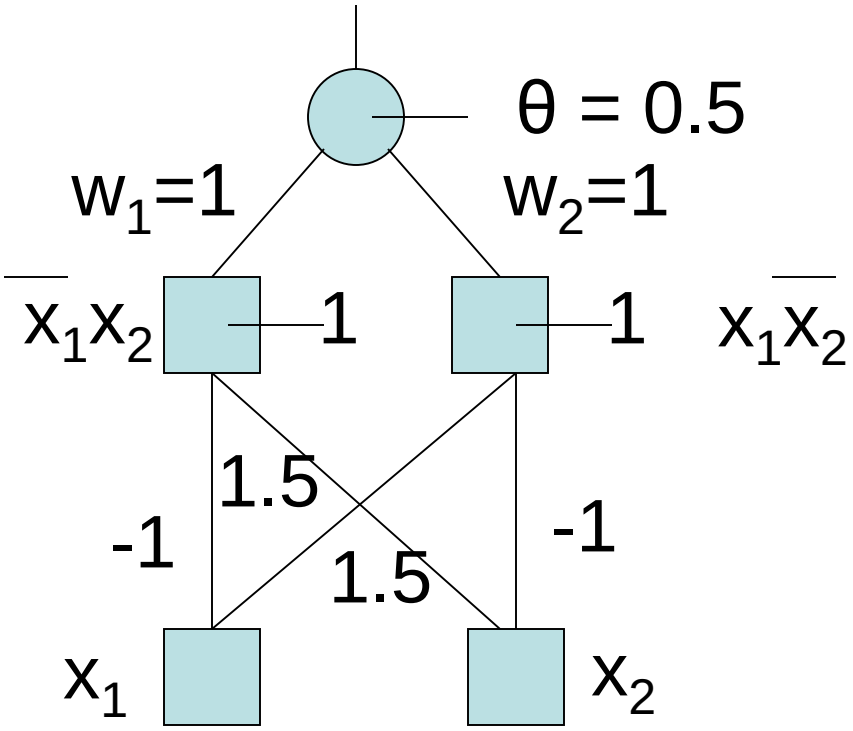
Just reverse the Input-Ouput

Skip Gram: more details

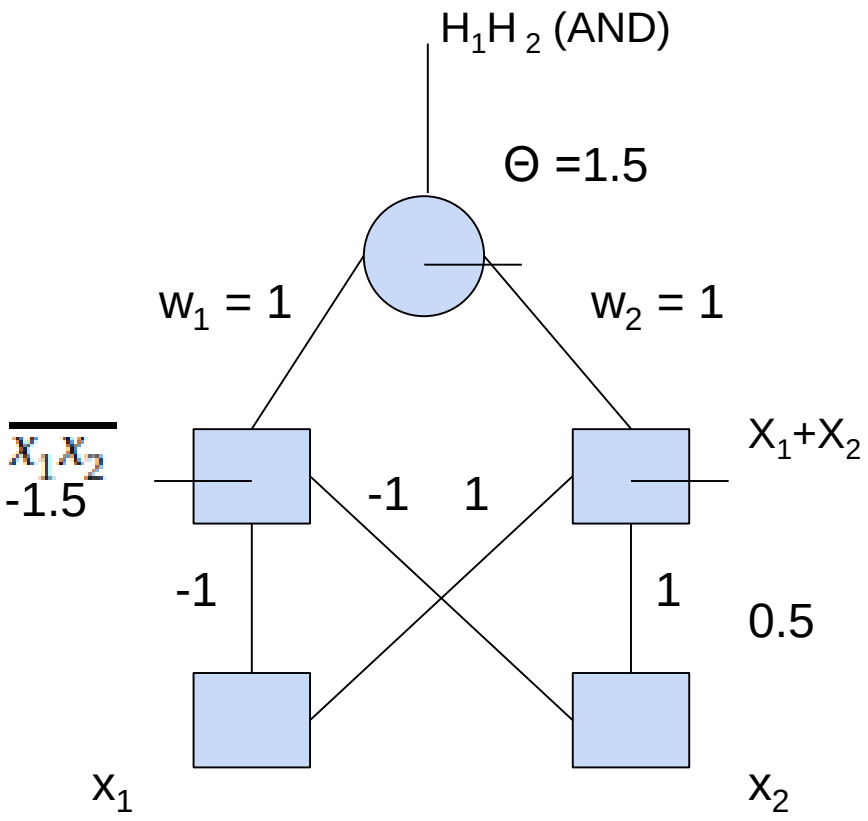


Feedforward Network and Backpropagation

Example - XOR

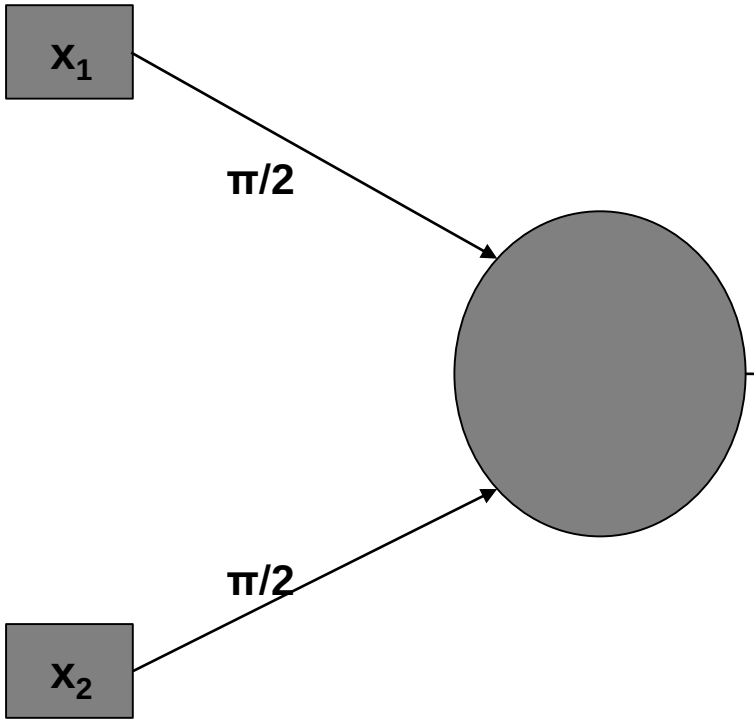


Alternative network for XOR



- 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.

XOR neuron with sin()



$sin\{(x_1+x_2)\pi/2\}$

x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Exercise: Back-propagation

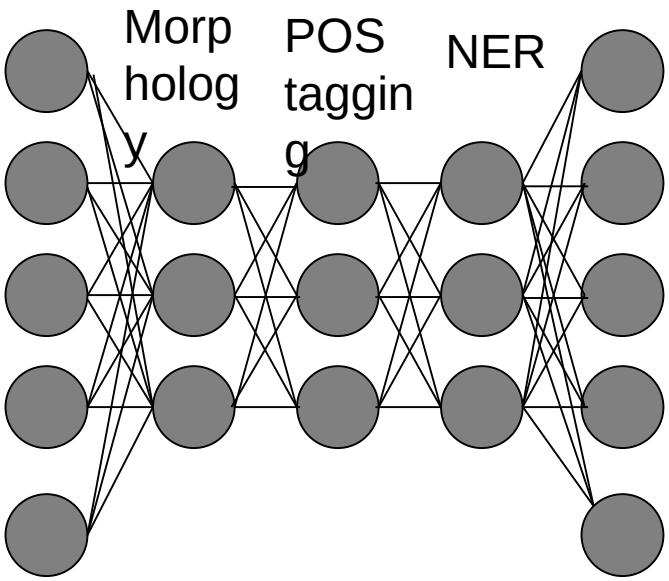
- Implement back-propagation for XOR network

- 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?

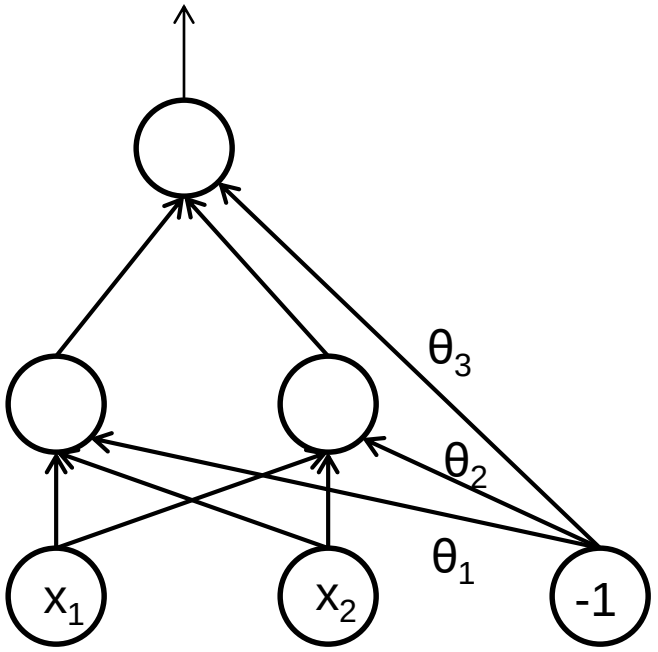
0, 0	0
0, 1	1
1, 0	1
1, 1	0

⇒

-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

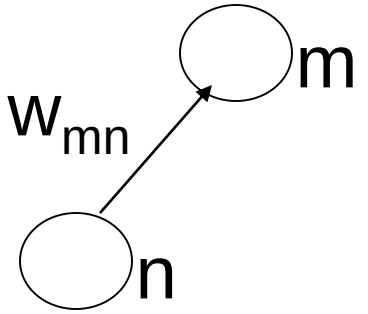
$$E = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{n}(t_i - o_i)_j^2$$

- 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

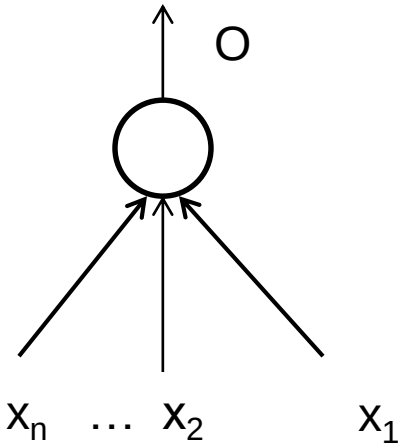
Weights in a FF NN

- w_{mn} is the weight of the connection from the n^{th} neuron to the m^{th} 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



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

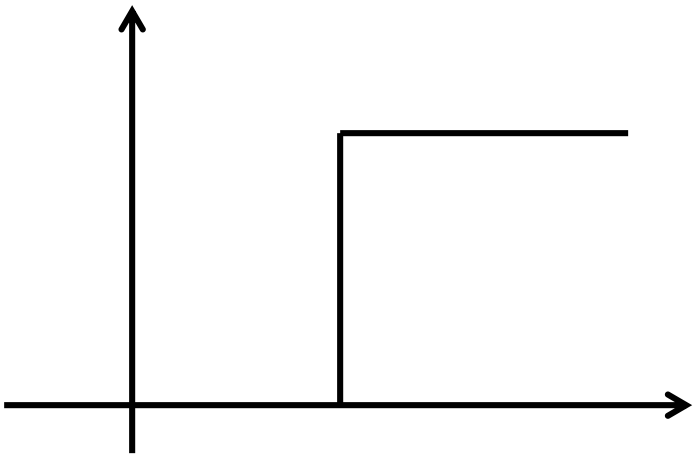
Step function v/s Sigmoid function



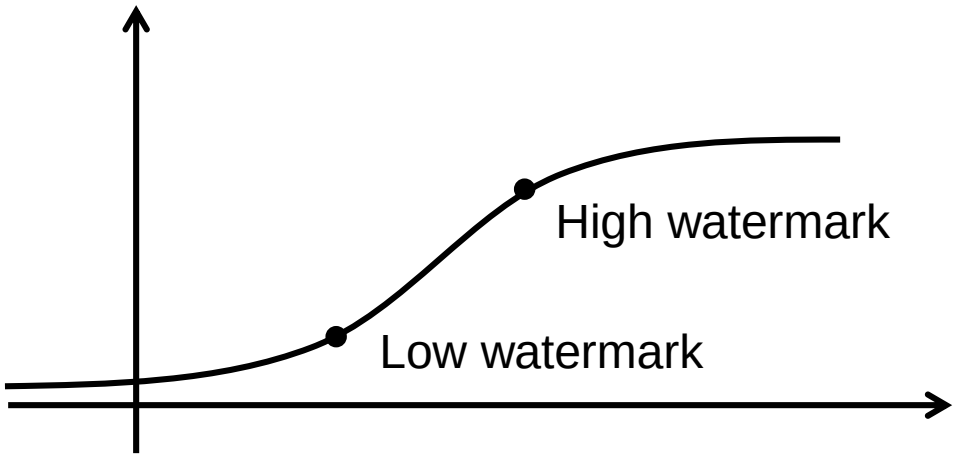
$$O = f(\sum w_i x_i)$$
$$= f(net)$$

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

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



Non-differentiable



Differentiable

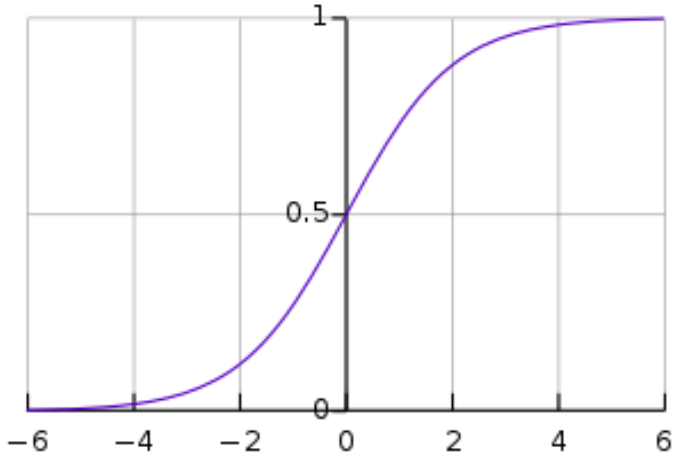
Sigmoid function

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

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

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

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

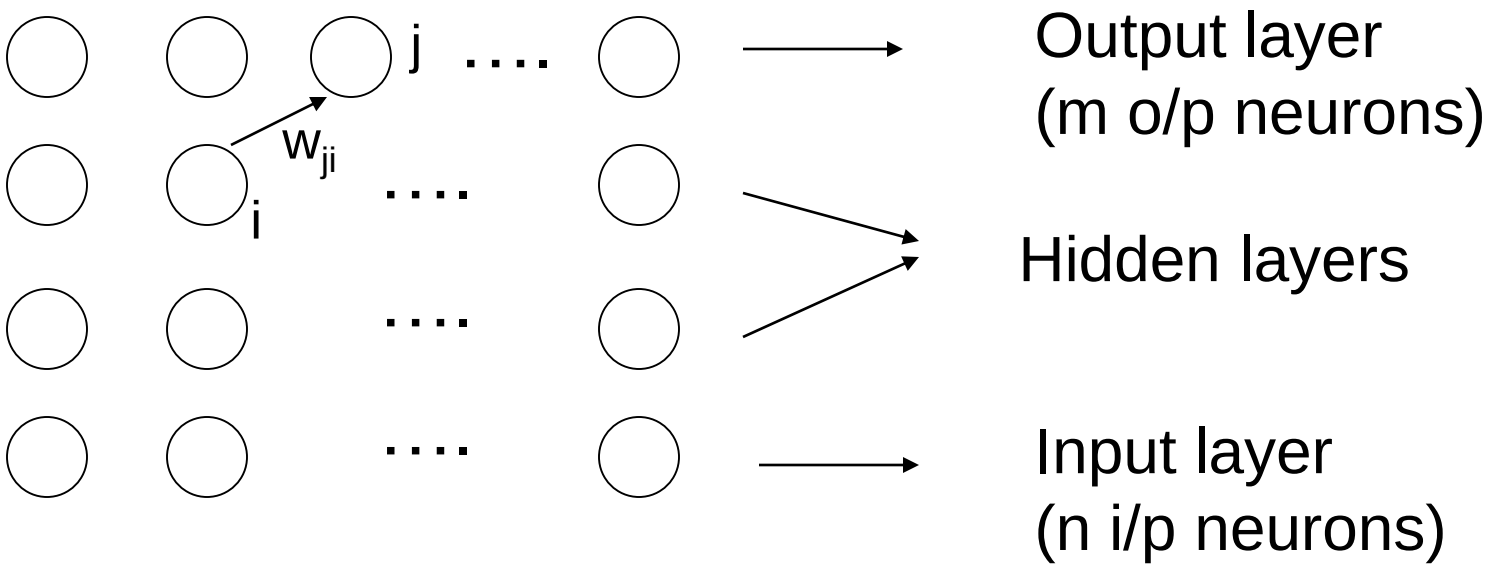
Cross-entropy

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

- 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

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

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

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

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

Backpropagation – for outermost layer

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

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

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

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

Observations from Δw_{ji}

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

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

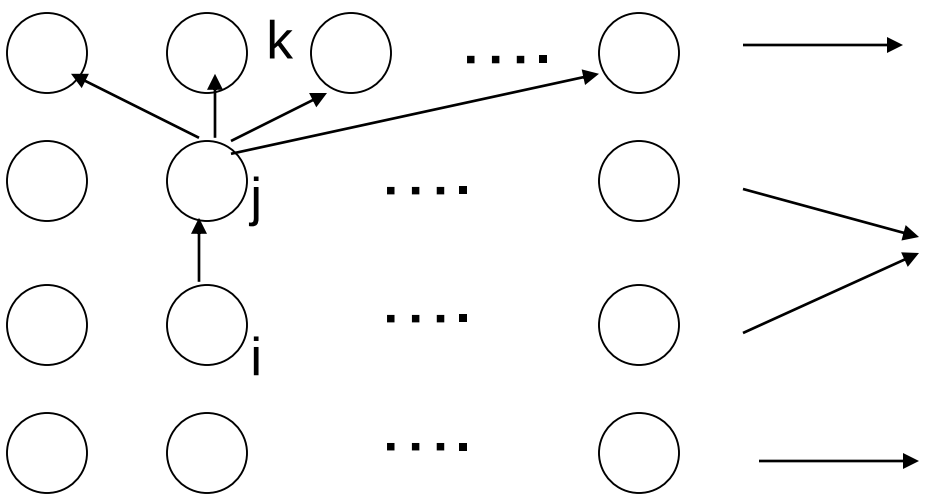
1. $O_j \to t_j$ and/or
2. $O_j \to 1$ and/or
3. $O_j \to 0$ and/or

} Saturation behaviour

4. $O_i \to 0$

} Credit/Blame assignment

Backpropagation for hidden layers



Output layer
(m o/p neurons)

Hidden layers

Input layer
(n i/p neurons)

δ_k is propagated backwards to find value of δ_j

Backpropagation – for hidden layers

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

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

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

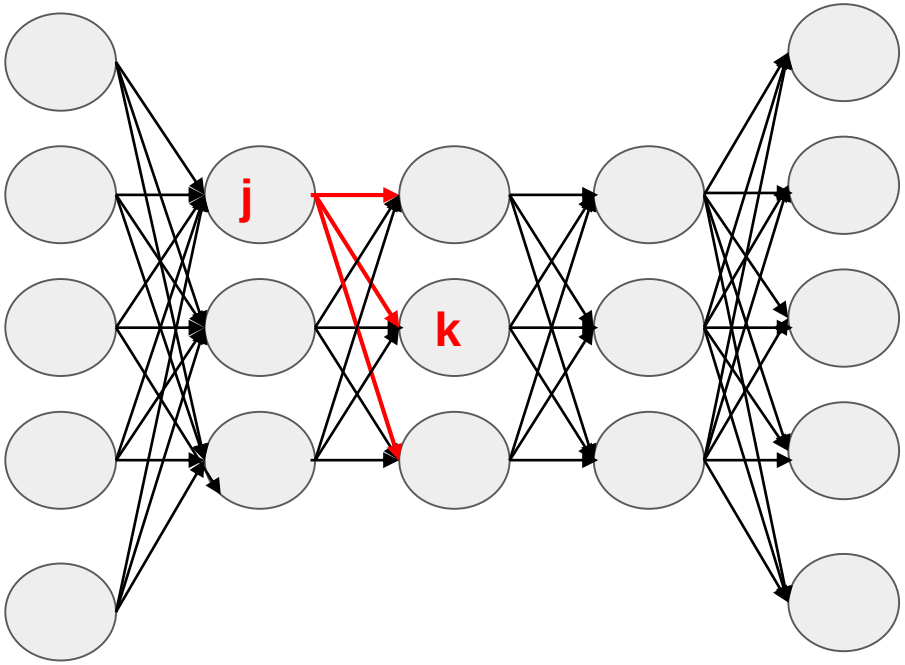
$$= -\sum_{k \in \text{next layer}} \left(\frac{\delta E}{\delta net_k} \times \frac{\delta net_k}{\delta o_j}\right) \times o_j (1 - o_j)$$

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

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

This recursion can give rise to vanishing and exploding Gradient problem

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:

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

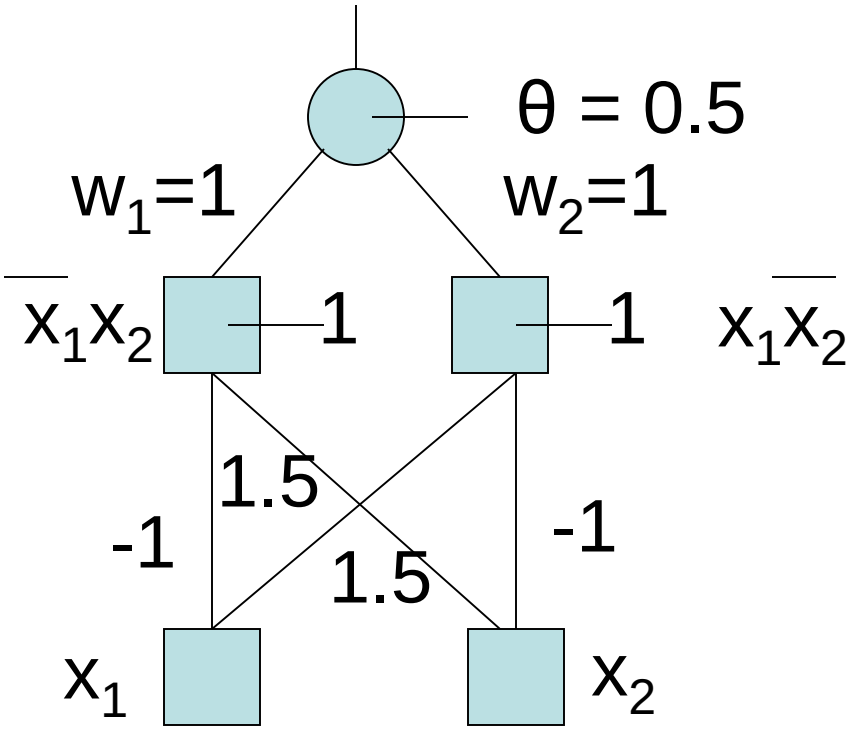
- Where

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

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

How does it work?

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



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*

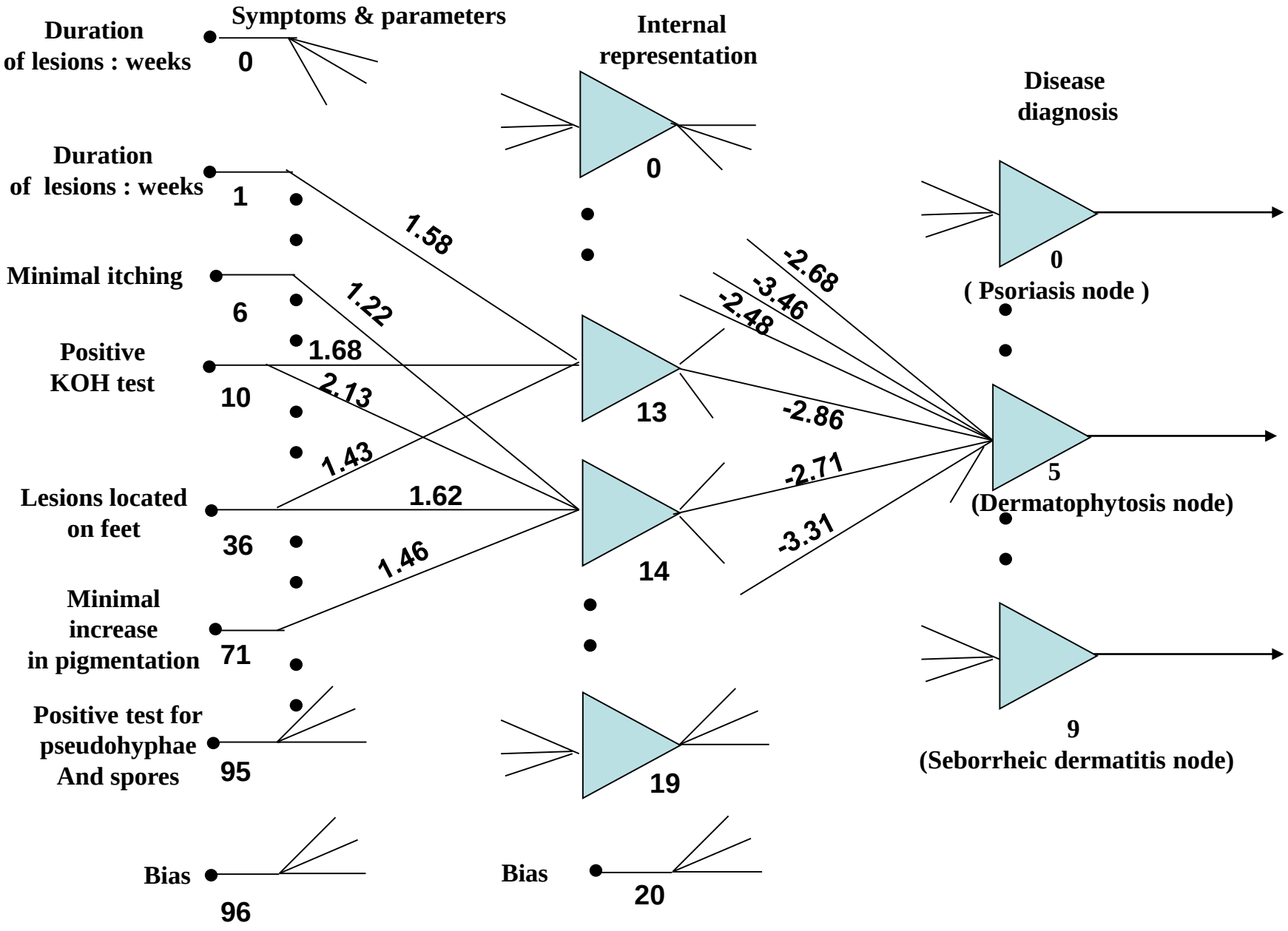


Figure : Explanation of dermatophytosis diagnosis using the DESKNET expert system.

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

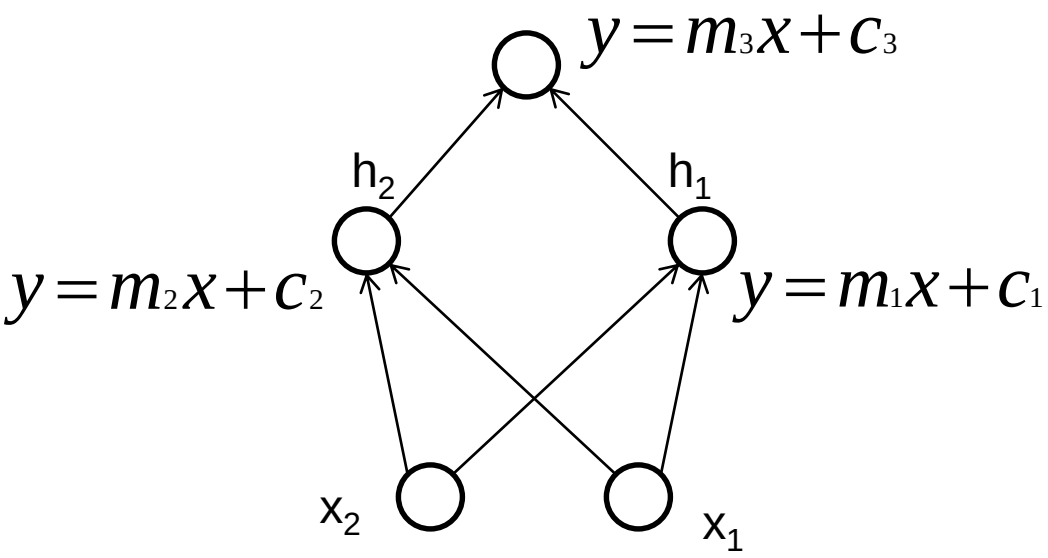
Explanation contd.

- 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

Can Linear Neurons Work?



$$h_1 = m_1(w_1 x_1 + w_2 x_2) + c_1$$

$$h_1 = m_2(w_1 x_1 + w_2 x_2) + c_2$$

$$Out = (w_5 h_1 + w_6 h_2) + c_3$$

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

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: $m.w_1 - m_2.\theta + c < 0.1$

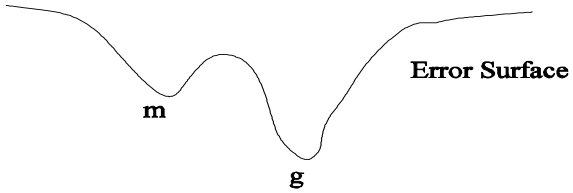
These equations are inconsistent. Hence X-OR can't be computed.

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

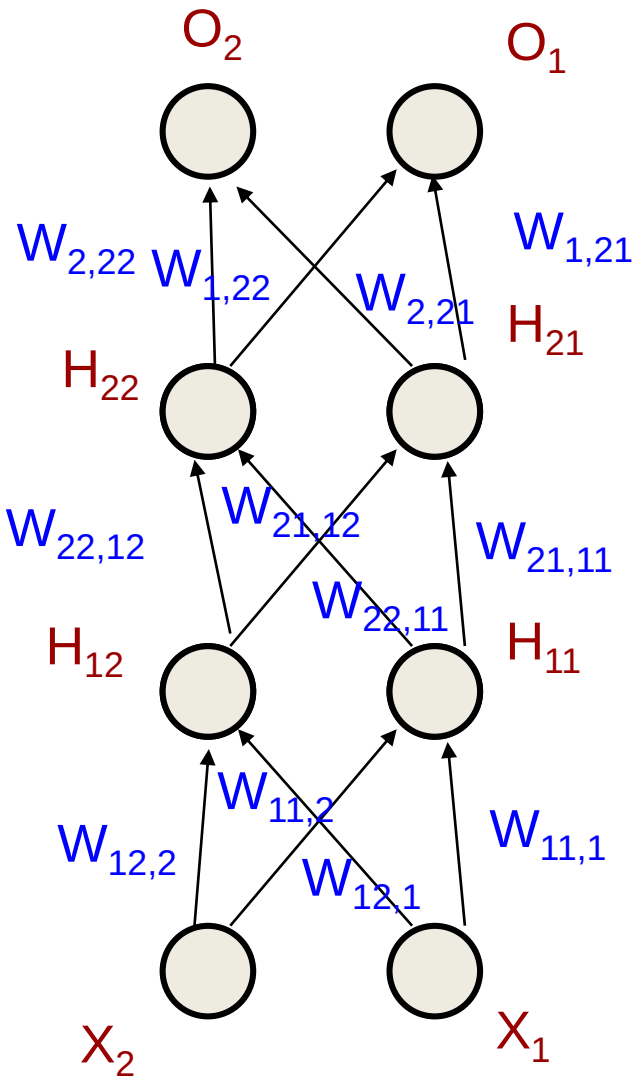
Momentum factor

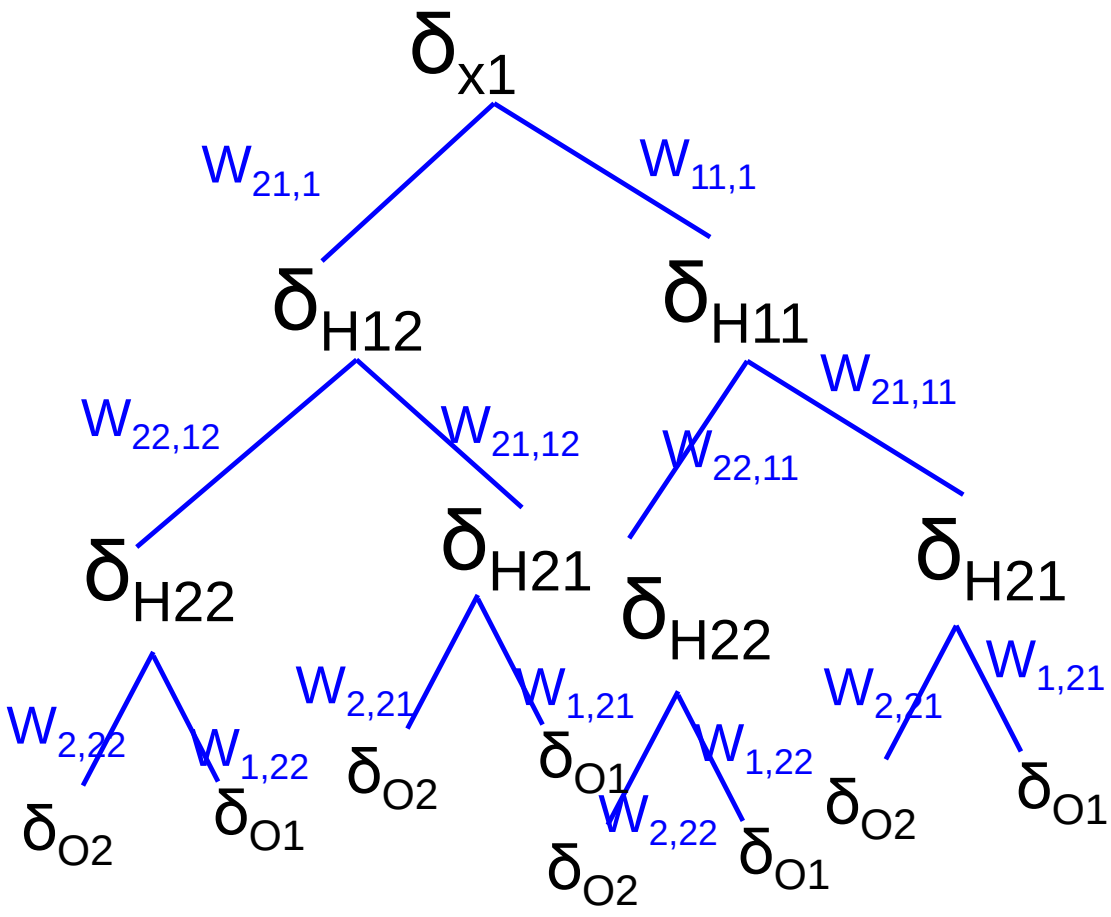
1. Introduce momentum factor.

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

- 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





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

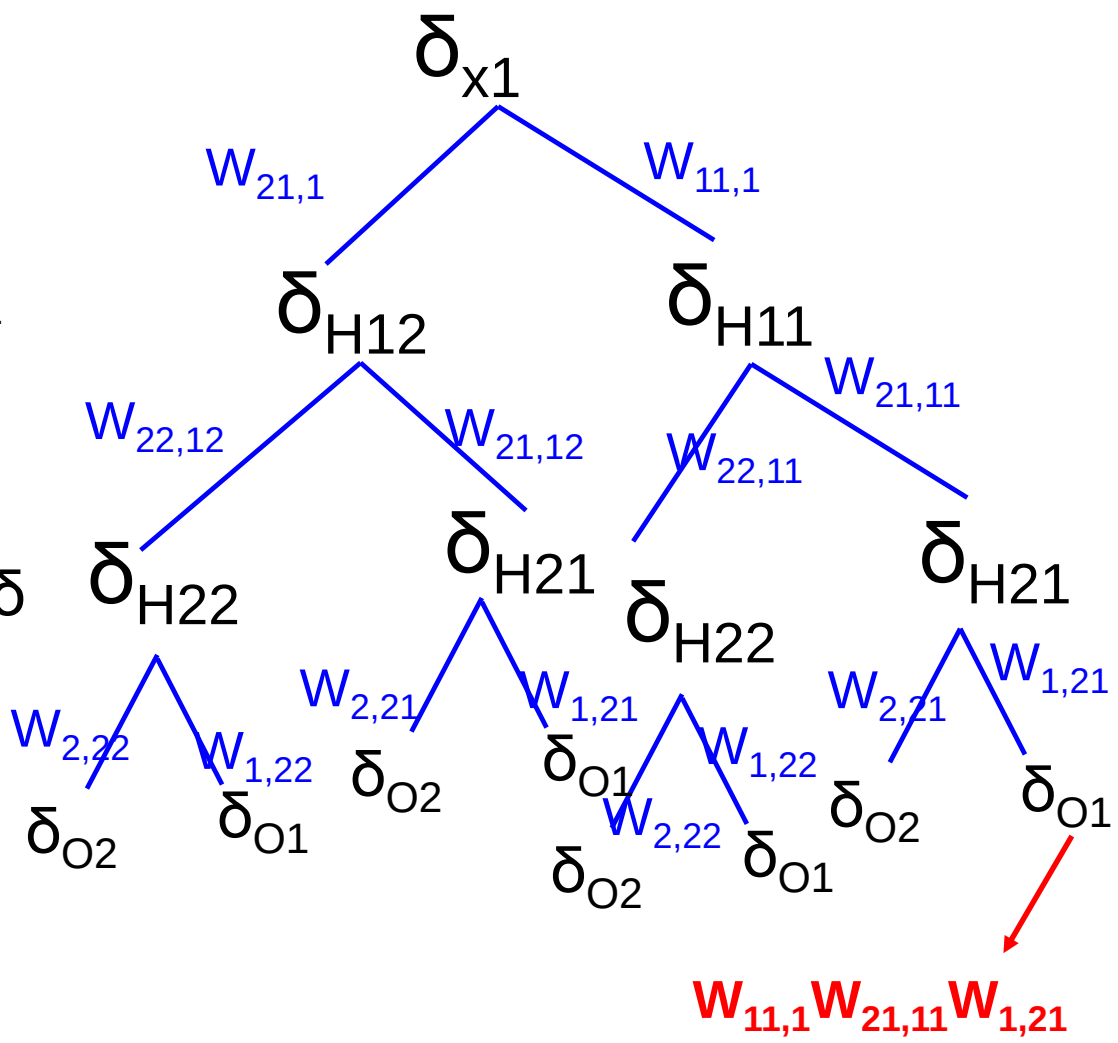
Vanishing/Exploding Gradient

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

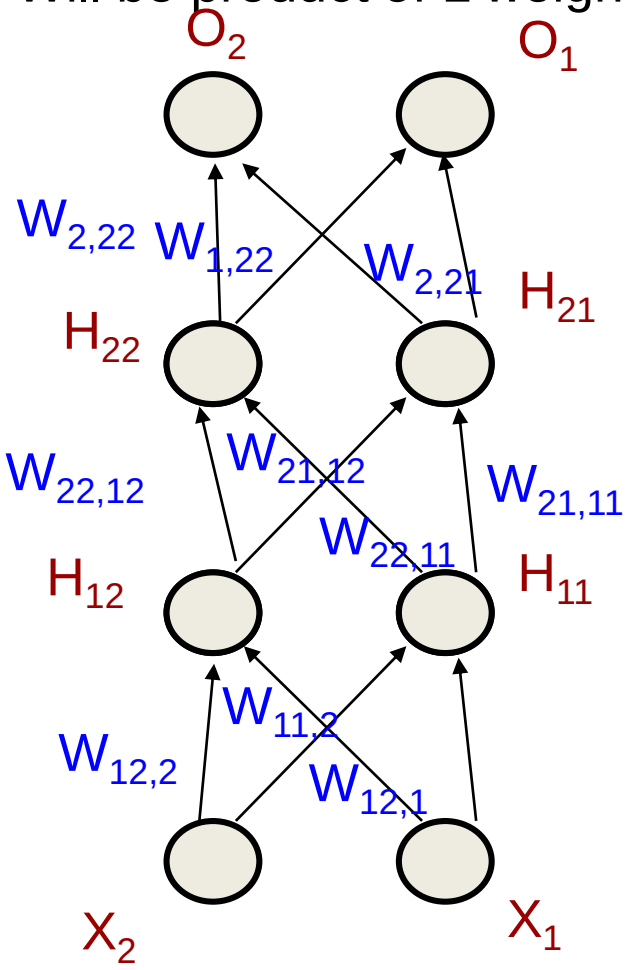
$= W_{11,1}W_{21,11}\delta_{H21}+W_{11,1}W_{22,21}\delta_{H22}+W_{21,1}W_{21,12}\delta_{H21}+W_{21,1}W_{22,12}\delta_{H22}$

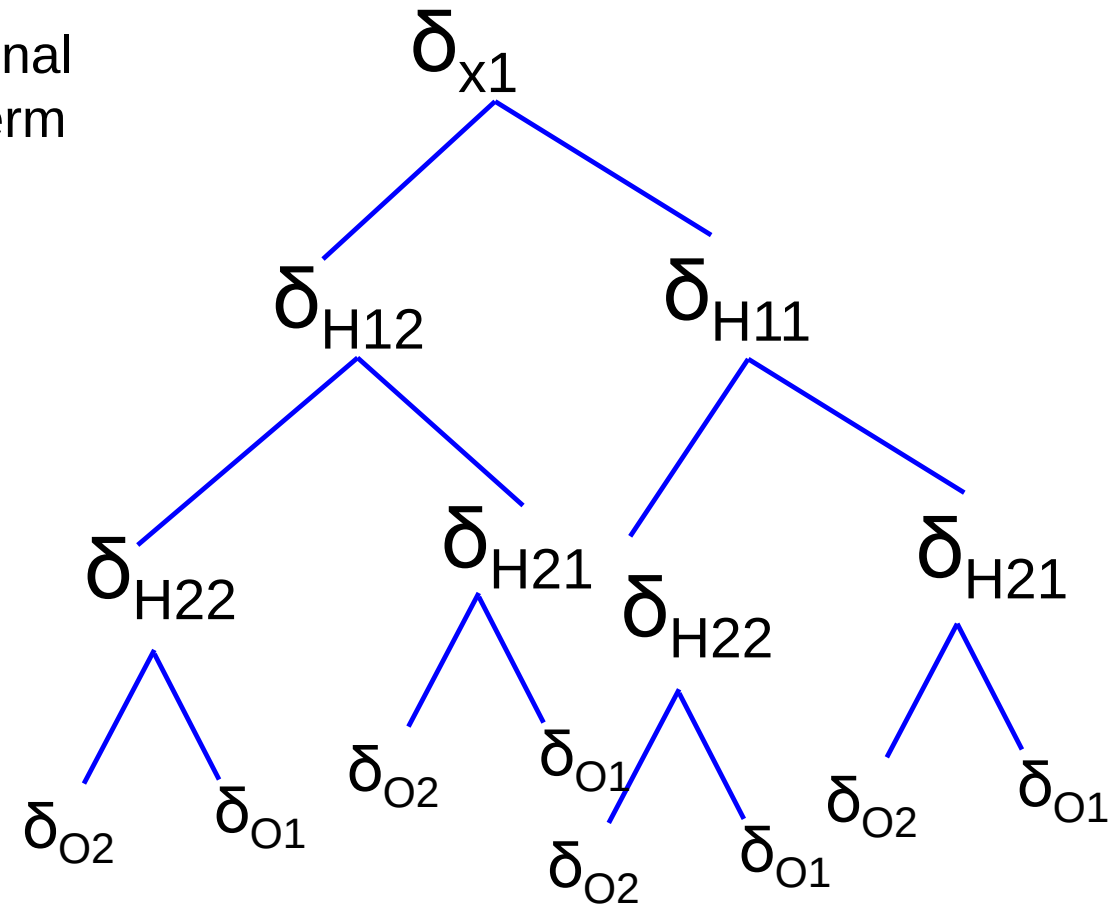
= (4 terms with δ_{o1}) + (4 terms with δ_{o2}; one term shown for the leftmost leaf's weight)



Vanishing/Exploding Gradient

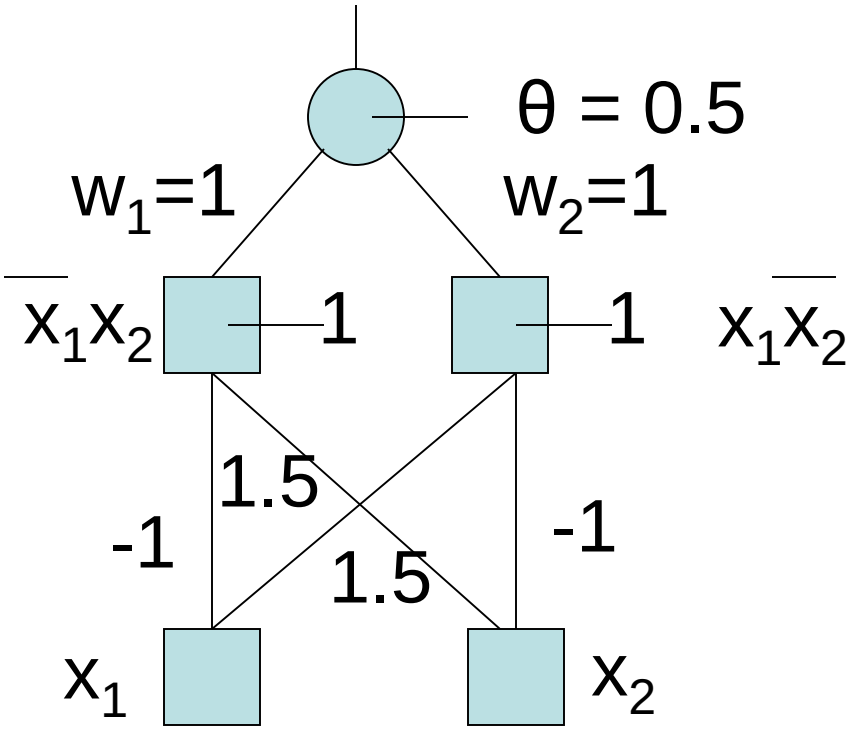
With '*B*' as branching factor and '*L*' as number of levels, There will be B^L terms in the final Expansion of δ_{x1}. Also each term Will be product of L weights





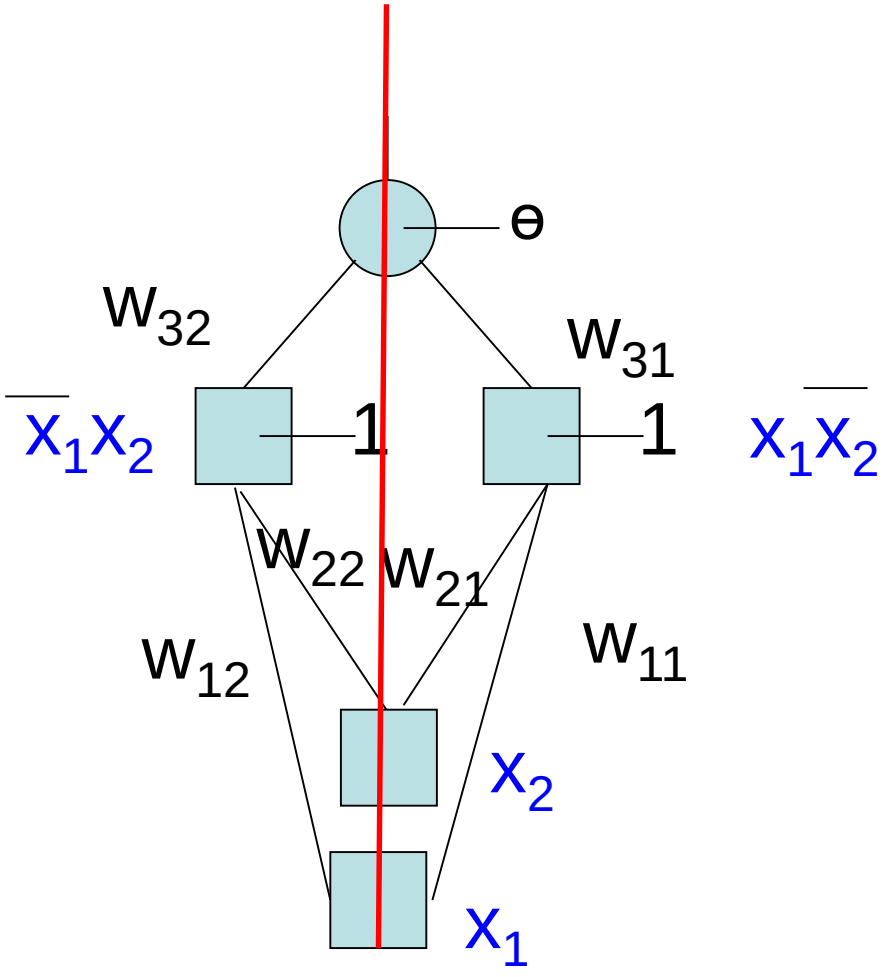
Symmetry breaking

- 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