# CS772: Deep Learning for Natural Language Processing (DL-NLP)

**Fine points of BP, Word Embedding**


Pushpak Bhattacharyya

Computer Science and Engineering Department

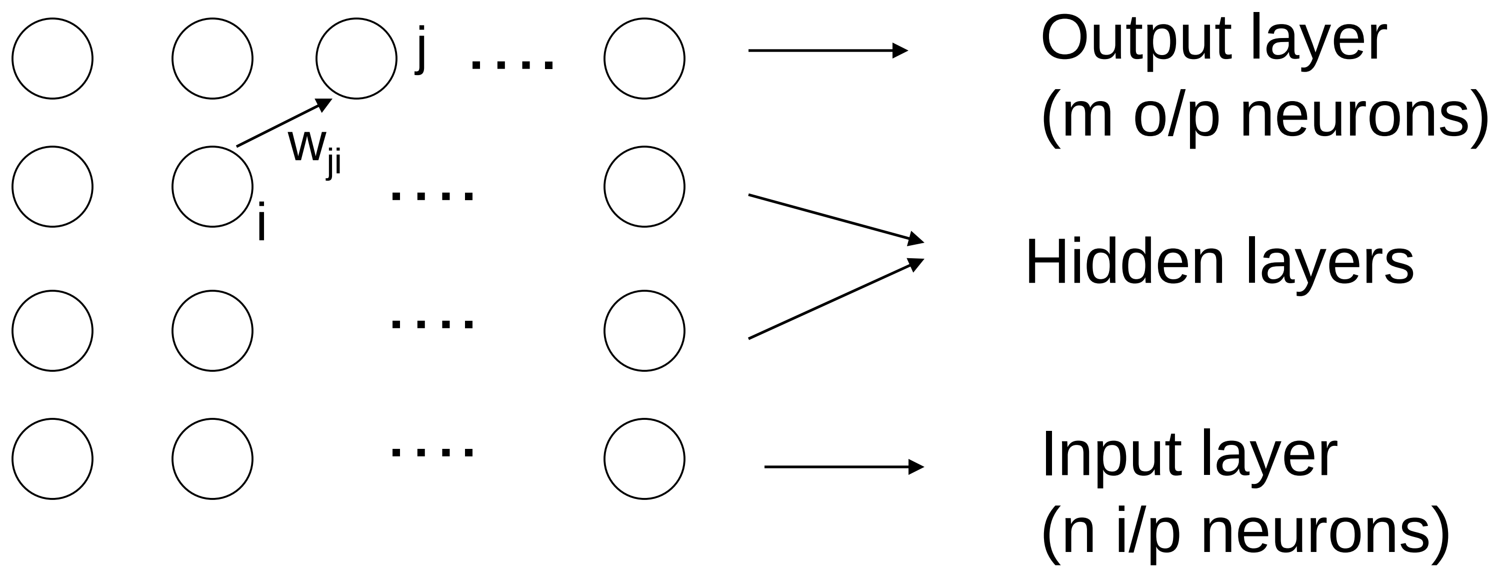
IIT Bombay


*Week 5 of 29jan24*

# 1-slide recap

- Small and Large LMs
- BP weight change rules
- Recurrent Perceptron
- Application of BP- skin disease prediction
- Vanishing gradient
- Derivation of word embeddings

# Backpropagation algorithm



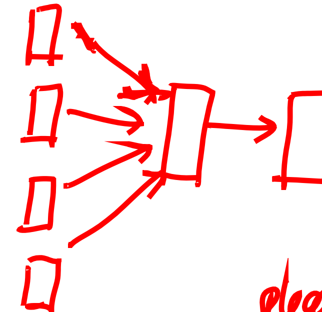

- Fully connected feed forward network
- Pure FF network (no jumping of connections over layers)

Word-2-vec → CBOW
→ skip-gram

Captures semantics of the word/word embedding.

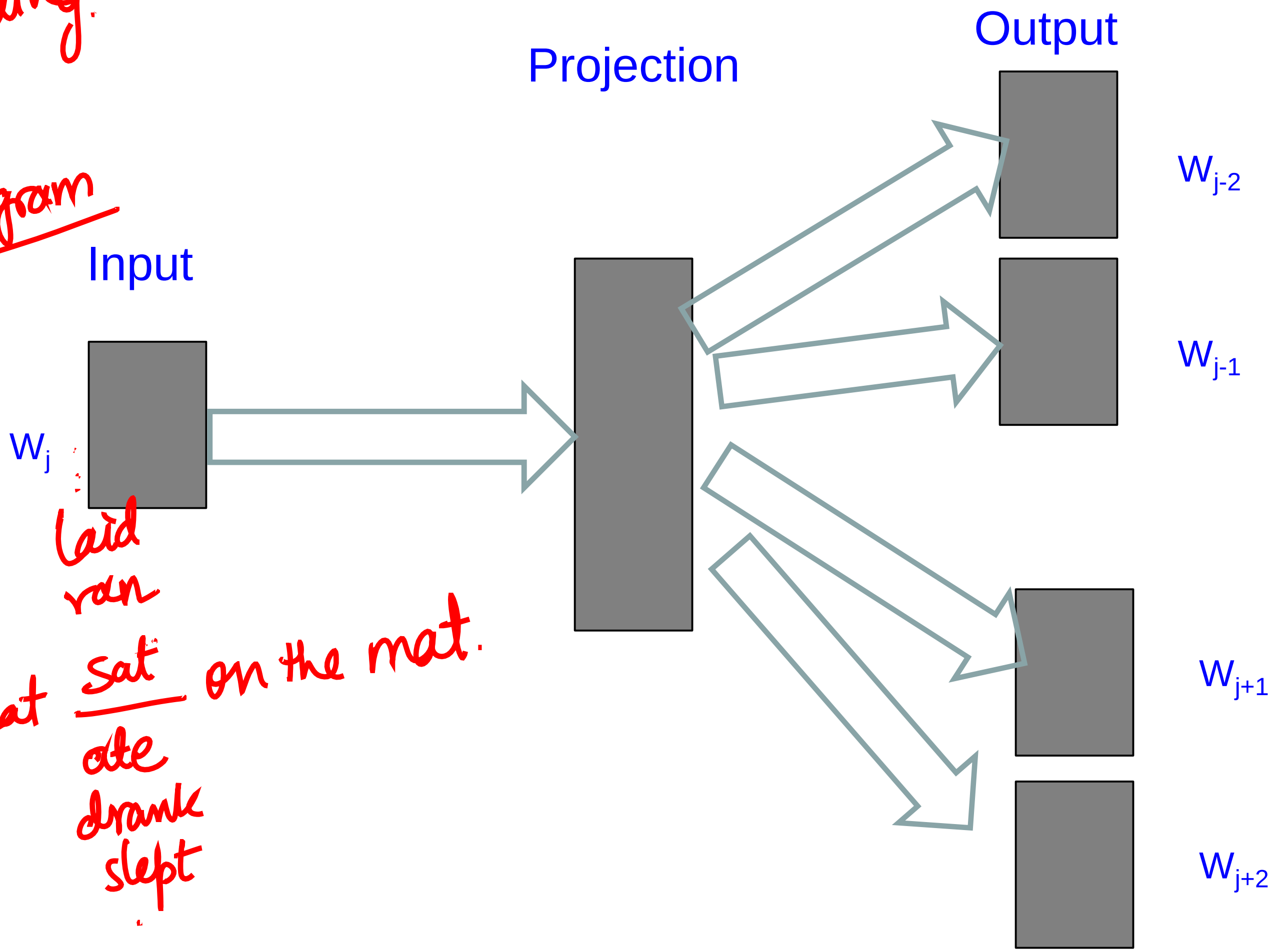
# Modelling $p(w_{t+j}|w_t)$

CBOW



today is a ___ day
(pleasant)

skip-gram

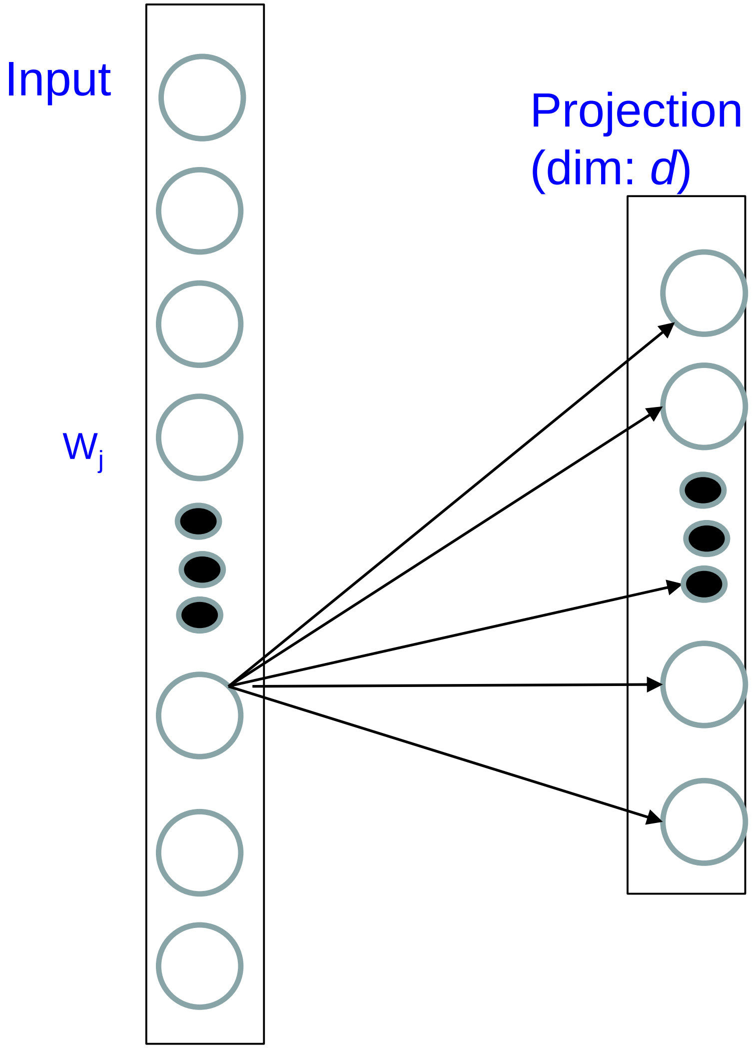



laid
ran

the cat sat on the mat.

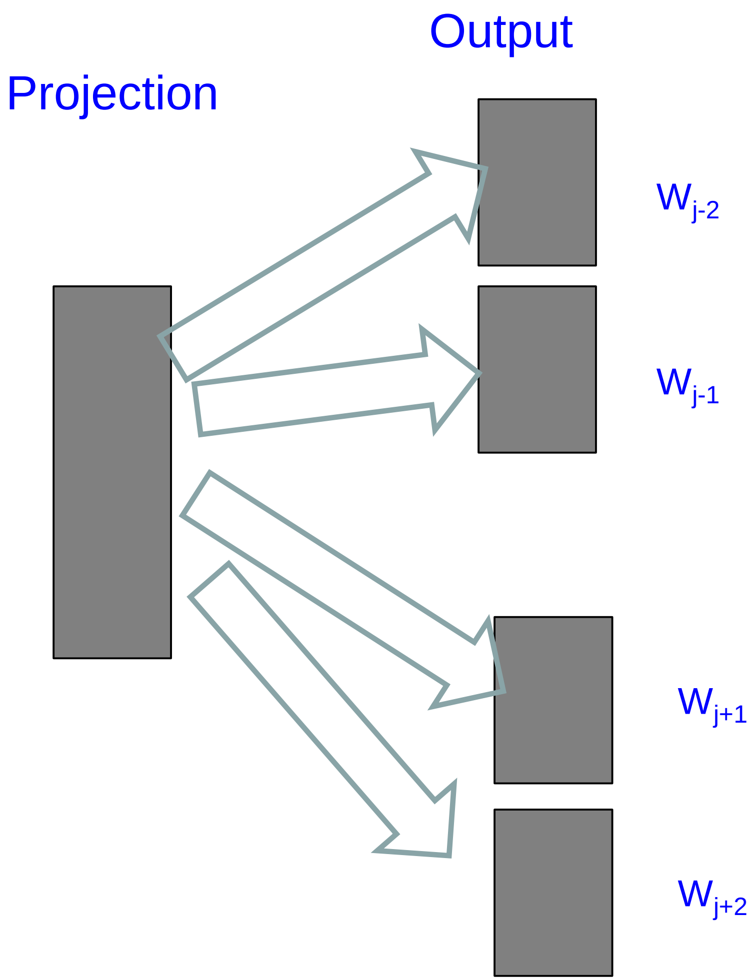
ate
drank
slept
⋮

# Input to Projection (shown for one neuron only)




- From each input neuron, a weight vector of dim *d*
- Input vector is of dim *V*, where V is the vocab size
- Input to projection we have a weight matrix *W* which is *V X d*
- Each row gives the weight vector of dim *d* REPRESENTING that word
- E.g., rows for ‘dog’, ‘cat, ‘lamp’, ‘table’ etc.

# Projection to output




- From the whole projection layer a weight vector of dim *d* to each neuron in each compartment, where the compartment represents a context word
- Each fat arrow is a *d X V* matrix

# Capturing word association

# Basic concept: Co-occurrence Matrix

Corpora: I enjoy cricket. I like music. I like deep learning

| | **I** | **enjoy** | **cricket** | **like** | **music** | **deep** | **learning** |
|---|---|---|---|---|---|---|---|
| **I** | - | 1 | 1 | 2 | 1 | 1 | 1 |
| **enjoy** | 1 | - | 1 | 0 | 0 | 0 | 0 |
| **cricket** | 1 | 1 | - | 0 | 0 | 0 | 0 |
| **like** | 2 | 0 | 0 | - | 1 | 1 | 1 |
| **music** | 1 | 0 | 0 | 1 | - | 0 | 0 |
| **deep** | 1 | 0 | 0 | 1 | 0 | - | 1 |
| **learning** | 1 | 0 | 0 | 1 | 0 | 1 | - |

# Co-occurence Matrix

Fundamental to NLP

Also called **Lexical Semantic Association (LSA)**

Very sparse, many 0s in each row

Apply Principal Component Analysis (PCA) or Singular Value Decomposition (SVD)

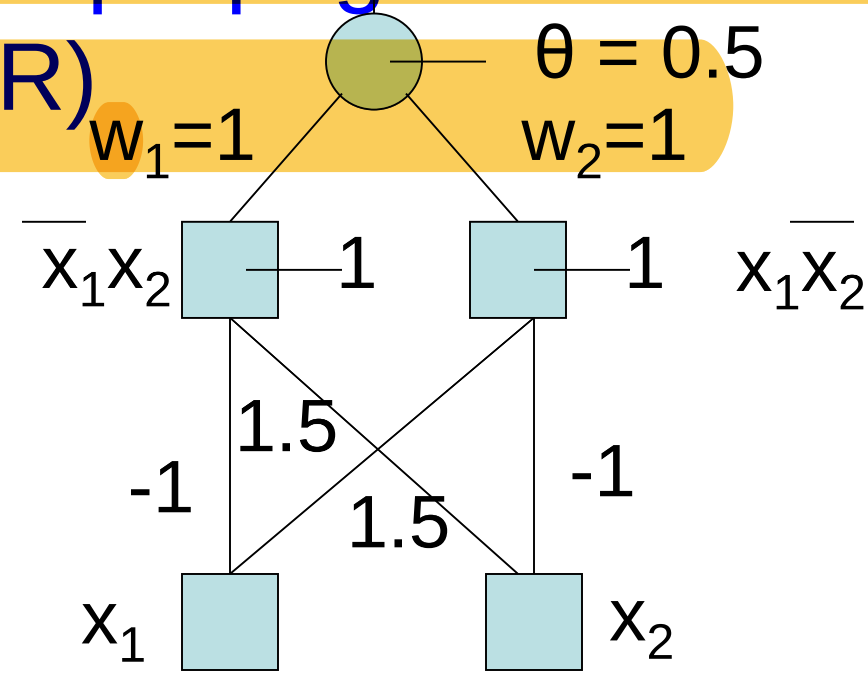
Do Dimensionality Reduction; merge columns with high internal affinity (e.g., *cricket* and *bat*)

Compression achieves better semantics capture

# Important concepts associated with FFNN-BP

# How does BP work?

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




$\bar{x}_1 x_2 = y_1$

$x_1 \bar{x}_2 = y_2$

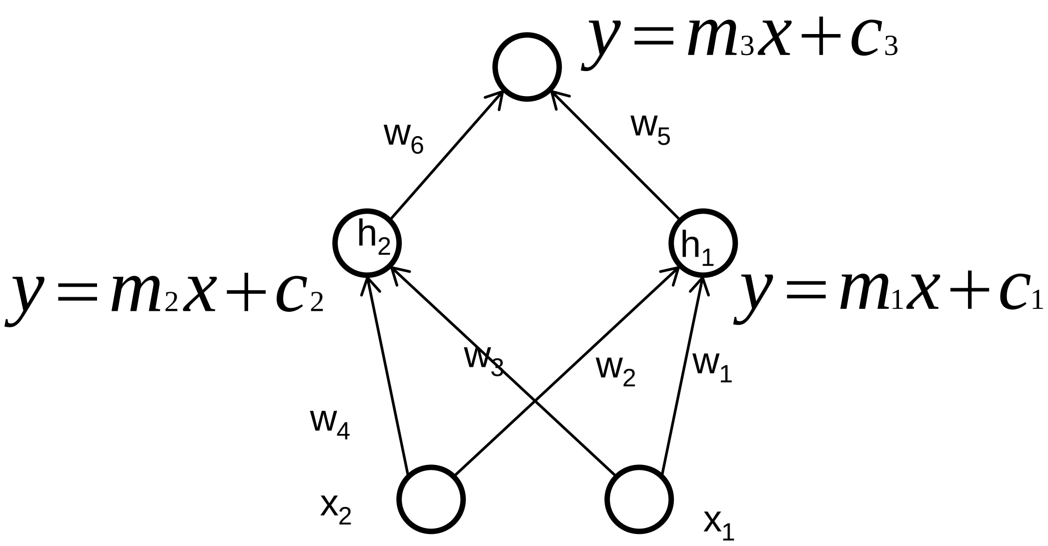
| $y_1$ | $y_2$ | $\geq 1$? |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Take $\geq \theta$ as the threshold

## Work it out !

1) In the XOR network, if the activation function of the hidden layer neurons is changed from sigmoid to the ReLU function how will the weight update rule change for minimizing the ‘total sum-squared error’ of the network?
2) Suppose we have two neurons each in both the hidden and the output layer. Softmax is used at the output. Find out the weight update expressions for the following two cases:
   a) The hidden layer uses ReLU activation.
   b) The hidden layer uses sigmoid activation.

# Can Linear Neurons Work?




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

$$h_2 = m_2(w_3 x_1 + w_4 x_2) + c_2$$

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

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

**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: $m.(w_1 + w_2) - m\ .\theta + c < 0.1$

These equations are inconsistent. Because when we add these inequalities after adjusting for sign, we get 0>1.6!

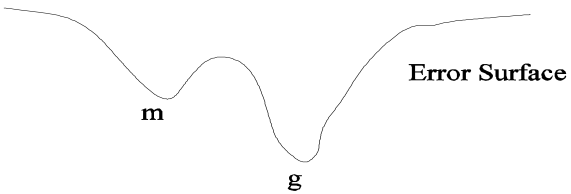
Hence X-OR can't be computed.

**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

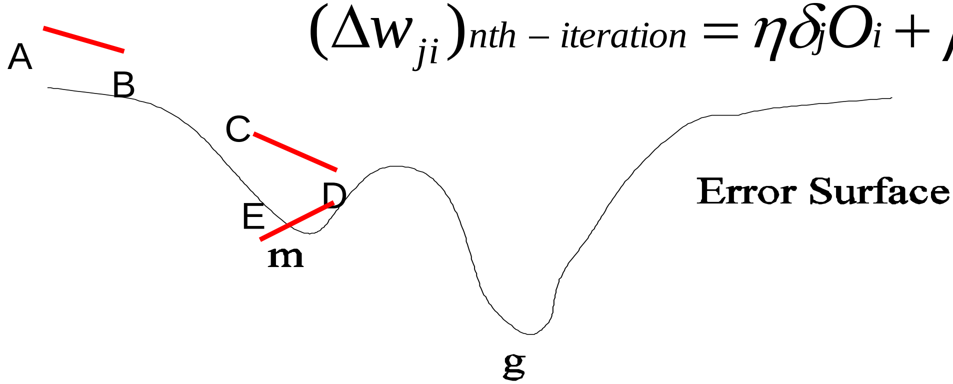
Figure- Getting Stuck in local minimum

# Momentum factor

1. Introduce momentum factor.

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

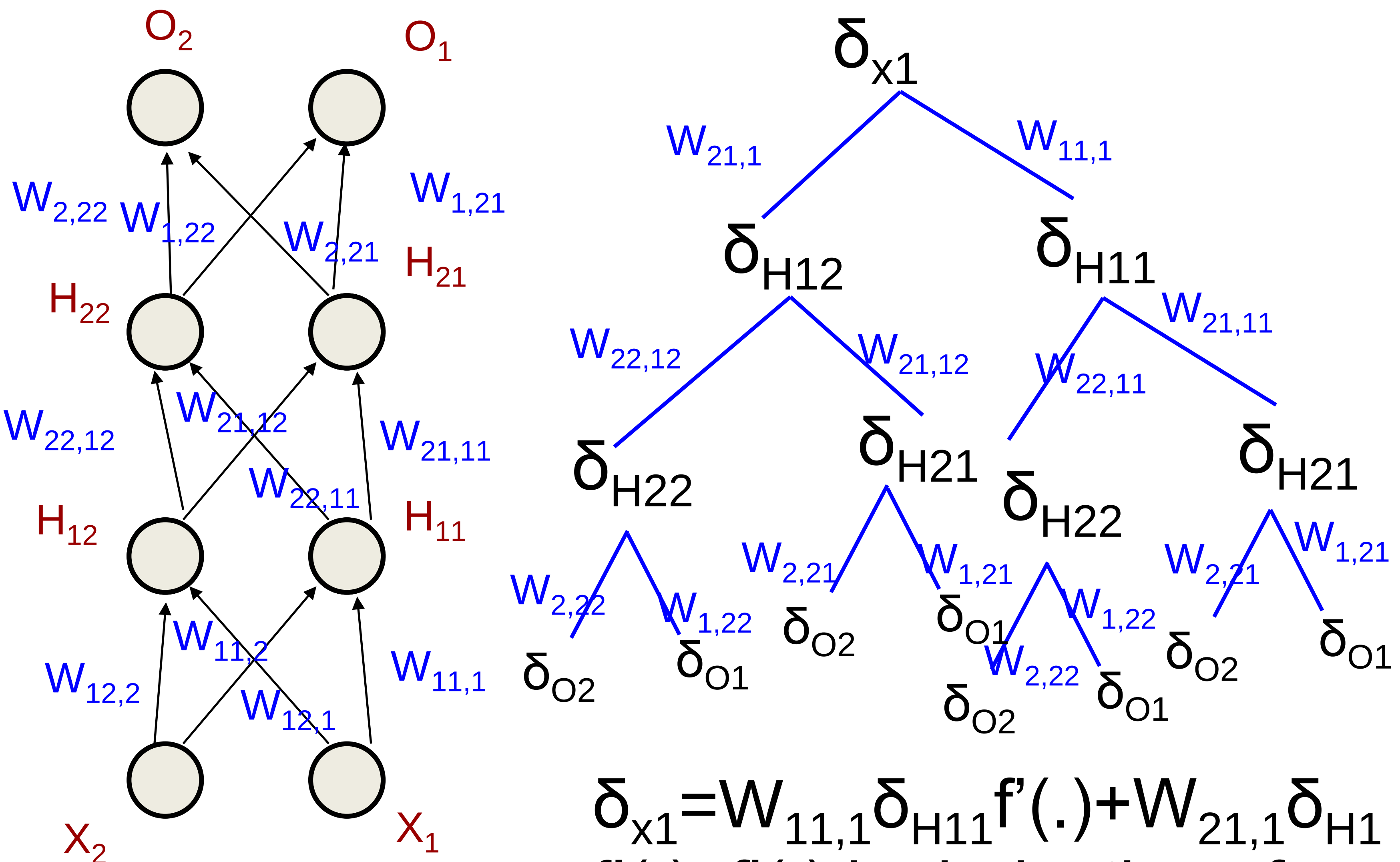
$$(\Delta w_{ji})_{nth-iteration} = \eta \delta_j O_i + \beta (\Delta w_{ji})_{(n-1)th-iteration}$$




**m- local minima, g- global minima**

**Figure- Getting Stuck in local minimum**

# Vanishing/Exploding Gradient



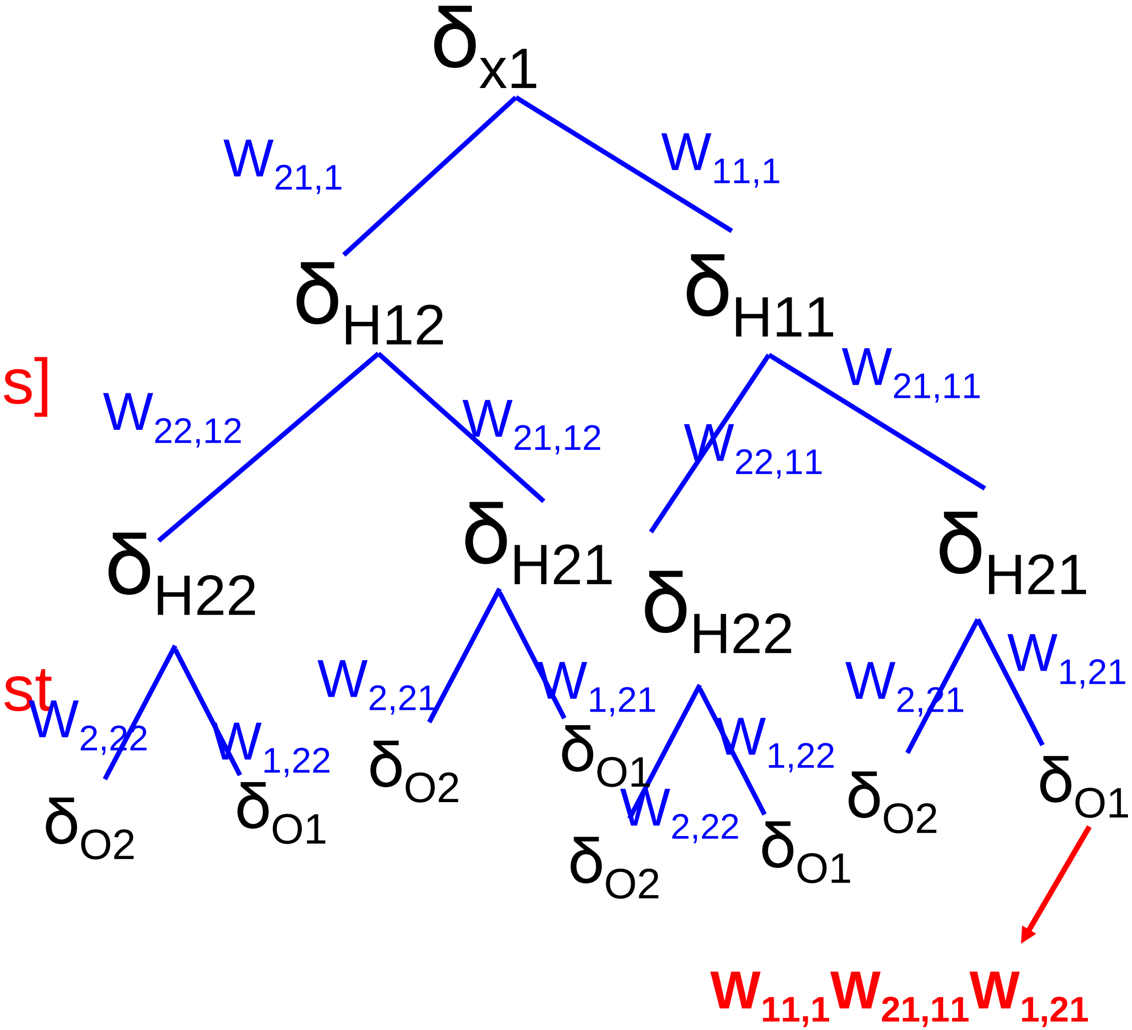

$\delta_{x1}=W_{11,1}\delta_{H11}f'(.)+W_{21,1}\delta_{H1}$ $f'(.)$, $f'(.)$ is derivative of sigmoid

# Vanishing/Exploding Gradient

$\delta_{x1}=W_{11,1}\delta_{H11}f'(.)+W_{21,1}\delta_{H12}f'(.)$ [2 terms]

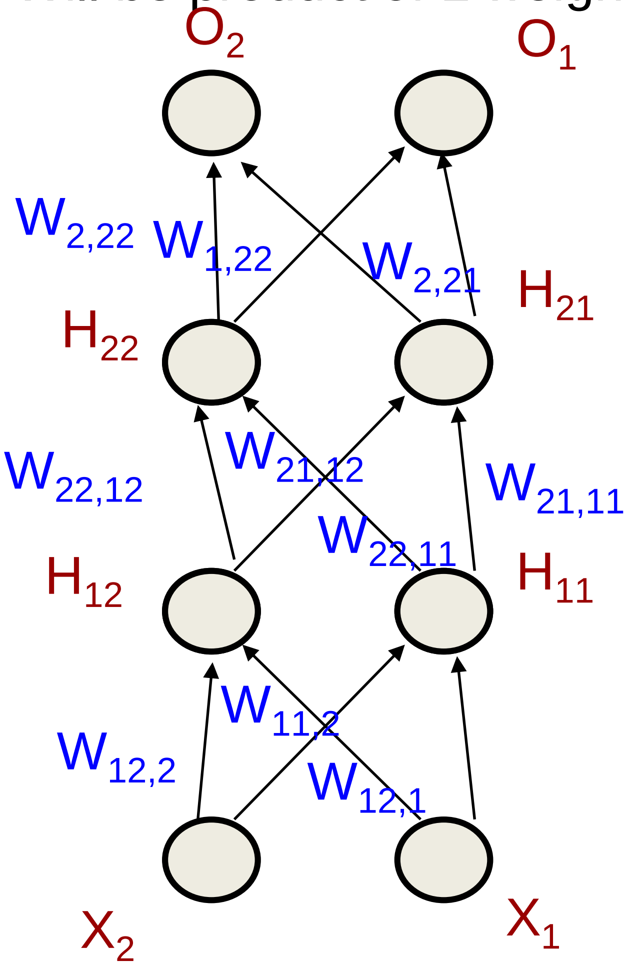
$=W_{11,1}(W_{21,11}\delta_{H21}f'(.)+W_{22,11}\delta_{H22}f'(.))f'(.)+W_{21,1}(W_{21,12}\delta_{H21}f'(.)+W_{22,12}\delta_{H22}f'(.))\ f'(.)$ [4 terms]

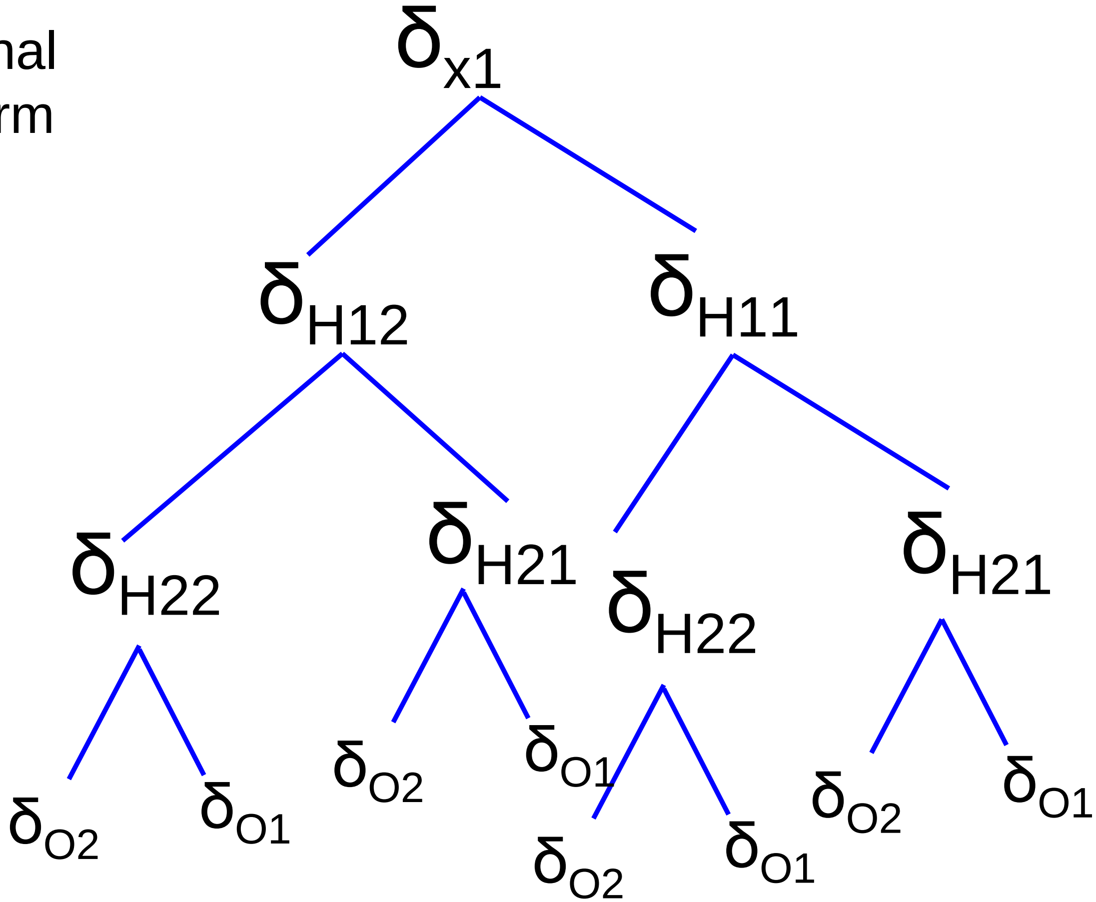
= (4 terms with $\delta_{o1}$) + (4 terms with $\delta_{o2}$; one term shown for the leftmost leaf's weight); also each term has product of derivatives



# 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 $\delta_{x1}$. Also each term
Will be product of L weights

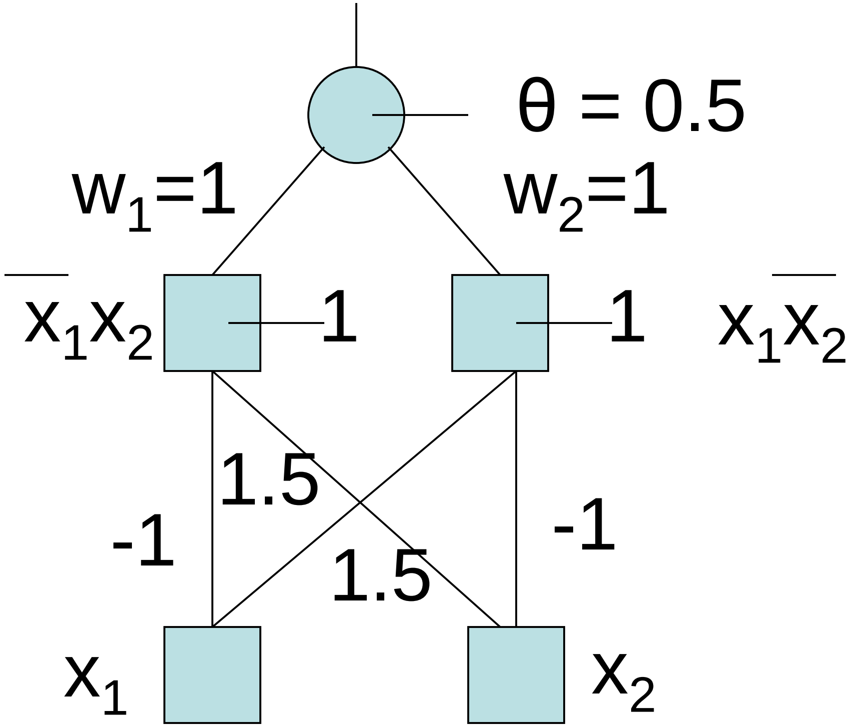



Each term also gets multiplied with
product of derivatives of sigmoid L times.
These products can vanish or explode.

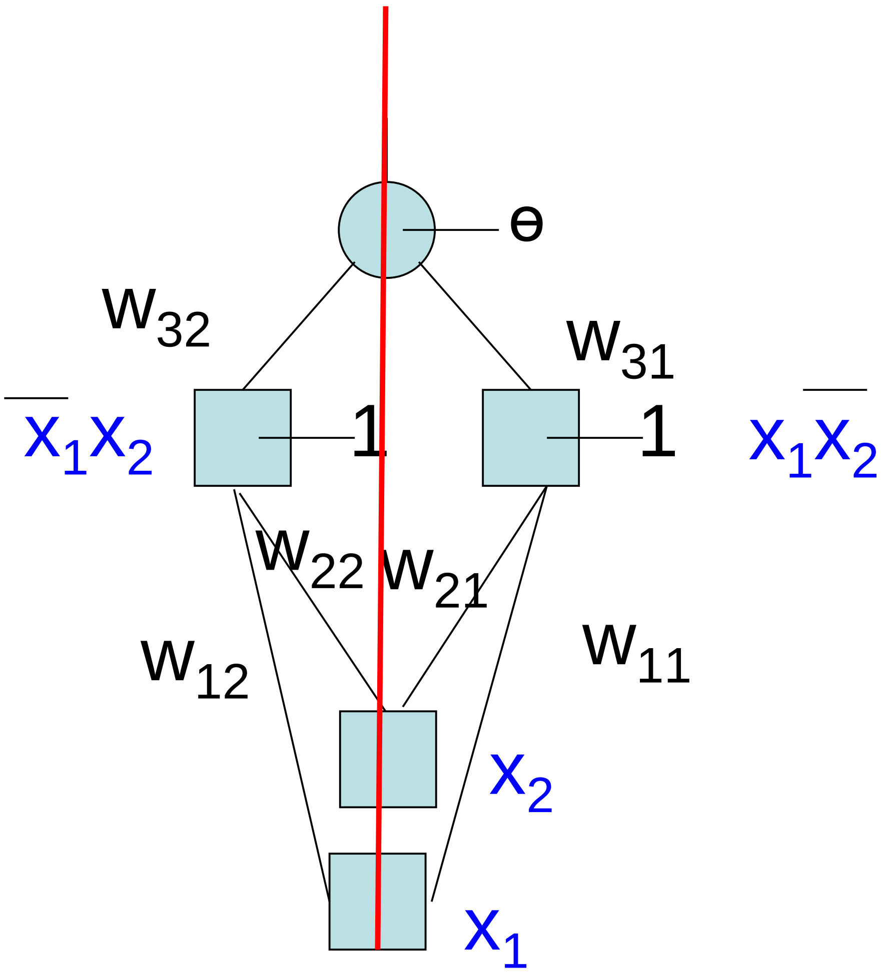
# Symmetry breaking

- 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



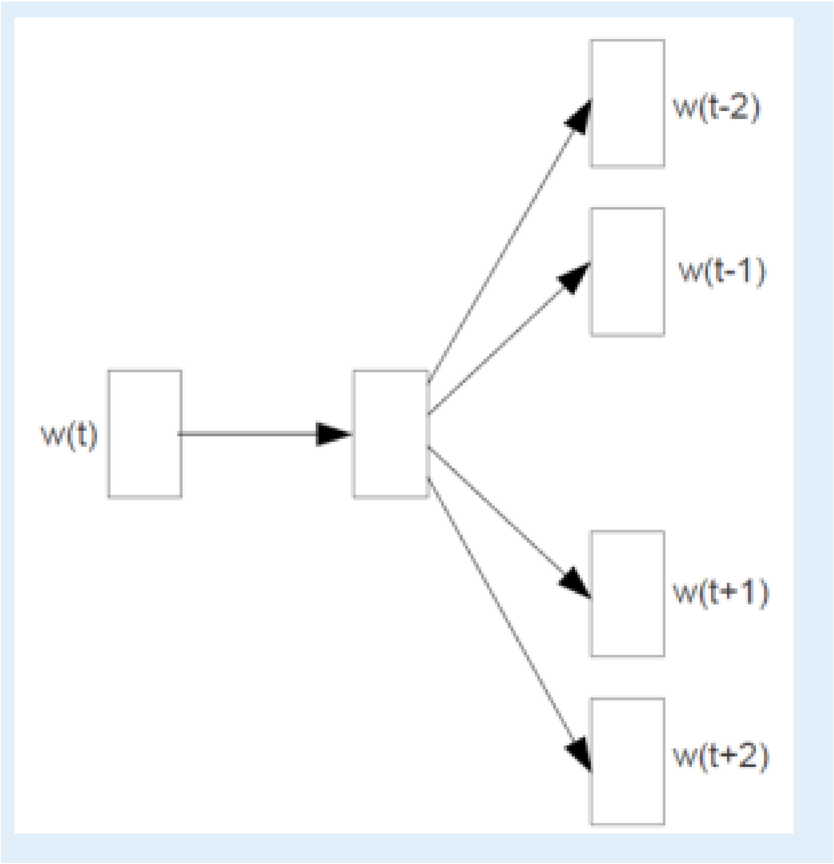

***Symmetry About The red Line should Be broken***

# Linguistic foundation of word representation by vectors

# “Linguistics is the eye”: Harris Distributional Hypothesis

- Words with similar distributional properties have similar meanings. (Harris 1970)
- 1950s: Firth- “A word is known by the company its keeps”
- Model **differences** in meaning rather than the proper meaning itself

# "Computation is the body": Skip gram- predict context from word




## For CBOW:

Just reverse the Input-Ouput

# Dog – Cat - Lamp

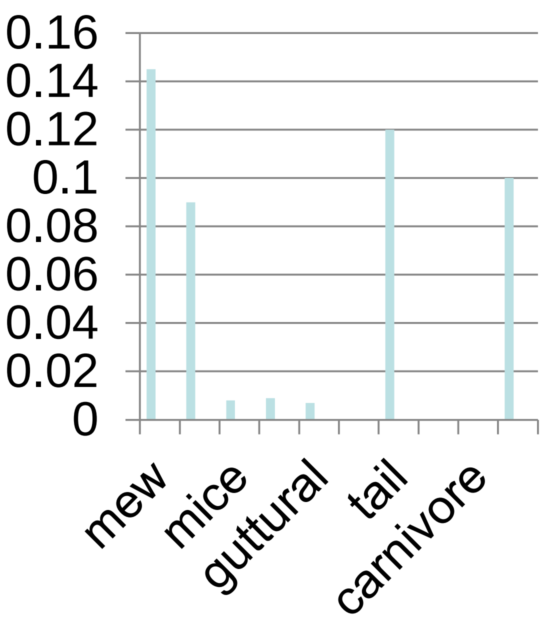
{bark, police, thief, vigilance, faithful, friend, animal, milk, carnivore)

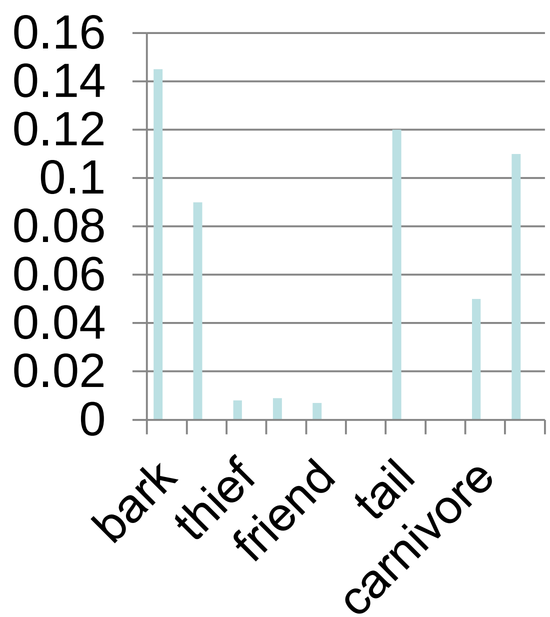
{mew, comfort, mice, furry, guttural, purr, carnivore, milk}

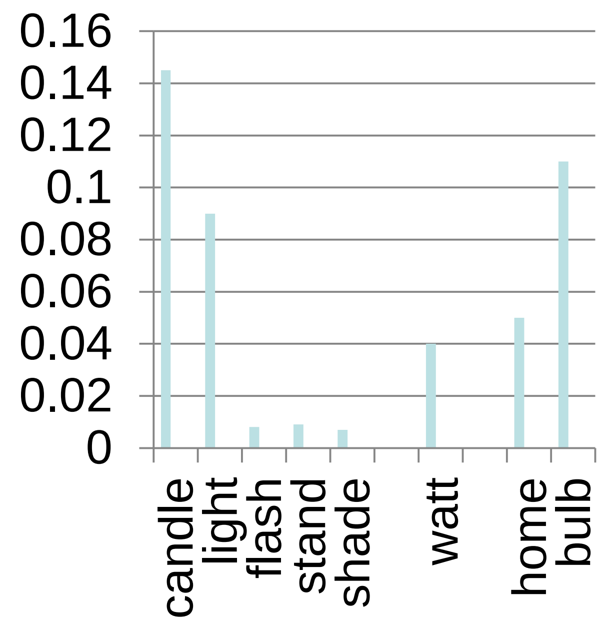
{candle, light, flash, stand, shade, Halogen}

# Probability distributions of context words
# CE(dog, lamp) > CE(dog, cat)





# Test of representation

- **Similarity**
  - ‘Dog’ more similar to ‘Cat’ than ‘Lamp’, because
  - Input- vector(‘dog’), output- vectors of associated words
  - More similar to output from vector(‘cat’) than from vector(‘lamp’)

# “Linguistics is the eye, Computation is the body”

The encode-decoder deep learning network is nothing but

Important.

the ***implementation*** of

Harris’s Distributional Hypothesis

# Fine point in Harris Distributional Hypothesis

- Words with similar distributional properties have similar meanings. (Harris 1970)
- Harris does mentions that distributional approaches can model differences in meaning rather than the proper meaning itself

# Learning objective (skip gram)

In skip gram we have different words.

1 2 3 4 5

and each word is taken into context.

$$J'(\theta) = \frac{1}{T}\prod_{t=1}^{T}\prod_{\substack{-m\le j\le m \\ j\ne 0}} p(w_{t+j} \mid w_t;\theta)$$

$$J(\theta) = -\frac{1}{T}\prod_{t=1}^{T}\prod_{\substack{-m\le j\le m \\ j\ne 0}} p(w_{t+j} \mid w_t;\theta)$$

$$Minimize \quad L = -\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\ne 0}} \log[p(w_{t+j} \mid w_t;\theta)]$$

# Modelling *P(context word|input word)* *(1/2)*

- We want, say, *P('bark'|'dog')*
- Take the weight vector **FROM** 'dog' neuron **TO** projection layer (call this $u_{dog}$)
- Take the weight vector **TO** 'bark' neuron **FROM** projection layer (call this $v_{bark}$)
- When initialized $u_{dog}$ and $v_{bark}$ give the initial estimates of word vectors of 'dog' and 'bark'
- The weights and therefore the word vectors get fixed by back propagation

# Modelling *P(context word|input word)* *(2/2)*

- To model the probability, first compute dot product of $u_{dog}$ and $v_{bark}$
- Exponentiate the dot product
- Take softmax over all dot products over the whole vocabulary

$$P('bark'|'dog') = \frac{\exp(u_{dog}^T v_{bark})}{\sum_{v_k \varepsilon Vocabulary} \exp(u_{dog}^T v_k)}$$

# Exercise

- Why cannot we model *P('bark'|'dog')* as the ratio of counts of <bark, dog> and <dog> in the corpus?
- Why this way of modelling probability through dot product of weight vectors of input and output words, exponentiation and soft-maxing works?

# Working out a simple case of word2vec

# Example (1/3)

- 4 words: *heavy, light, rain, shower*
    - *Heavy:* $U_0$ $<0,0,0,1>$
    - *light:* $U_1$: $<0,0,1,0>$
    - *rain:* $U_2$: $<0,1,0,0>$
    - *shower:* $U_3$: $<1,0,0,0>$
- We want to predict as follows:
    - *Heavy* → *rain*
    - *Light* → *shower*

# Note

- Any bigram is theoretically possible, but actual probability differs

- E.g., heavy-heavy, heavy-light are possible, but unlikely to occur
- Language imposes constraints on what bigrams are possible
- Domain and corpus impose further restriction

# Example (2/3)

- Input-Output
  - *Heavy: $U_0$ <0,0,0,1>, light: $U_1$: <0,0,1,0>, rain: $U_2$: <0,1,0,0>, shower: $U_3$: <1,0,0,0>*

  - *Heavy: $V_0$ <0,0,0,1>, light: $V_1$: <0,0,1,0>, rain: $V_2$: <0,1,0,0>, shower: $V_3$: <1,0,0,0>*

# Example (3/3)

- *heavy → rain*
  - *heavy: $U_0$ <0,0,0,1>*

    *→*

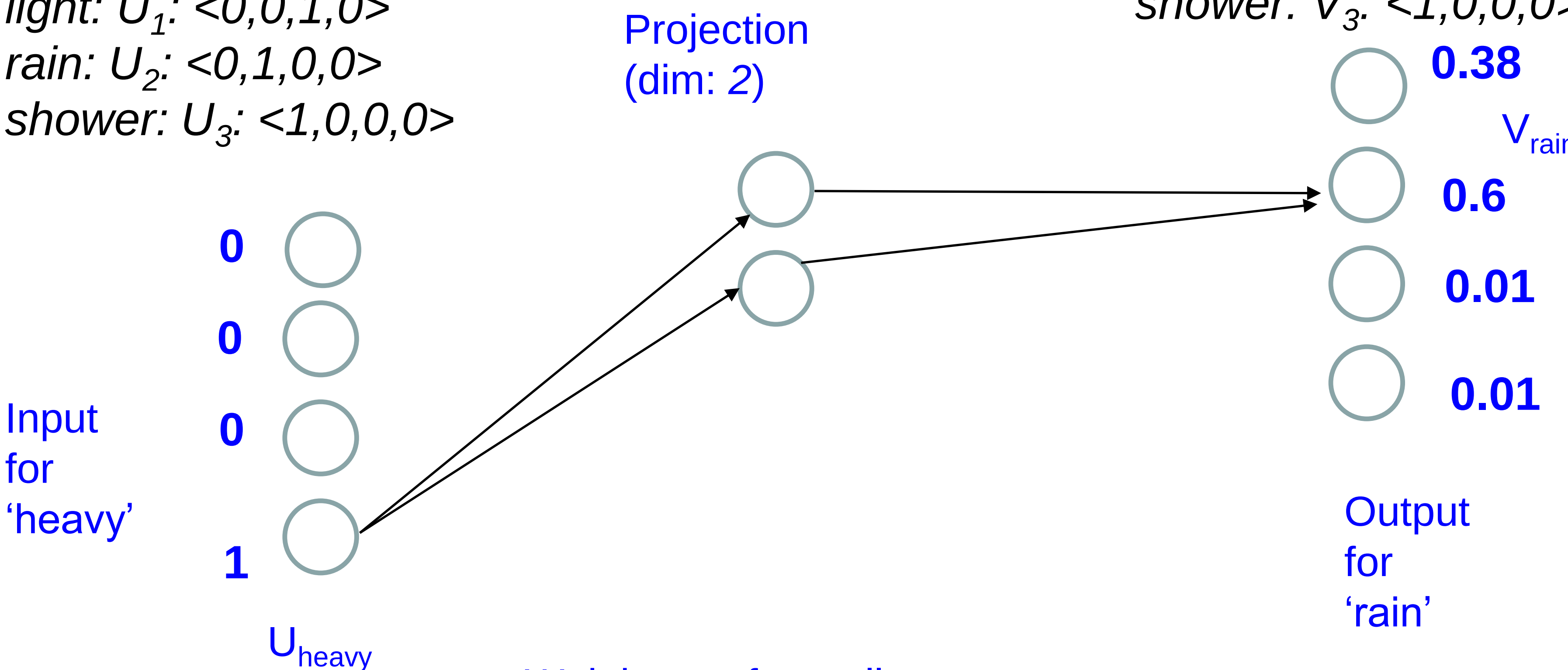
  - *rain: $V_2$: <0,1,0,0>*

- *light → shower*
  - *light: $U_1$: <0,0,1,0>, → shower: $V_3$: <1,0,0,0>*

# Word2vec n/w

*Heavy: $U_0$ <0,0,0,1>*
*light: $U_1$: <0,0,1,0>*
*rain: $U_2$: <0,1,0,0>*
*shower: $U_3$: <1,0,0,0>*

*Heavy: $V_0$ <0,0,0,1>*
*light: $V_1$: <0,0,1,0>*
*rain: $V_2$: <0,1,0,0>*
*shower: $V_3$: <1,0,0,0>*




Weights go from all neurons to
all neurons in the next layer; shown
For only one input and output

# Chain of thinking

- *P(rain|heavy)* should be the highest
- So the output from V2 should be the highest because of softmax
- This way of converting an English statement into probability in insightful