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

Word Embedding

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

Week 6 of 5Feb24

1-slide recap

- Linguistic foundation and derivation of WE
- Objective to be maximized- probability of context words (1-hot embeddings)
- Discriminative vs. Generative modeling- sentiment analysis example
  - Approach1: $C^*=argmax_C[P(C|S)];$ S is the sentence and C is the sentiment class
  - Approach2: $C^*=argmax_C[P(C)P(S|C)];$ apply Bayes theorem
  - Approach1 discriminative; features like adjectives give rise to the classification (e.g., 'wonderful'$\rightarrow$positive sentiment)
  - Approach2 generative; class C 'generates' the features like adjectives (positive sentiment$\rightarrow$'wonderful')

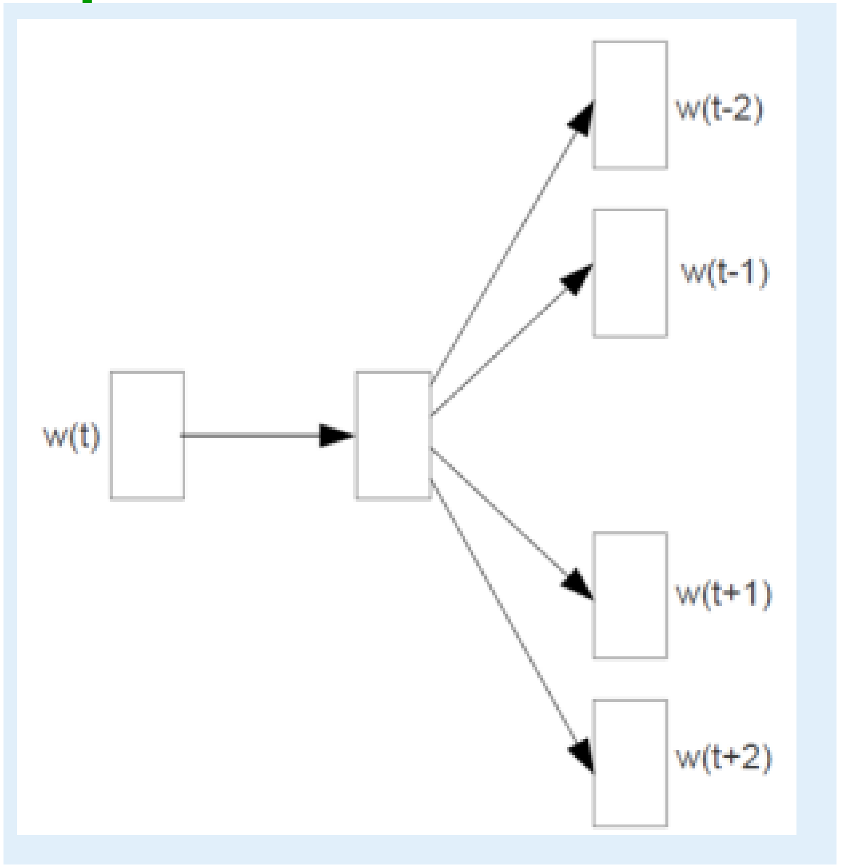
link

Linguistic foundation of word representation by vectors

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-Output

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}

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')

Important.

“Linguistics is the eye, Computation is the body”

The encode-decoder deep learning network is nothing but

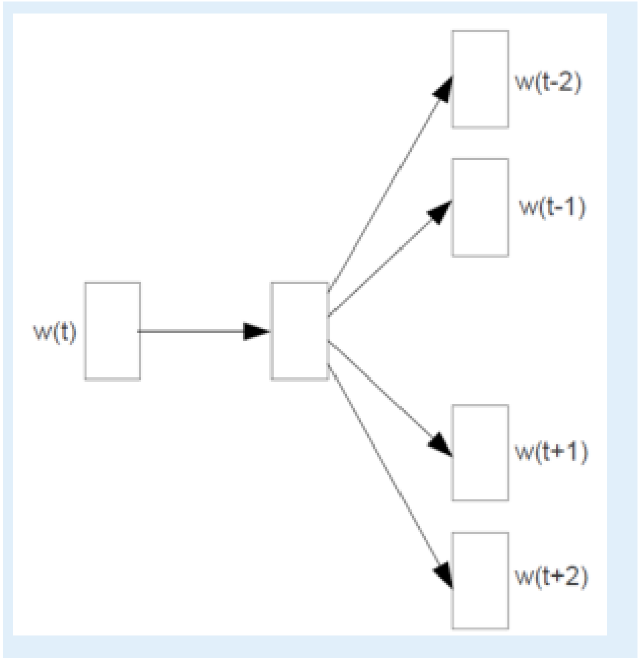
the ***implementation*** of

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)



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

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

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$\rightarrow$ rain*
  - *Light$\rightarrow$ 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 $\rightarrow$ rain*
  - *heavy: U_0 <0,0,0,1>*

    $\rightarrow$

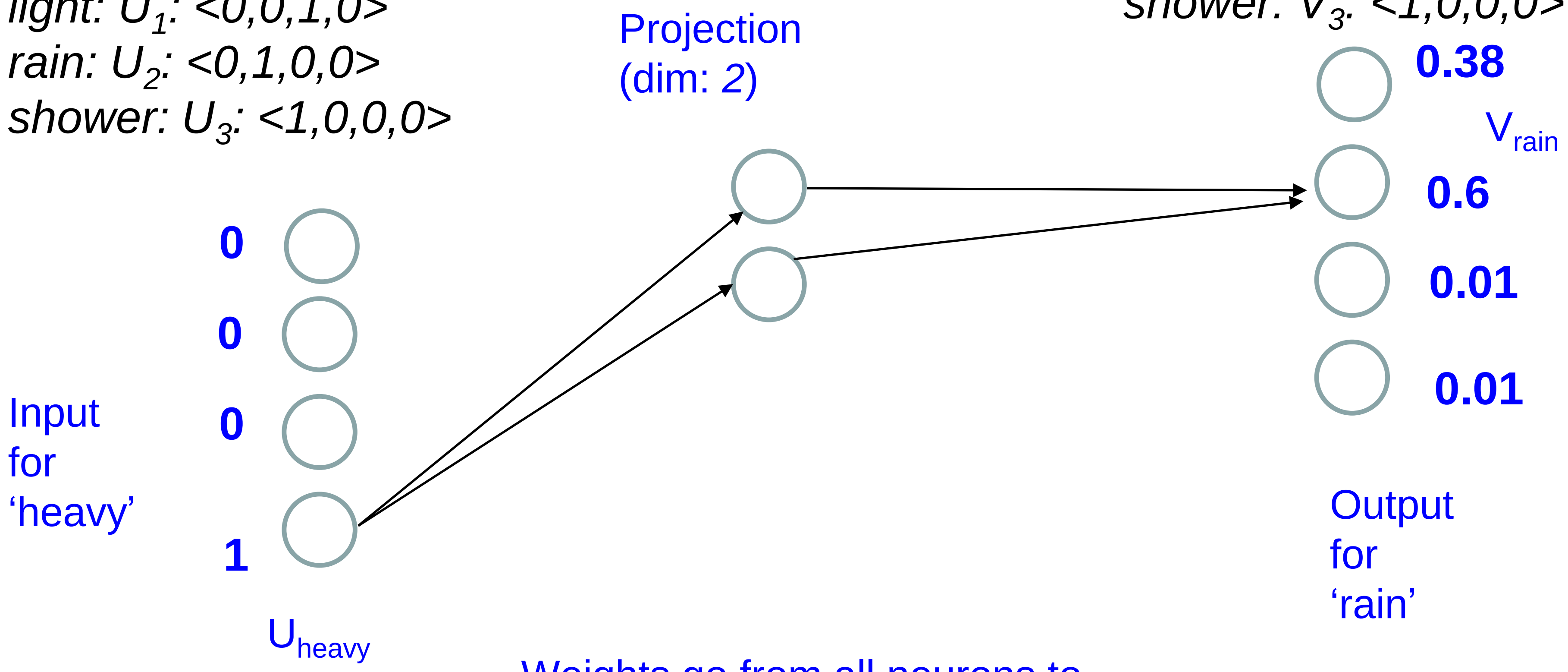
  - *rain: V_2: <0,1,0,0>*

- *light $\rightarrow$ shower*
  - *light: U_1: <0,0,1,0>, $\rightarrow$ 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

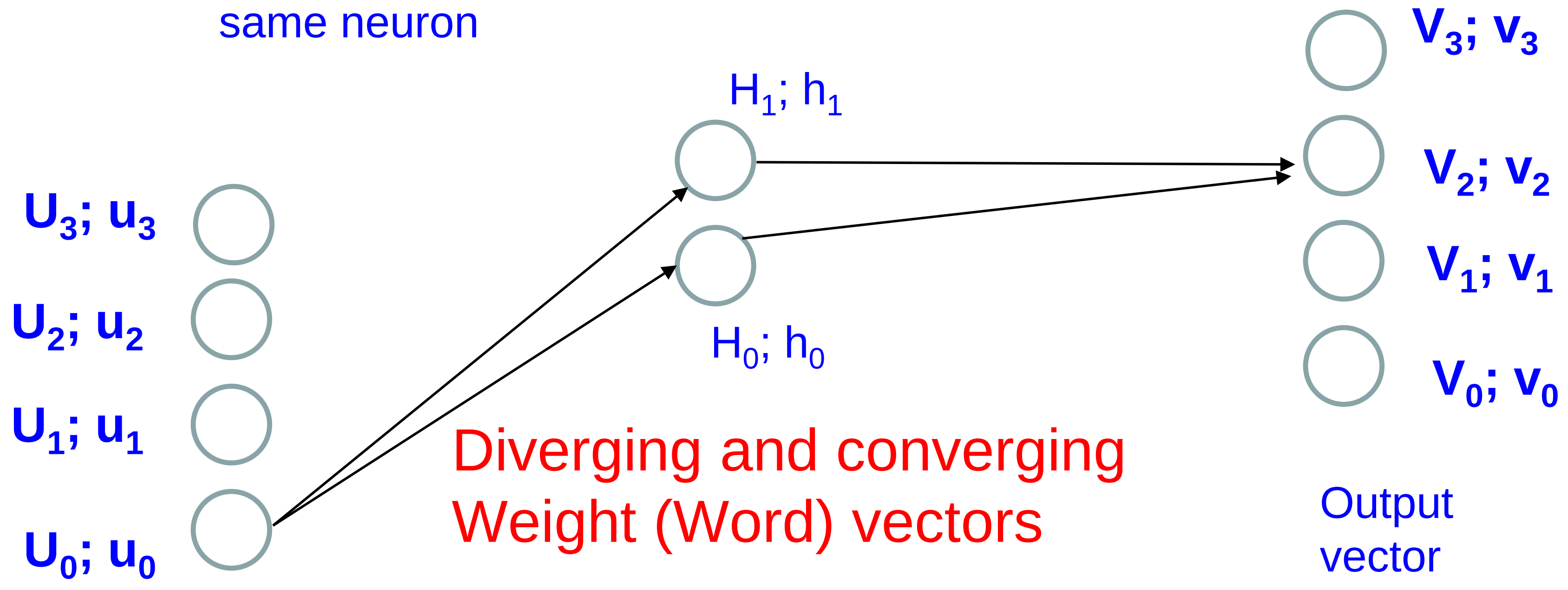
- $P(rain|heavy)$ should be the highest
- So the output from V_2 should be the highest because of softmax
- This way of converting an English statement into probability in insightful

Developing word2vec weight change rule

Illustrated with 4 words only

Word2vec n/w

Convention: Capital letter for NAME of neuron; small letter for output from the same neuron



Diverging and converging Weight (Word) vectors

Output vector V

Input vector U

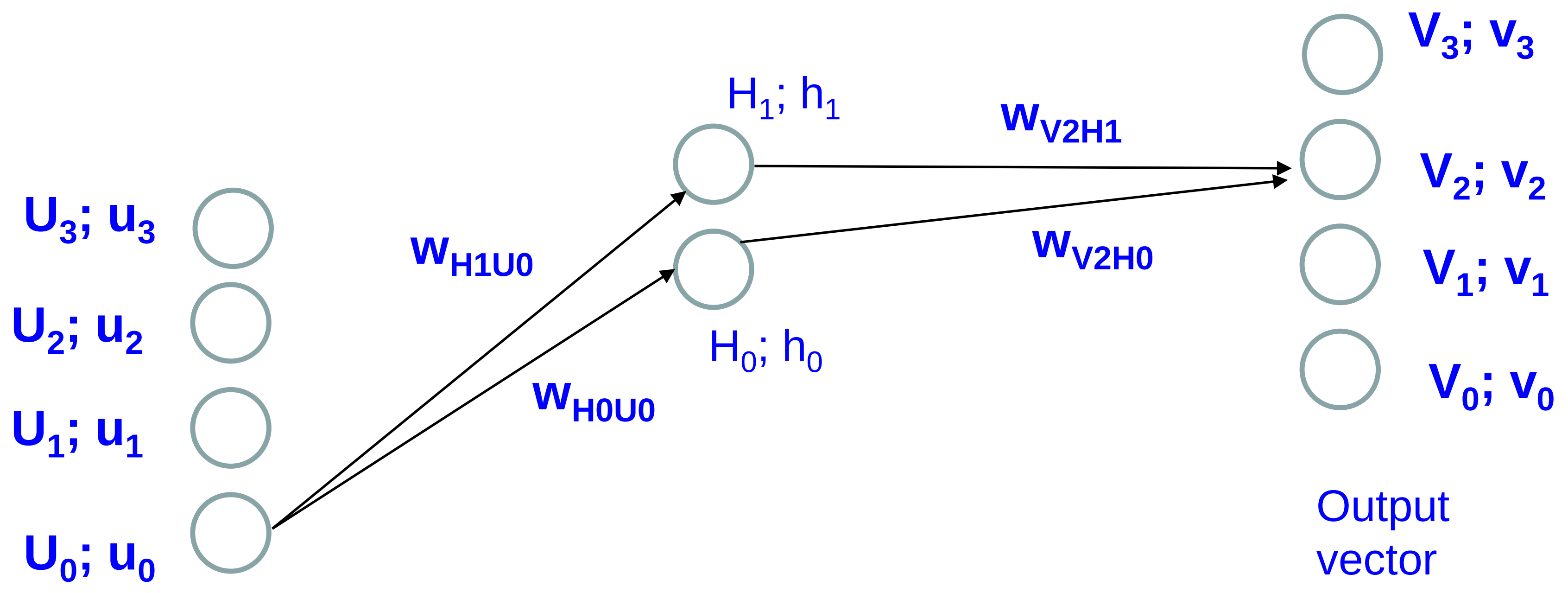
Weights go from all neurons to all neurons in the next layer; shown For only one input and output

Notation Convention

- Weights indicated by small ‘w’
- Index close to ‘w’ is for the destination neuron
- The other index is for the source neuron

Word2vec n/w

Capital letter for NAME of neuron; small letter for output from the same neuron



Output vector V

Input vector U

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

More notation

- Net input to hidden and output layer neurons play an important role in BP
- Net input to hidden layer neurons: net_{H0} and net_{H1}
- Net input to output layer neurons: net_{V0}, net_{V1}, net_{V2}, net_{V3}

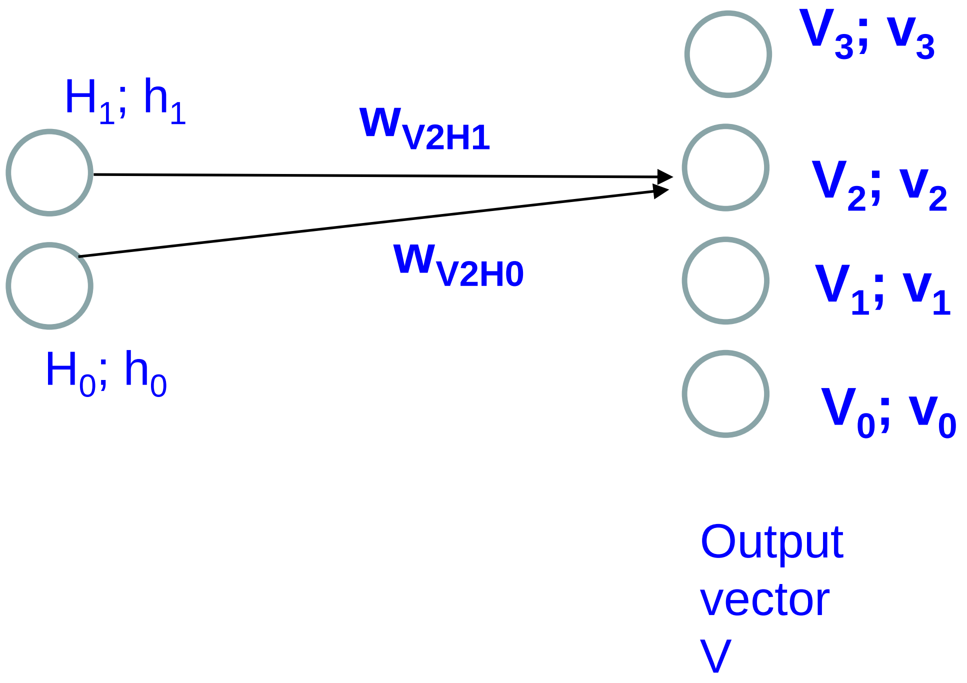
Outputs at the outermost layer

$$v_0 = \frac{e^{net_{V_0}}}{e^{net_{V_0}} + e^{net_{V_1}} + e^{net_{V_2}} + e^{net_{V_3}}}$$

$$v_1 = \frac{e^{net_{V_1}}}{e^{net_{V_0}} + e^{net_{V_1}} + e^{net_{V_2}} + e^{net_{V_3}}}$$

$$v_2 = \frac{e^{net_{V_2}}}{e^{net_{V_0}} + e^{net_{V_1}} + e^{net_{V_2}} + e^{net_{V_3}}}$$

$$v_3 = \frac{e^{net_{V_3}}}{e^{net_{V_0}} + e^{net_{V_1}} + e^{net_{V_2}} + e^{net_{V_3}}}$$



Note

- No non-linearity in the hidden layer
- Why?
- Hidden layer should do ONLY dimensionality reduction
- Can be proved: hidden layer with linearity gives the principal components (will discuss of which Matrix)

Why Dimensionality Reduction?

- The vectors of words represent their distributional similarity

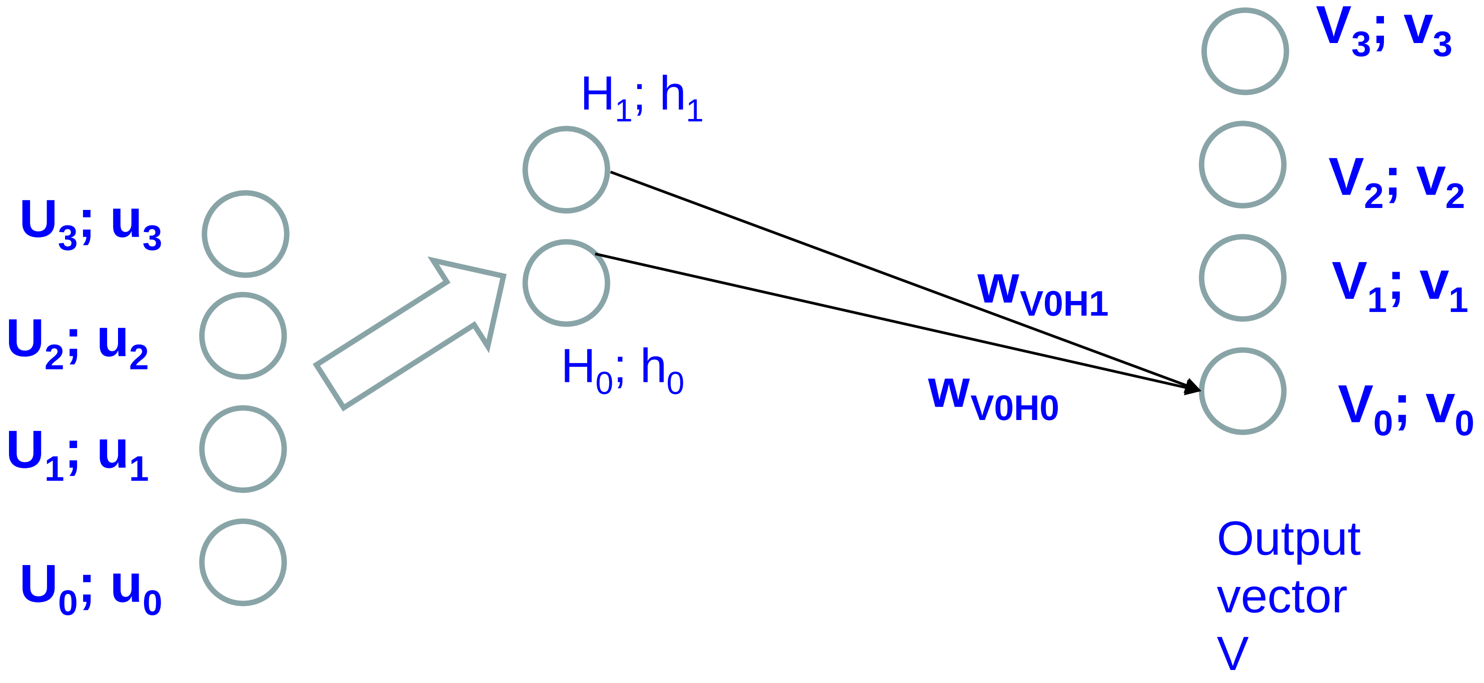
- Dimensionality reduction achieves capturing commonality of these distributional similarities across words

Developing "net$_{vi}$" (1/2)

$$net_{V_0} = w_{V_0H_0}h_0 + w_{V_0H_1}h_1$$

$$h_0 = w_{H_0U_0}u_0 + w_{H_0U_1}u_1 + w_{H_0U_2}u_2 + w_{H_0U_3}u_3$$

$$h_1 = w_{H_1U_0}u_0 + w_{H_1U_1}u_1 + w_{H_1U_2}u_2 + w_{H_1U_3}u_3$$



Developing "net$_{vi}$" (2/2)

- For "heavy", only u_0 is 1, $u_1=u_2=u_3=0$
- So,

$$h_0 = w_{H_0U_0}$$

$$h_1 = w_{H_1U_0}$$

$$net_{v_0} = w_{V_0H_0}w_{H_0U_0} + w_{V_0H_1}w_{H_1U_0}$$

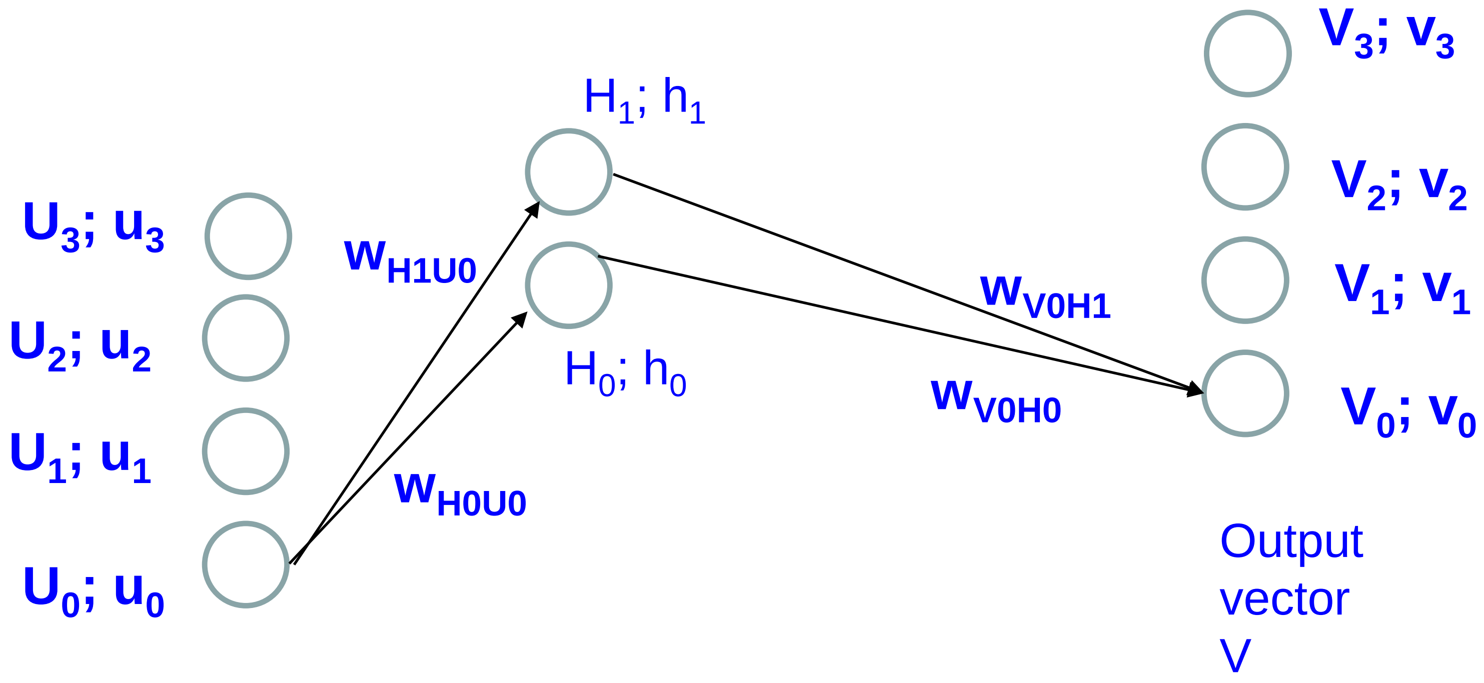
$$= \begin{bmatrix} w_{H_0U_0} & w_{H_1U_0} \end{bmatrix} \begin{bmatrix} w_{V_0H_0} \\ w_{V_0H_1} \end{bmatrix}$$

More Notation

- Weight vector FROM U_0 is called W_{U0} (capital 'W')
- Weight vector INTO V_0 is called W_{V0}
- Slight liberty with notation, but has intuitive advantage

For "heavy" (=U_0), the value of net_{v0}

$$net_{V_0} = W_{U_0} . W_{V_0}^T$$



For “heavy” (=U_0), values of other net_{vi}s

$$net_{V_0} = W_{U_0} . W_{V_0}^T$$

$$net_{V_1} = W_{U_0} . W_{V_1}^T$$

$$net_{V_2} = W_{U_0} . W_{V_2}^T$$

$$net_{V_3} = W_{U_0} . W_{V_3}^T$$

We want to maximize *P('rain'=V₂|'heavy'=U₀)*

- This probability is in terms of softmax.

$$P('rain'=V_2 \,|'\, heavy'=U_0)$$

$$=v_2=\frac{e^{net_{V_2}}}{e^{net_{V_0}}+e^{net_{V_1}}+e^{net_{V_2}}+e^{net_{V_3}}}$$

Equivalent to

- minimize error function

E= -log[P('rain'=V$_2$*|'heavy'=U*$_0$*)]*

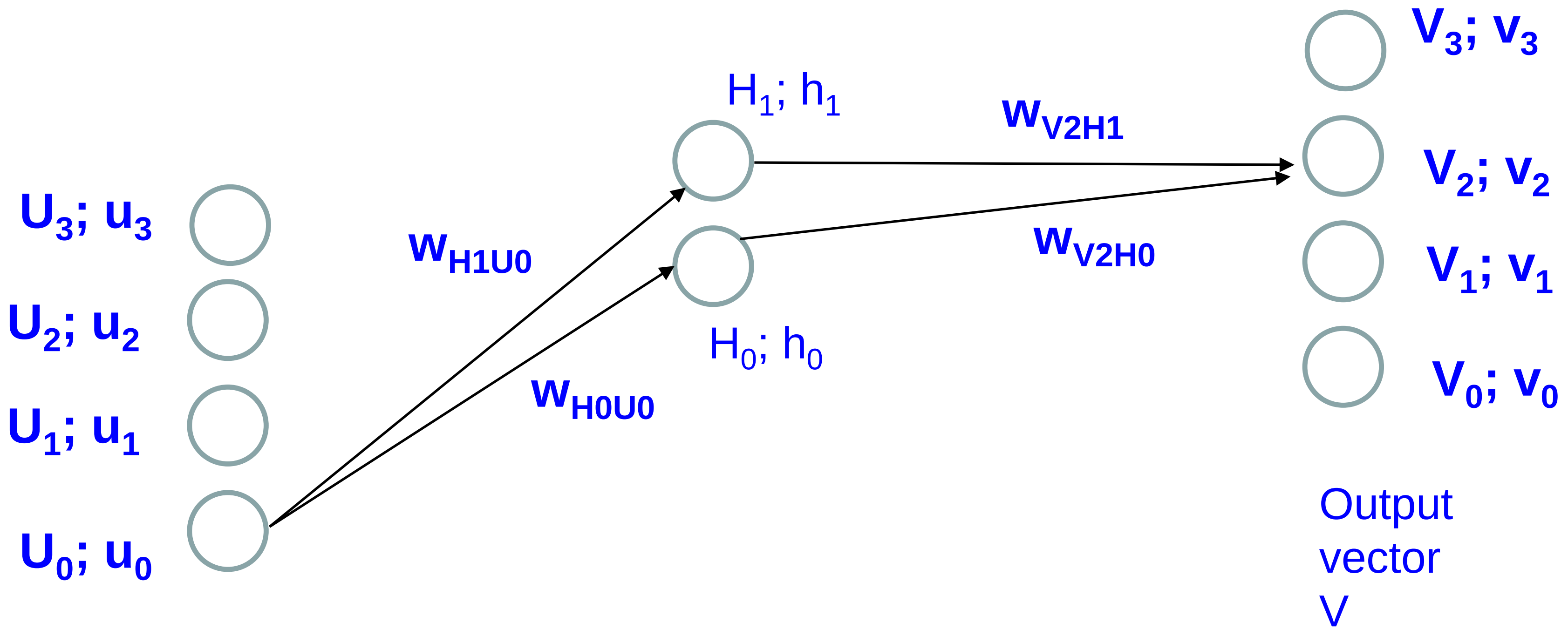
$$-\log[P('rain'=V_2\,|'heavy'=U_0)]$$

$$=-net_{V_2}+\log(e^{net_{V0}}+e^{net_{V1}}+e^{net_{V2}}+e^{net_{V3}})$$

$$=-W_{U_0}W_{V_2}^T+\log(e^{net_{V0}}+e^{net_{V1}}+e^{net_{V2}}+e^{net_{V3}})$$

Word2vec n/w

Capital letter for NAME of neuron; small letter for output from the same neuron



Input vector U

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

Computing Δw_{V2H0}

$$\Delta w_{V_2H_0} = -\eta \frac{\delta E}{\delta w_{V_2H_0}}$$

$$E = -net_{V_2} + \log(e^{net_{V_0}} + e^{net_{V_1}} + e^{net_{V_2}} + e^{net_{V_3}})$$

$$= -W_{U_0} W_{V_2}^T + \log(e^{net_{V_0}} + e^{net_{V_1}} + e^{net_{V_2}} + e^{net_{V_3}})$$

$$W_{U_0} W_{V_2}^T = w_{V_2H_0} w_{H_0U_0} + w_{V_2H_1} w_{H_1U_0}$$

$$\frac{\delta E}{\delta w_{V_2H_0}} = -w_{H_0U_0} + \frac{e^{W_{V_2}.W_{U_0}}}{e^{W_{V_0}.W_{U_0}} + e^{W_{V_1}.W_{U_0}} + e^{W_{V_2}.W_{U_0}} + e^{W_{V_3}.W_{U_0}})} .w_{H_0U_0}$$

$$= -w_{H_0U_0} + v_2 . w_{H_0U_0}$$

$$\Rightarrow \Delta w_{V_2H_0} = \eta(1 - v_2).w_{H_0U_0} = \eta(1 - v_2) o_{H_0}$$

o/p of hidden neuron H_0

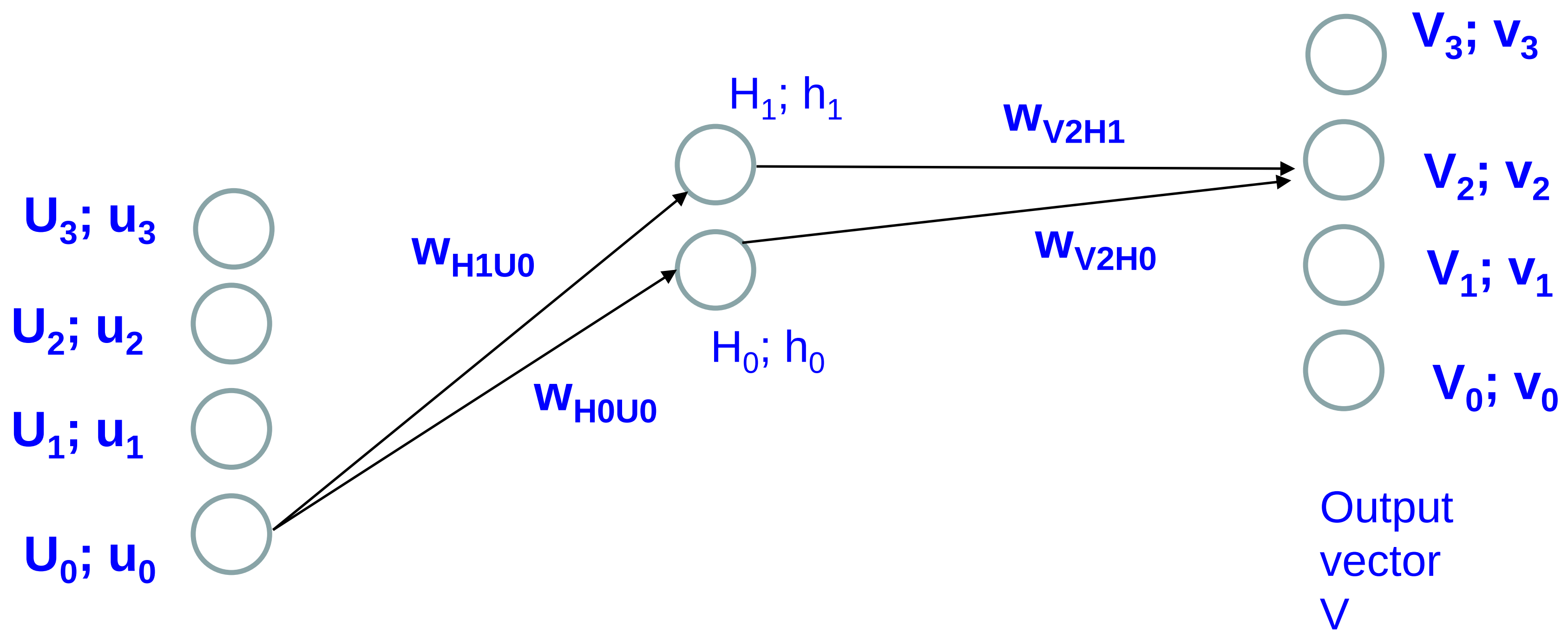
Interpretation of weight change rule for V_2

- If v_2 is close to 1, change in weight too is small

- w_{H0U0} is equal to the input to H_0 (since $u_0=1$) and to its output too, since hidden neurons simply transmit the output.

Word2vec n/w

Capital letter for NAME of neuron; small letter for output from the same neuron



Input vector U

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

Change in other weights to output layer, say, *V_1*, due to input *U_0*

$$\Delta w_{V_1H_0} = -\eta \frac{\delta E}{\delta w_{V_1H_0}}$$

$$E = -net_{V_2} + \log(e^{net_{V0}} + e^{net_{V1}} + e^{net_{V2}} + e^{net_{V3}})$$

$$= -W_{U_0}W_{V_2}^T + \log(e^{net_{V0}} + e^{net_{V1}} + e^{net_{V2}} + e^{net_{V3}})$$

$$W_{U_0}W_{V_2}^T = w_{V_2H_0}w_{H_0U_0} + w_{V_2H_1}w_{H_1U_0}$$

$$\frac{\delta E}{\delta w_{V_1H_0}} = -0 + \frac{e^{W_{V1}.W_{U0}}}{e^{W_{V0}.W_{U0}} + e^{W_{V1}.W_{U0}} + e^{W_{V2}.W_{U0}} + e^{W_{V3}.W_{U0}})}.w_{H_0U_0}$$

$$= v_1.w_{H_0U_0}$$

$$\Rightarrow \Delta w_{V_1H_0} = -\eta v_1 w_{H_0U_0} = -\eta v_1 o_{H_0} = \eta(0 - v_1)o_{H_0}$$

Same weight change rule for any neuron in the o/p layer

learning rate X difference X output of source neuron

Interpretation of weight change rule for V_1

- Assume w_{H0U0} to be positive
- For training $U0 \rightarrow V2$, i.e., ‘heavy’→’rain’, if v_2 is not 1, Δw_{V2H0} is +ve
- For the same input, Δw_{V1H0} is negative
- So the two weight changes are of opposite sign.
- The effect is that while v_2 increases, v_1 decrease for the input U_0, as it should be since we want to increase *P(‘rain’|’heavy’)* and depress all other probabilities

Weight change for input to **hidden layer**, say, w_{H0U0}

$$\Delta w_{H_0U_0} = -\eta \frac{\delta E}{\delta w_{H_0U_0}}$$

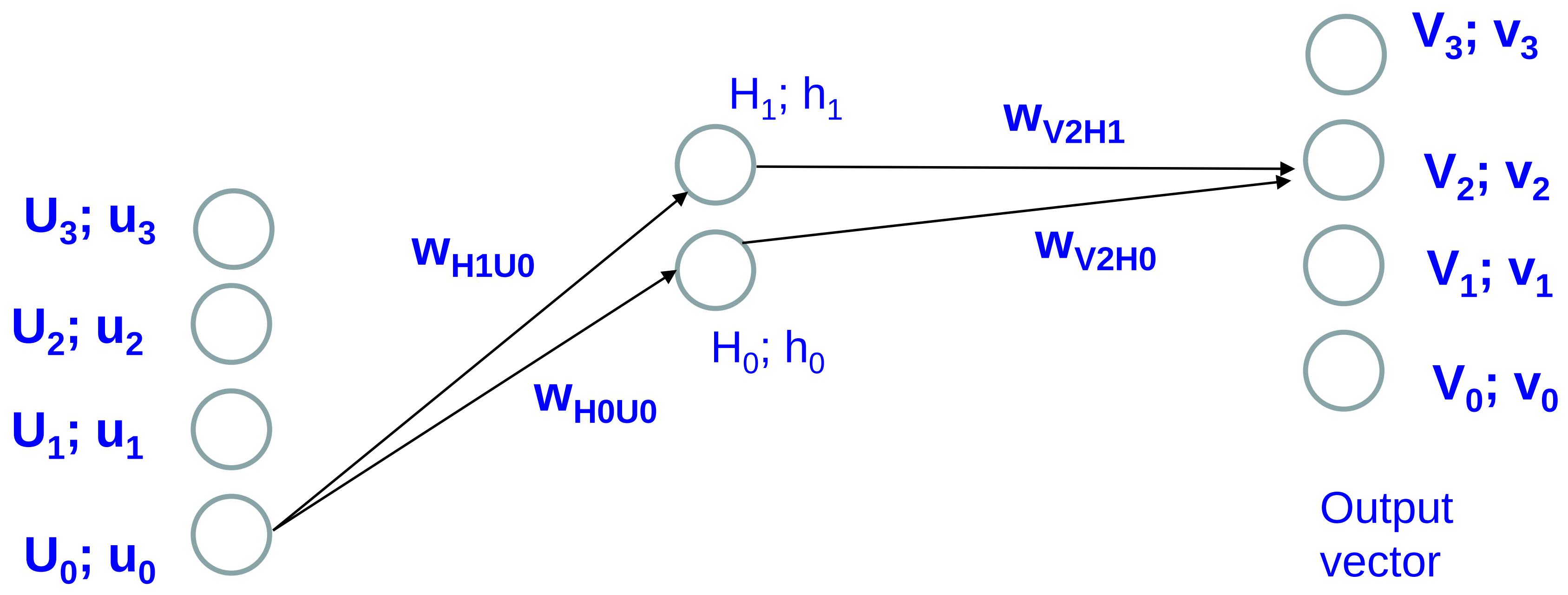
$$E = -net_{V_2} + \log(e^{net_{V0}} + e^{net_{V1}} + e^{net_{V2}} + e^{net_{V3}})$$

$$= -W_{U_0}W_{V_2}^{T} + \log(e^{net_{V0}} + e^{net_{V1}} + e^{net_{V2}} + e^{net_{V3}})$$

$$W_{U_0}W_{V_2}^{T} = w_{V_2H_0}w_{H_0U_0} + w_{V_2H_1}w_{H_1U_0}$$

Word2vec n/w

Capital letter for NAME of neuron; small letter for output from the same neuron



Output vector V

Input vector U

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

Cntd: Weight change for input to hidden layer, say, w_{H0U0}

$$\frac{\delta E}{\delta w_{H_0U_0}}$$

$$= -w_{V_2H_0} + \frac{w_{V_0H_0}e^{W_{V_0}.W_{U_0}} + w_{V_1H_0}e^{W_{V_1}.W_{U_0}} + w_{V_2H_0}e^{W_{V_2}.W_{U_0}} + w_{V_3H_0}e^{W_{V3}.W_{U_0}}}{e^{W_{V_0}.W_{U_0}} + e^{W_{V_1}.W_{U_0}} + e^{W_{V_2}.W_{U_0}} + e^{W_{V_3}.W_{U_0}}}$$

$$= -w_{V_2H_0} + w_{V_0H_0}v_0 + w_{V_1H_0}v_1 + w_{V_2H_0}v_2 + w_{V_3H_0}v_3$$

$$\Rightarrow \Delta w_{H_0U_0} = \eta[(1-v_2)w_{V_2H_0} - w_{V_0H_0}v_0 - w_{V_1H_0}v_1 - w_{V_3H_0}v_3]$$

$$= \eta[(1-v_2)w_{V_2H_0} + (0-v_0)w_{V_0H_0} + (0-v_1)w_{V_1H_0} + (0-v_3)w_{V_3H_0}]$$

$$= \eta[(1-v_2)w_{V_2H_0} + (0-v_0)w_{V_0H_0} + (0-v_1)w_{V_1H_0} + (0-v_3)w_{V_3H_0}].1$$

$$= \eta[(1-v_2)w_{V_2H_0} + (0-v_0)w_{V_0H_0} + (0-v_1)w_{V_1H_0} + (0-v_3)w_{V_3H_0}].u_0$$

Weight change rule for hidden layer neurons

learning rate X backpropagated delta X input from source neuron

Representation Learning

Basics

- What is a good representation? At what granularity: words, n-grams, phrases, sentences
- Sentence is important- (a) *I <u>bank</u> with SBI; (b) I took a stroll on the river <u>bank</u>; (c) this <u>bank</u> sanctions loans quickly*
- Each ‘bank’ should have a different representation
- We have to LEARN these representations

Principle behind representation

- Proverb: “A man is known by the company he keeps”

- Similarly: “A word is known/represented by the company it keeps”

- “Company” $\rightarrow$ Distributional Similarity

Starting point: 1-hot representation

- Arrange the words in lexicographic order
- Define a vector *V* of size |*L*|, where *L* is the lexicon
- For word w_i in the i^{th} position, set the ith bit to 1, all other bits being 0.
- Problem: cosine similarity of ANY pair is 0; wrong picture!!

Representation: to learn or not learn?

- 1-hot representation does not capture many nuances, e.g., semantic similarity
  - But is a good starting point

- Co-occurences also do not fully capture all the facets
  - But is a good starting point

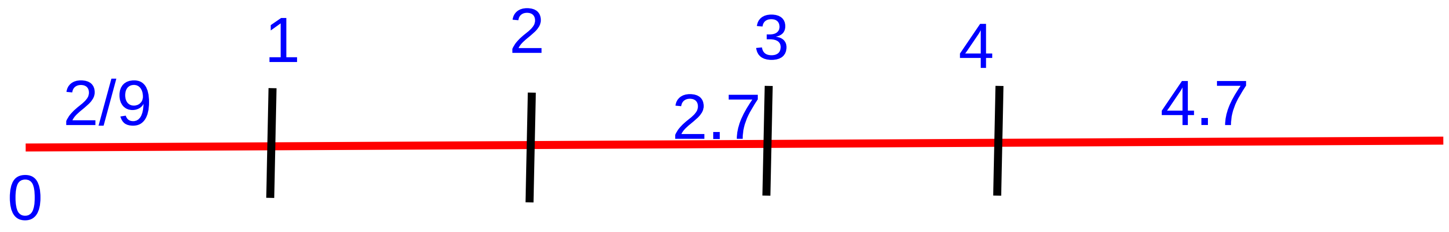
So learn the representation…

- Learning Objective

- ***MAXIMIZE CONTEXT PROBABILITY***

Foundations-1: Embedding

- Way of taking a discrete entity to a continuous space
- E.g., 1, 2, 3, 2.7, 2/9, $22^{1/2}$, … are numerical symbols
- But they are points on the real line
- Natural embedding
- Words' embedding not so intuitive!



Foundations-2: Purpose of Embedding

- Enter geometric space
- Take advantage of “distance measures”- Euclidean distance, Riemannian distance and so on
- “Distance” gives a way of computing similarity

Foundations-3: Similarity and difference

- Recognizing similarity and difference-foundation of intelligence
- Lot of Pattern Recognition is devoted to this task (Duda, Hart, Stork, 2nd Edition, 2000)
- Lot of NLP is based on Text Similarity
- Words, phrases, sentences, paras and so on (verticals)
- Lexical, Syntactic, Semantic, Pragmatic (Horizontal)

Similarity study in MT

English:

This blanket is very soft

Hindi:

yaha kambal bahut naram hai

Bangla:

ei kambal ti khub naram <null>

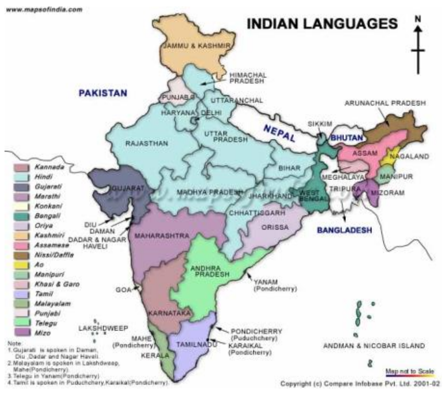
Marathi:

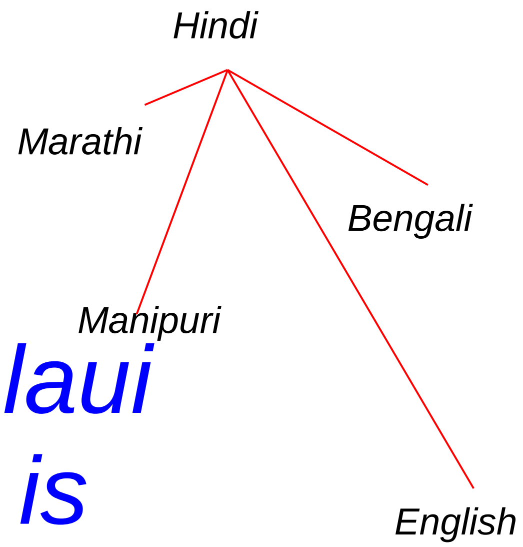
haa kambal khup naram aahe

Manipuri:

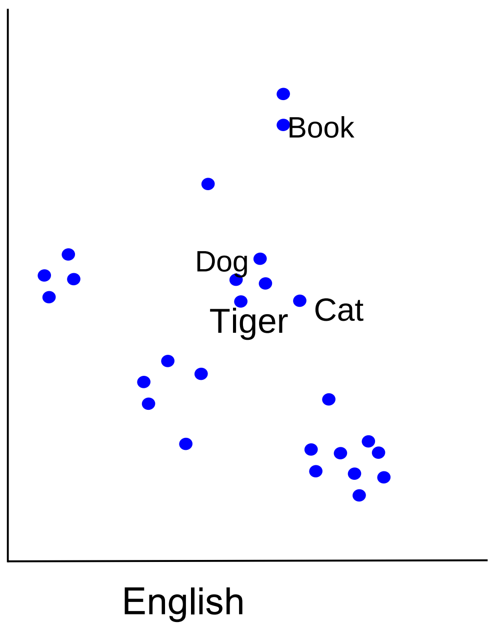
kampor asi mon mon laui

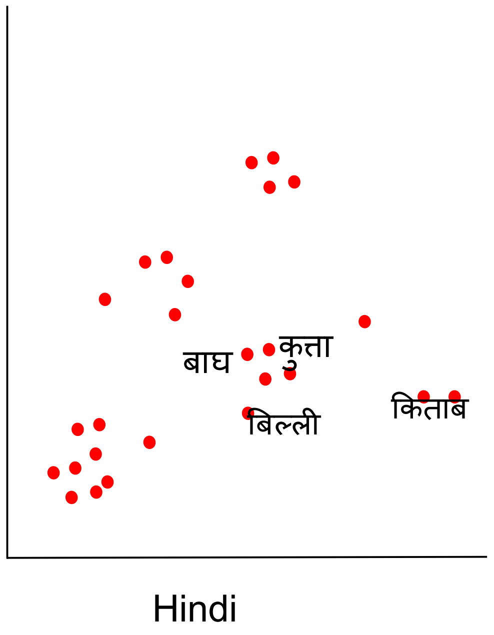
blanket this soft soft is





ISO-Metricity

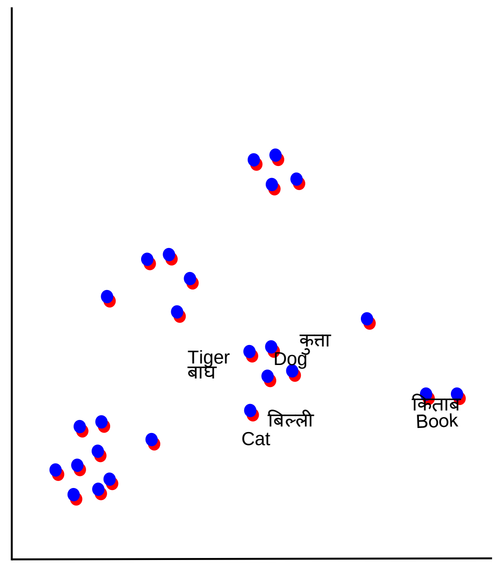




Across Cross-lingual Mapping

This involves strong assumption that embedding spaces across languages are isomorphic, which is not true specifically for distance languages (Søgaard et al. 2018). However, without this assumption unsupervised NMT is not possible.

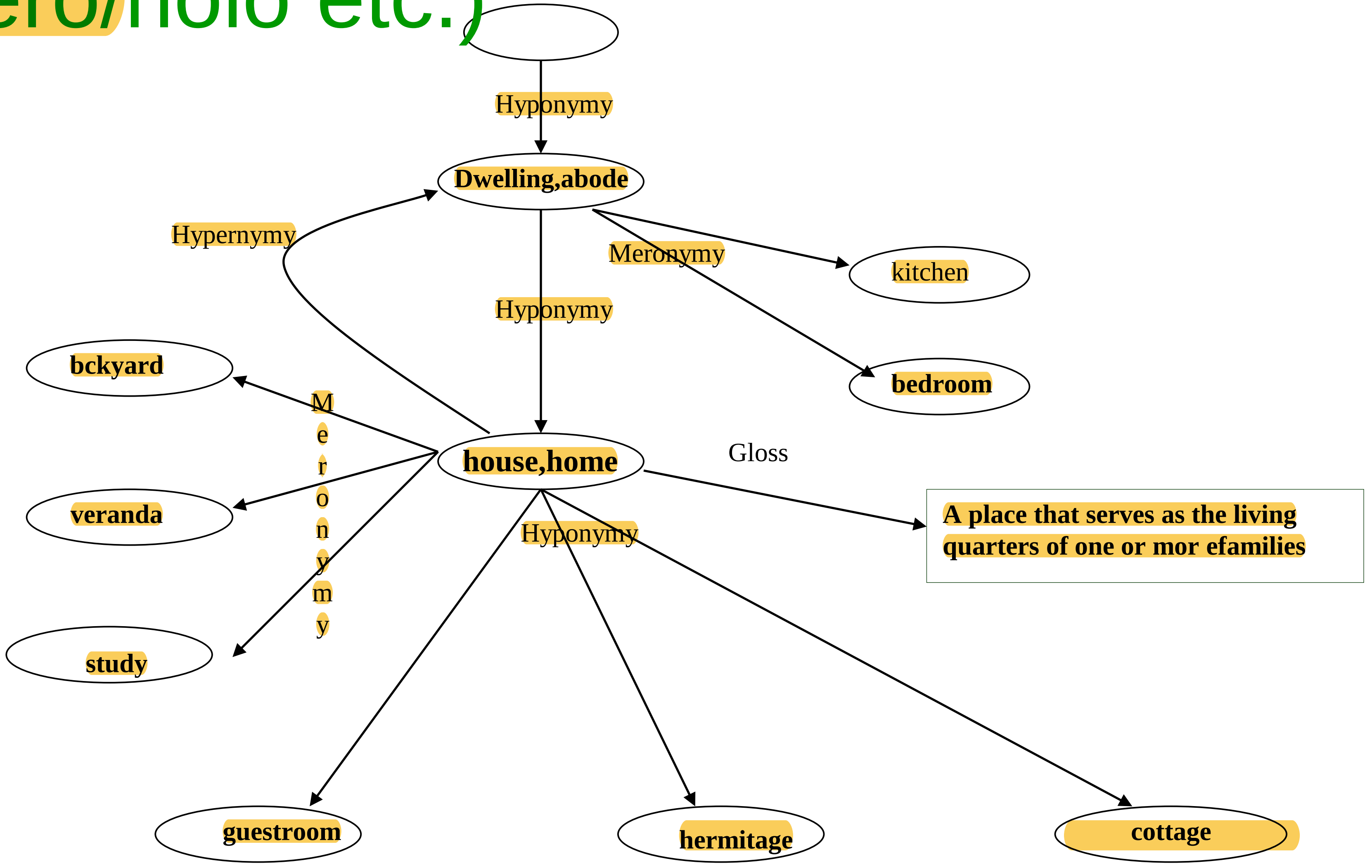
Søgaard, Anders, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. ACL



Foundations-4: Syntagmatic and Paradigmatic Relations

- Syntagmatic and paradigmatic relations
  - Lexico-semantic relations: synonymy, antonymy, hypernymy, mernymy, troponymy etc. **CAT is-a ANIMAL**
  - Coccurence: **CATS MEW**
- Wordnet: primarily paradigmatic relations
- ConceptNet: primarily Syntagmatic Relations

WordNet Sub-Graph with lexico-semantic relations (hyper/hypo, mero/holo etc.)



A place that serves as the living quarters of one or mor efamilies

Lexical and Semantic relations in wordnet

1. Synonymy (e.g., *house, home*)
2. Hypernymy / Hyponymy (kind-of, e.g., *cat* ←→ *animal*)
3. Antonymy (e.g., *white and black*)
4. Meronymy / Holonymy (part of, e.g., *cat and tail*)
5. Gradation (e.g., *sleep*→*doze*→*wake up*)
6. Entailment (e.g., *snoring* → *sleeping*)
7. Troponymy (manner of, e.g., *whispering and talking*)

1, 3 and 5 are lexical (*word to word)*, rest are semantic (*synset to synset).*

‘Paradigmatic Relations’ and ‘Substitutability’

- Words in paradigmatic relations can substitute each other in the sentential context
- E.g., ‘The cat is drinking milk’ → ‘The animal is drinking milk’
- Substitutability is a foundational concept in linguistics and NLP

Foundations-5: Learning and Learning Objective

- Probability of getting the context words given the target should be maximized (skip gram)
- Probability of getting the target given context words should be maximized (CBOW)