Word Embedding

Group 13

Instructor: Dr. Nguyễn An Khương

Team Members

Name	Student ID
Ngô Hoài Tú	2570536
Nguyễn Hoàng Kiên	2570435
Huỳnh Đức Nhâm	2570154
Trần Quốc Việt	2570276

Table of Contents

1. INTRODUCTION TO WORD EMBEDDINGS
2. THE ONE-HOT ENCODING APPROACH 00100
3. PREREQUISITE MATHEMATICAL CONCEPTS
4. SELF-SUPERVISED WORD TO VECTOR METHOD
5. APPROXIMATE TRAINING

Introduction to Word Embeddings

Introduction to Word Embeddings

Problem Statement

In the field of Natural Language Processing (NLP), the fundamental challenge lies in translating human language—which consists of discrete, symbolic, and unstructured data—into a format that machine learning algorithms can process.

Computers operate on numerical data, specifically vectors and matrices. Therefore, we need a robust mapping function to transform the vocabulary of a language into a vector space.

Introduction to Word Embeddings

The Semantic Gap

The core problem is not merely assigning numbers to words, but preserving semantic meaning.

- **Traditional approaches** treated words as atomic symbols with no inherent relationship.
- **The Result:** The system could not understand that "laptop" and "notebook" are related.
- **The Goal:** Word Embedding bridges this gap by creating a continuous vector space where geometric proximity reflects semantic similarity.

Introduction to Word Embeddings

Applications

- Semantic Search.
- Machine Translation
- Recommendation Systems
- Sentiment Analysis

Solving CS Problems

- The Curse of Dimensionality
- Unsupervised Feature Extraction:
- Analogical Reasoning

The One-Hot Encoding Approach

The One-Hot Encoding Approach

Definition and Mechanism

- Before distributed representations, the standard method was **One-Hot Encoding.**
- A One-Hot vector represents a word as a vector $v \in R^{|V|}$ where only one element is 1 and all others are 0. For a vocabulary V=[apple,banana,cat] :

$$w_{\text{apple}} = [1, 0, 0]$$
$$w_{\text{banana}} = [0, 1, 0]$$
$$w_{\text{cat}} = [0, 0, 1]$$

The One-Hot Encoding Approach

Why is One-Hot a bad choice?

While simple, One-Hot encoding suffers from critical flaws:

- **High Dimensionality**: For 100,000 words, every vector length is 100,000. This is computationally wasteful.
- **Data Sparsity**: Vectors are mostly zeros, making gradient updates inefficient in neural networks.
- **Lack of Semantic Similarity (Orthogonality)**: In One-Hot space, every word is orthogonal (90◦) to every other word.The dot product of any two distinct words is always zero.

The model learns zero information about the relationship between synonyms

The One-Hot Encoding Approach

Transition to Probability-Based Representations

To overcome One-Hot limitations, we shift from counting to predicting.

We utilize Self-Supervised Learning based on the Distributional Hypothesis:

> *"A word is characterized by the company it keeps."*

By analyzing the probability of a word appearing within a specific context, Word2Vec (CBOW/Skip-gram) learns dense vectors where similar words have similar probability distributions.

Prerequisite Mathematical Concepts

[Reference] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality.

Prerequisite Mathematical Concepts

Dot Product

- Definition:

$$\mathbf{a}\cdot\mathbf{b}=\sum_{i=1}^{n}a_ib_i=a_1b_1+a_2b_2+\cdots+a_nb_n$$

- Geometrically, it relates to the angle θ :

$$a\cdot b=||a||||b||\cos(\theta)$$

Prerequisite Mathematical Concepts

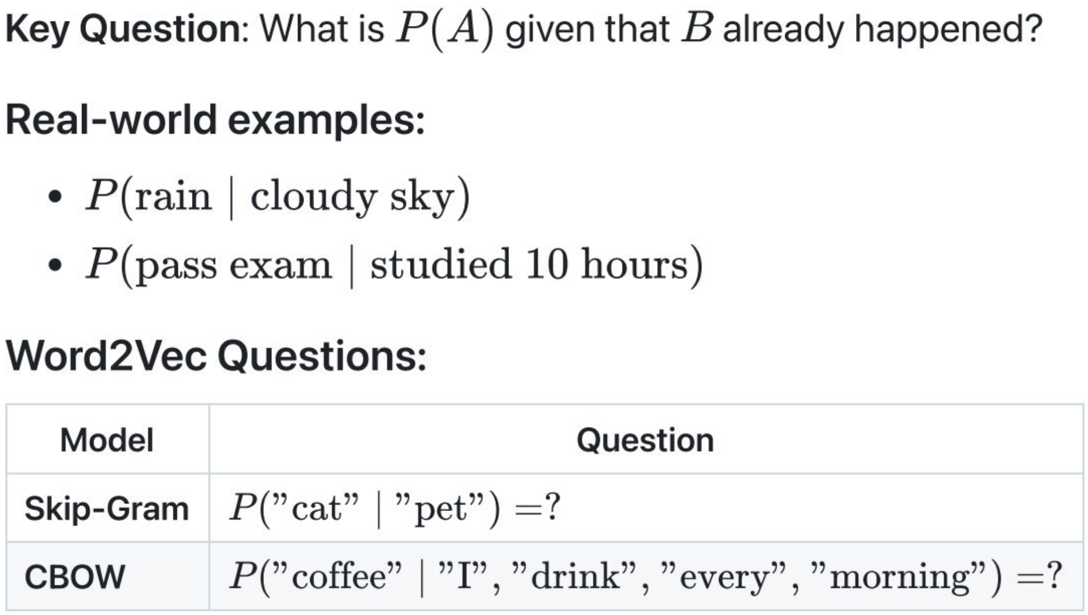
Cosine Similarity

- To overcome the magnitude dependency of the dot product, we use Cosine Similarity. This is the gold standard in NLP for determining semantic similarity. It measures the cosine of the angle θ between two vectors.

$$\text{Cosine Similarity} = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||\,||\mathbf{b}||}$$

Prerequisite Mathematical Concepts

What is Conditional Probability?



Key Question: What is $P(A)$ given that B already happened?

Real-world examples:

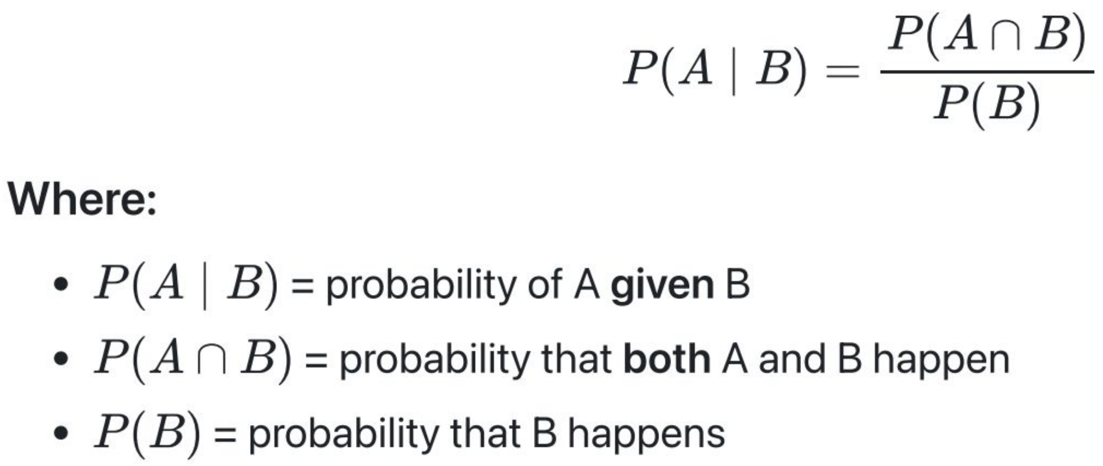
- $P(\text{rain} \mid \text{cloudy sky})$
- $P(\text{pass exam} \mid \text{studied 10 hours})$

Word2Vec Questions:

Model	Question
Skip-Gram	$P(\text{”cat”} \mid \text{”pet”}) = ?$
CBOW	$P(\text{”coffee”} \mid \text{”I”}, \text{”drink”}, \text{”every”}, \text{”morning”}) = ?$

Prerequisite Mathematical Concepts

The Formula of Conditional Probability



$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

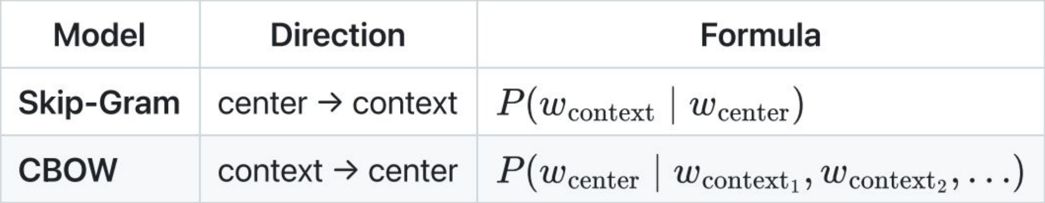
Where:

- $P(A \mid B)$ = probability of A **given** B
- $P(A \cap B)$ = probability that **both** A and B happen
- $P(B)$ = probability that B happens

Key Insight: *Conditioning = **narrowing down** the sample space*

Prerequisite Mathematical Concepts

Conditional Probability (connection to Word2Vec)



Model	Direction	Formula
Skip-Gram	center → context	$P(w_{\text{context}} \mid w_{\text{center}})$
CBOW	context → center	$P(w_{\text{center}} \mid w_{\text{context}_1}, w_{\text{context}_2}, \ldots)$

Example: "The cat sat on the mat"

- **Skip-Gram** asks: Given "cat", predict:

$$P(\text{”The”} \mid \text{”cat”}), P(\text{”sat”} \mid \text{”cat”}), P(\text{”on”} \mid \text{”cat”})$$

- **CBOW asks**: Given ["The","sat","on"], predict:

$$P(\text{”cat”} \mid \text{”The”}, \text{”sat”}, \text{”on”})$$

Prerequisite Mathematical Concepts

What is Maximum Likelihood Estimation (MLE)?

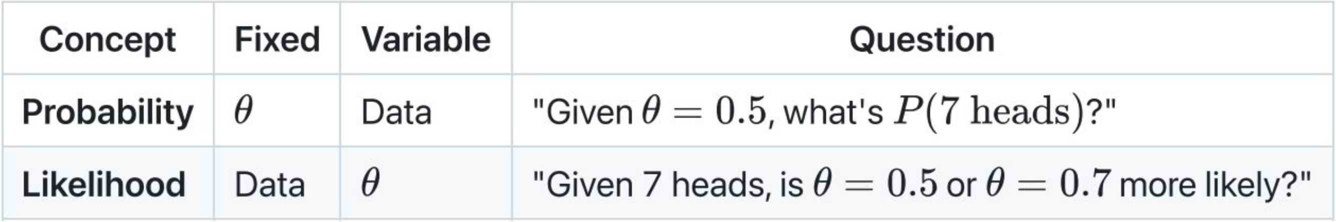
Core Idea: Find parameters that make observed data **MOST PROBABLE**

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} P(\text{Data} \mid \theta)$$

Analogy: *"Detective work" - which best explains what we observed?*

Prerequisite Mathematical Concepts

Likelihood vs Probability



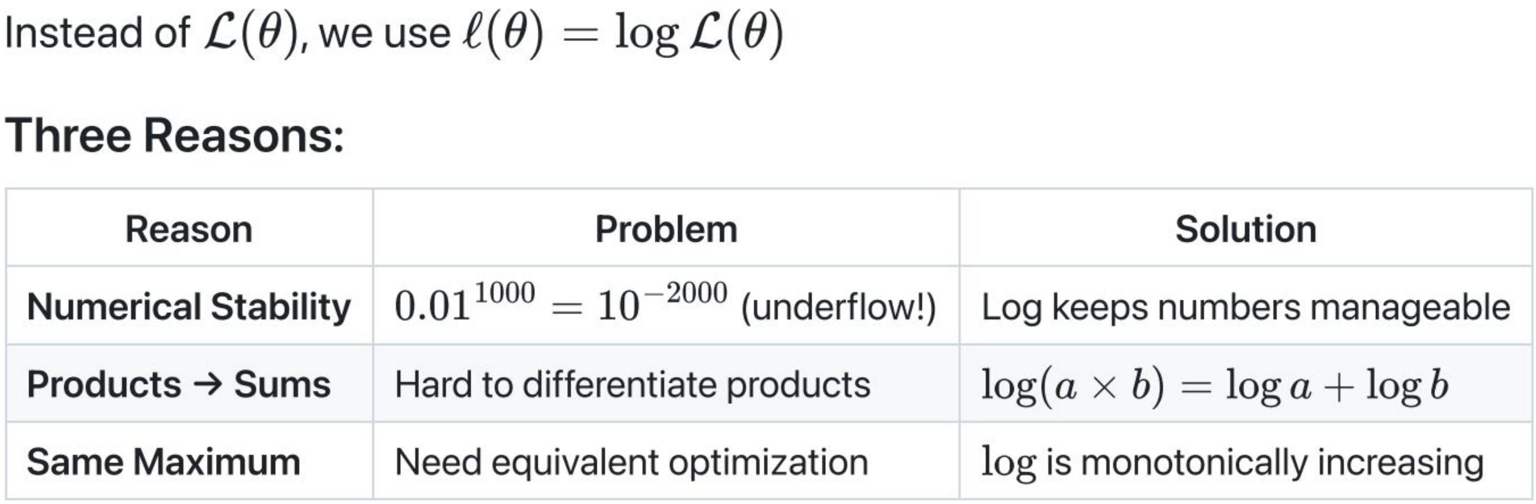
Concept	Fixed	Variable	Question
Probability	θ	Data	"Given $\theta = 0.5$, what's $P(7 \text{ heads})$?"
Likelihood	Data	θ	"Given 7 heads, is $\theta = 0.5$ or $\theta = 0.7$ more likely?"

Key Insight: *Same number, different interpretation!*

- Probability: Variable is fixed, Data varies
- Likelihood: Data is fixed, Variable varies

Prerequisite Mathematical Concepts

Why Log-Likelihood?



Instead of $\mathcal{L}(\theta)$, we use $\ell(\theta) = \log \mathcal{L}(\theta)$

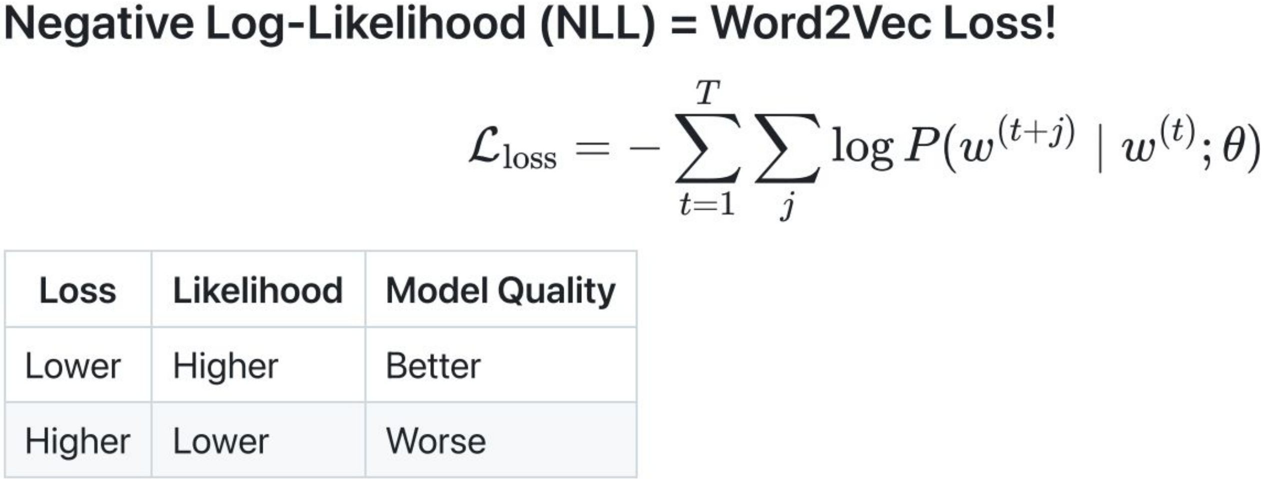
Three Reasons:

Reason	Problem	Solution
Numerical Stability	$0.01^{1000} = 10^{-2000}$ (underflow!)	Log keeps numbers manageable
Products → Sums	Hard to differentiate products	$\log(a \times b) = \log a + \log b$
Same Maximum	Need equivalent optimization	$\log$ is monotonically increasing

Prerequisite Mathematical Concepts

From MLE to Loss Function

$$\text{Maximize } \mathcal{L}(\theta) \iff \text{Maximize } \log \mathcal{L}(\theta) \iff \text{Minimize } -\log \mathcal{L}(\theta)$$



Negative Log-Likelihood (NLL) = Word2Vec Loss!

$$\mathcal{L}_{\text{loss}} = -\sum_{t=1}^{T} \sum_{j} \log P(w^{(t+j)} \mid w^{(t)}; \theta)$$

Loss	Likelihood	Model Quality
Lower	Higher	Better
Higher	Lower	Worse

Prerequisite Mathematical Concepts

What is Softmax?

Purpose: Convert arbitrary scores → **probability distribution**

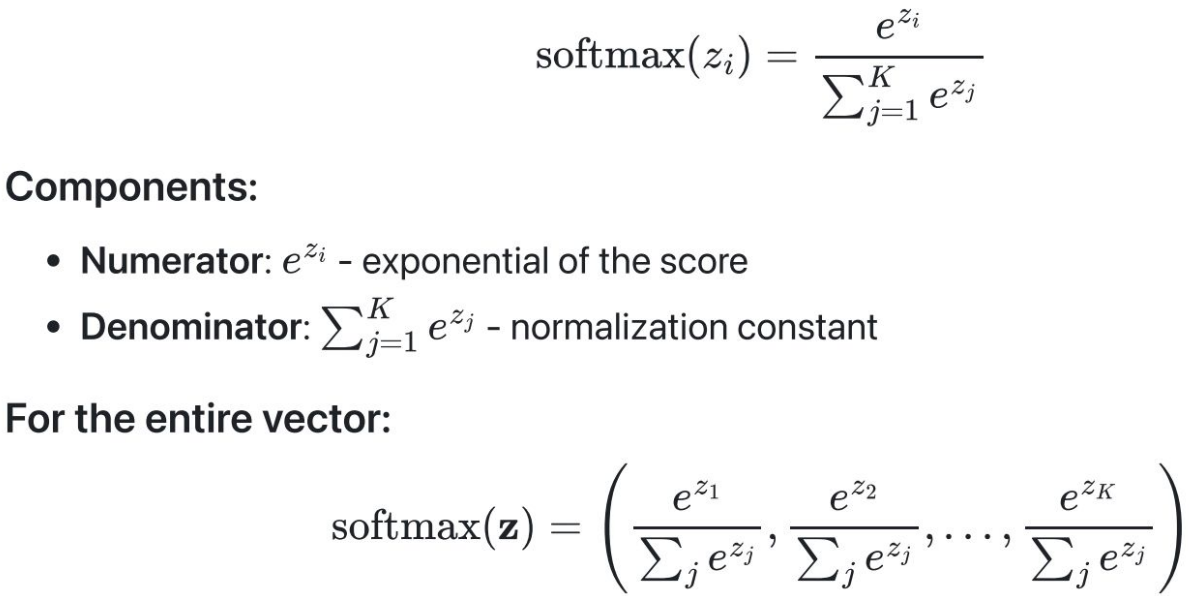
Properties:

1. All outputs are positive (between 0 and 1)
2. All outputs sum to exactly 1

Perfect for: *Multi-class classification where we need probability of each class*

Prerequisite Mathematical Concepts

The Softmax Formula



$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Components:

- **Numerator**: e^{z_i} - exponential of the score
- **Denominator**: $\sum_{j=1}^{K} e^{z_j}$ - normalization constant

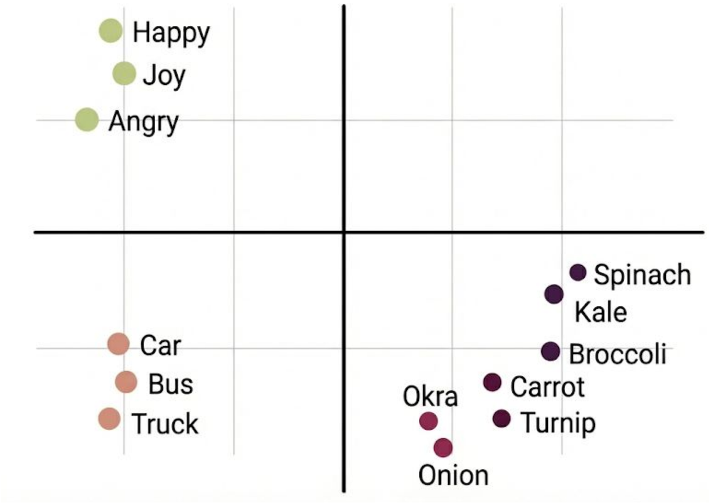
For the entire vector:

$$\text{softmax}(\mathbf{z}) = \left(\frac{e^{z_1}}{\sum_j e^{z_j}}, \frac{e^{z_2}}{\sum_j e^{z_j}}, \ldots, \frac{e^{z_K}}{\sum_j e^{z_j}} \right)$$

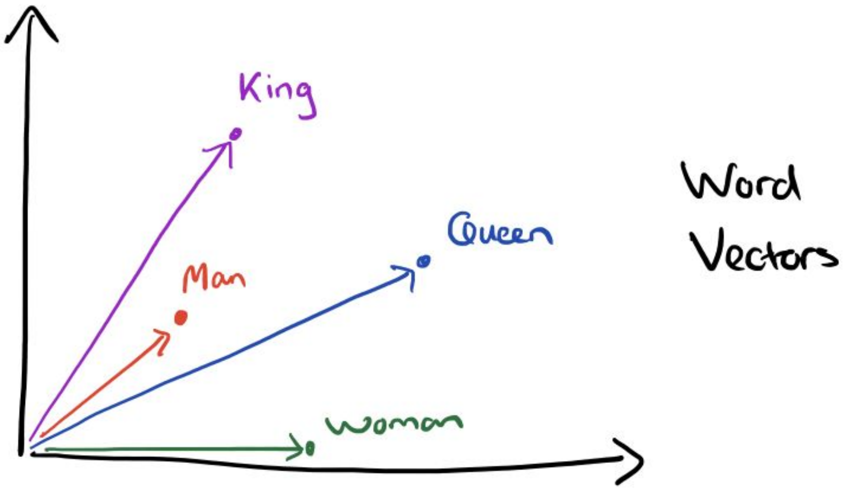
Self-Supervised Word to Vector Method

[Reference] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality.

Goal



Similarity in vector space



Optimize word offset technique

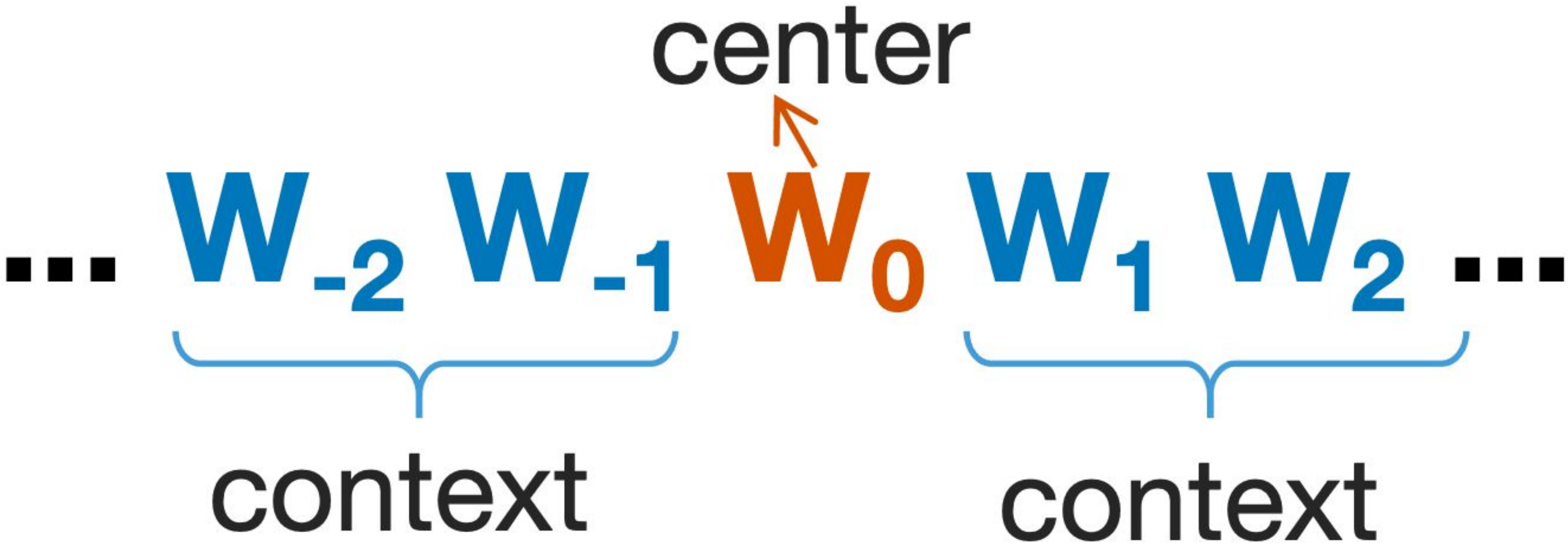
$$vector(''\text{King}'') - vector(''\text{Man}'') + vector(''\text{Woman}'') \sim vector(''\text{Queen}'')$$

Missions

Preserve the linear regularities among words

Maximize accuracy of simple algebraic vector operations

Minimizing computational complexity by using simple model



center

$\ldots\ w_{-2}\ w_{-1}\ w_0\ w_1\ w_2\ \ldots$

context

context

Continuous Bag of Words Model (CBOW)

The Dual-Vector of a Word

We have a dictionary of words calling **V**. Suppose, each word at index *i* is represented by a vector $\mathbf{v}_i$ (R^N) - This is also our desired outcome.

A model **M** (not CBOW model) help us predict a blank in a sentence.
Ex: **"we are ___ friends forever"**,
M("we", "are", "friends", "forever") = "good"

Continuous Bag of Words Model (CBOW)

The Dual-Vector of a Word

$$M(v_{we},\, v_{are},\, v_{friends},\, v_{forever}) = \text{''good''} \sim M(\bar{v}) = \text{''good''} \sim \max p(\text{''good''} \mid \bar{v}\ \ )$$

$$\bar{v} = \frac{v_{we} + v_{are} + v_{friends} + v_{forever}}{4}$$

Continuous Bag of Words Model (CBOW)

The Dual-Vector of a Word

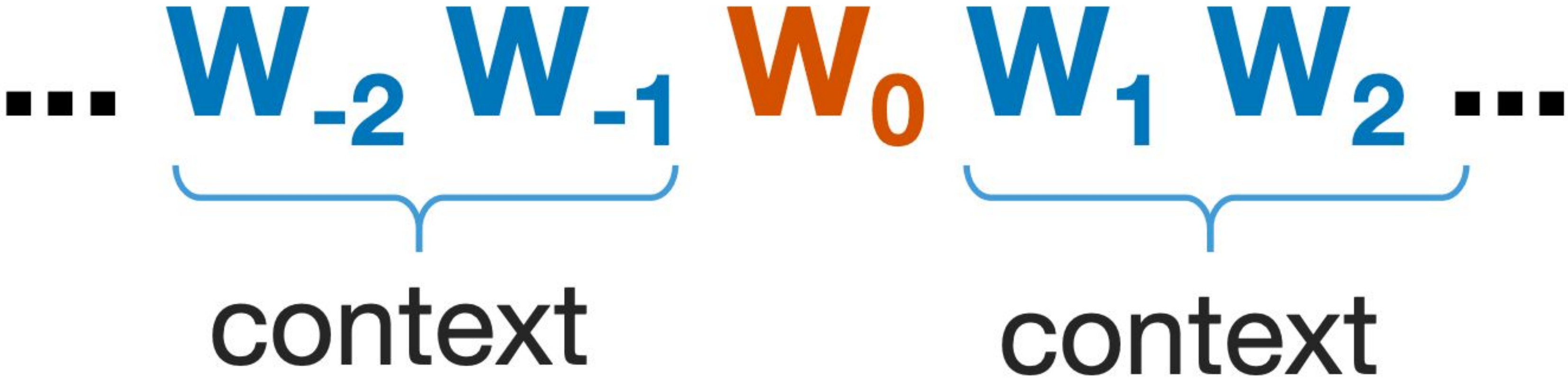
We define $\mathbf{u}_{\boldsymbol{good}}$ ($\boldsymbol{R}^{\mathrm{N}}$) to be the label of class “good”

$$p(\text{”good”}\,|\,\bar{v}\ \ ) = p(u_{good}\,|\,\bar{v}\ \ )$$

The blank word is called **center word**, the surrounding words are called **context words**

Continuous Bag of Words Model (CBOW)

v_{-1} v_{-2} u_c v_1 v_2



Continuous Bag of Words Model (CBOW)

The Dual-Vector of a Word

Set context window size to **m**, we have:

$$p(output = u_c \mid input = v_1, v_2, \ldots, v_{2m})$$

A word at index ***i*** in dictionary **V** has two vector ($\mathbf{R}^N$)

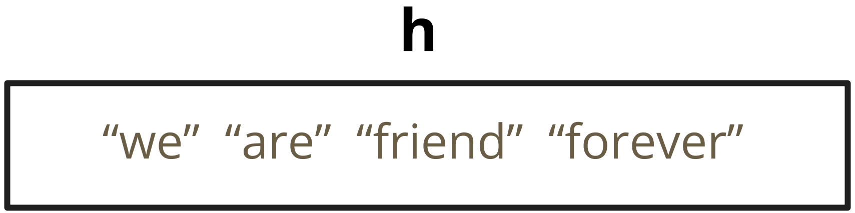
$\boldsymbol{v}_{\mathrm{i}}$ is the feature vector (outcome) of a word - representing its context role

$\boldsymbol{u}_{\mathrm{i}}$ is the auxiliary vector of a word - representing its center role

Continuous Bag of Words Model (CBOW)

The Dual-Vector of a Word

We define new vector concept calling **h** $\in \mathbb{R}^N$ to represent the context (list of words, **not only a single** word) in vector space



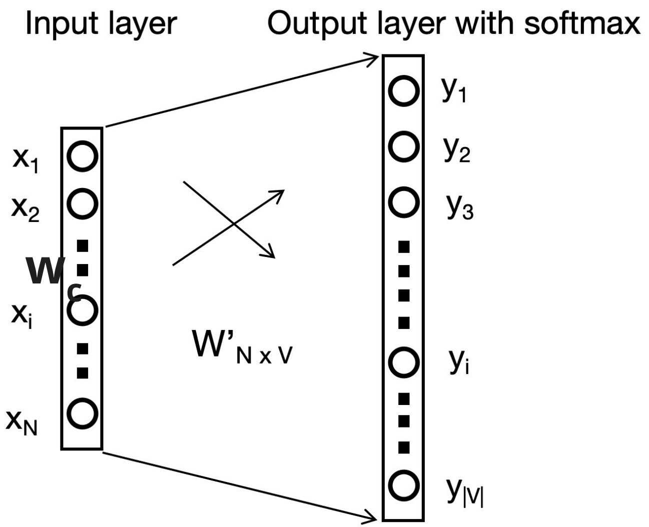
Context to Center Word

We define a new simple model $\boldsymbol{M'}$ containing only two layers:

- Input layer is context $\boldsymbol{h}$

- Output layer is One-hot vector $y \in \mathbb{R}^{|V|}$ of the center word $\boldsymbol{w_c}$ (c is index in Dict).

The output layer is handled with the softmax function



M' help us predict a blank in a sentence.
Ex: **"we are ___ friends forever"**,
M(“we”, “are”, “friends”, “forever”) = "good"

h

Context to Center Word

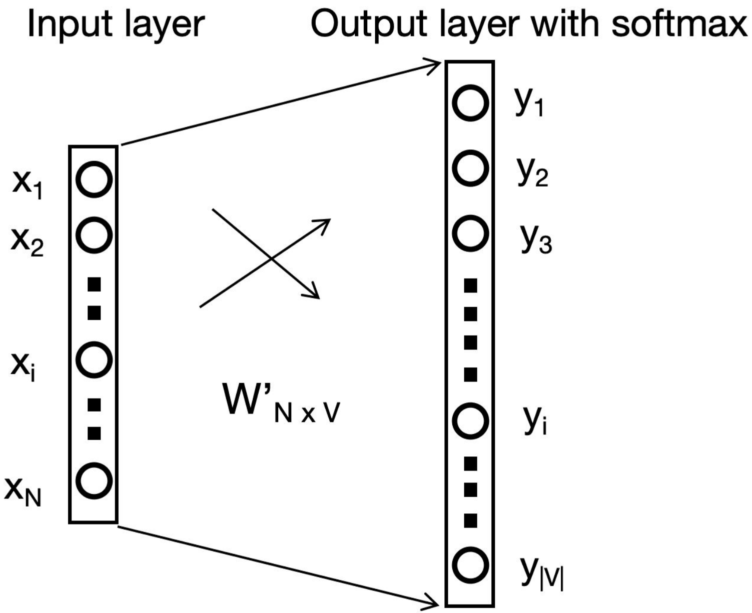
$$y = W'^{T}.x$$

Following the softmax of this simple neural net, we have:

$$p(word_i \mid x = h) = y_i = \frac{exp(W'^{T}_{(\cdot,\, i)} \cdot x)}{\Sigma_{j=1}^{|V|} exp(W'^{T}_{(\cdot,\, j)} \cdot x)}$$

where y_i is the output of the *i-th* unit in the output,

$W'_{(\cdot,\, i)} \in \mathbb{R}^{N x 1}$ is the *i-th* column of the matrix **W '**

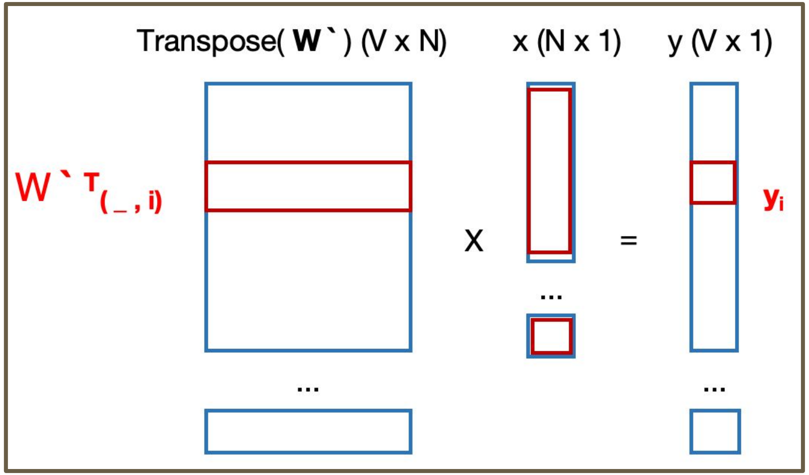


Context to Center Word

$$y = W'^T . x$$

Following the softmax of this simple neural net, we have:

$$p(word_i \mid x = h) = y_i = \frac{exp(W'^T_{(\cdot,\, i)} \cdot x)}{\Sigma_{j=1}^{|V|} exp(W'^T_{(\cdot,\, j)} \cdot x)}$$



where y$_i$ is the output of the *i-th* unit in the output,

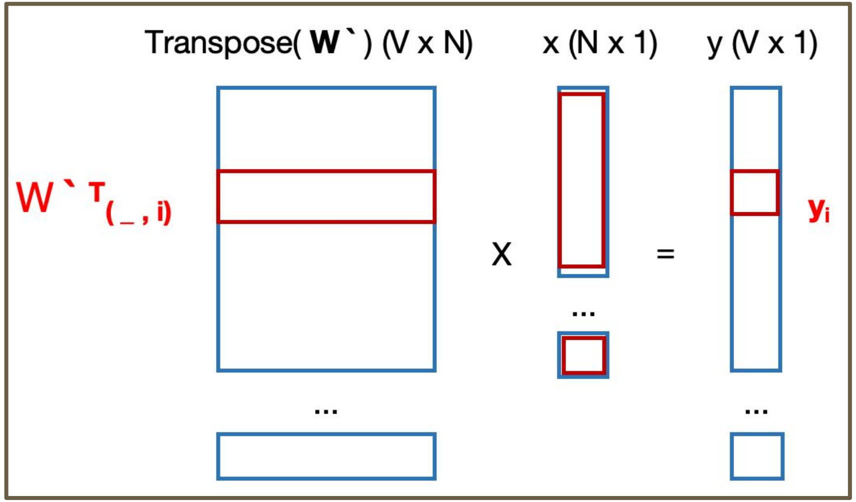
$W'_{(\cdot,\, i)} \in \mathbb{R}^{N\,x\,1}$ is the *i-th* column of the matrix **W ’**

Context to Center Word

Call c is the true index of the center word, the target of training is maximizing the probability:

$$p(word_c \mid x = h)$$

The maximum $p(word_c \mid x = h)$ is related with the weights column $W'(\cdot, c)$ so we assign:



$$u_c := W'_{(\cdot,\, c)} \text{ or } u_i := W'_{(\cdot,\, i)}$$

Therefore:

$$p(word_c \mid x = h) = y_c = \frac{exp(u_c^T \cdot h)}{\Sigma_{j=1}^{|V|} exp(u_j^T \cdot h)}$$

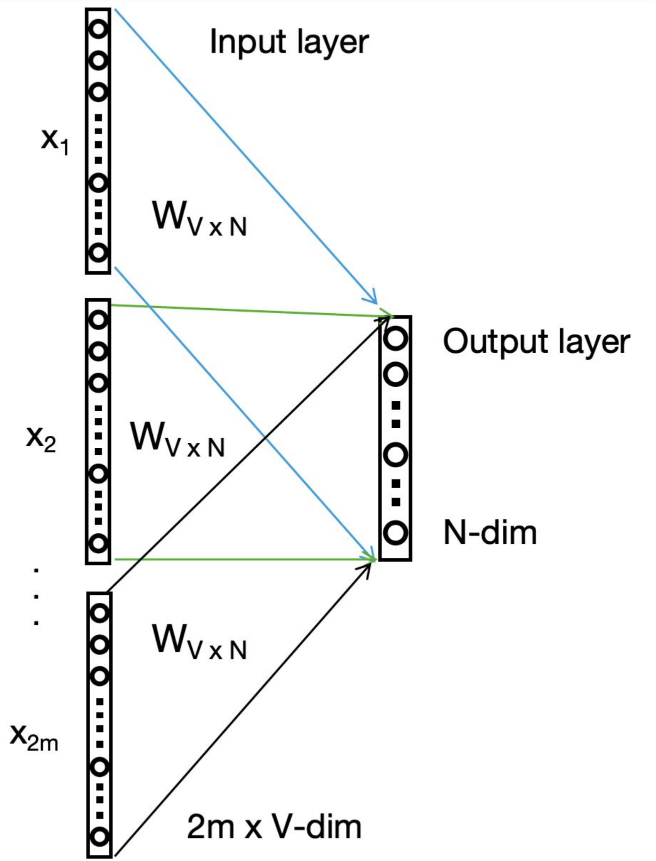
Surrounding words to Context vector

We define a new simple model ***M*** **``** containing only two layers:

- Input layer is multi-inputs which are One-hot vectors $x_i \in \mathbb{R}^{|V|}$ for context $w_1, w_2, \ldots, w_{2m}$

- Output layer is $y \in \mathbb{R}^N$

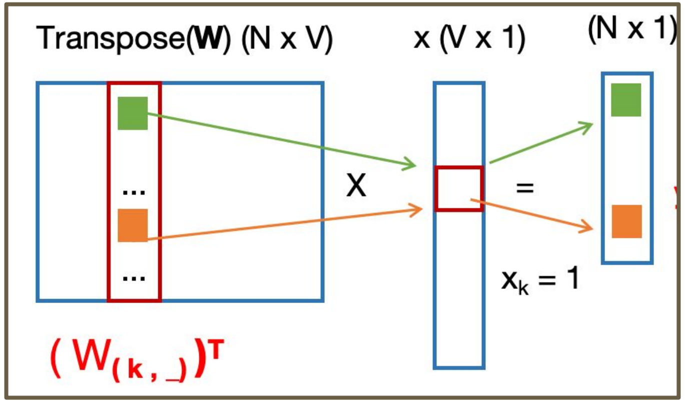
*Target of **M ' '** : the output vector is the representation of **the context (list of words)** in vector space (vector h).*



Surrounding words to Context vector

We take the average of the vectors of the input context words, and use the product of the input:

$$y = \frac{1}{2m}(W^T x_1 + W^T x_2 + \cdots + W^T x_{2m}) = \frac{1}{2m} W^T \cdot (x_1 + x_2 + \cdots + x_{2m})$$



W is the weights of the model M'' shared for all inputs.

Given a context, assuming $x^k_i = 1$ and $x^{k'}_i \neq 0$ for $k \neq k'$, we set:

$$W^T \cdot x_i = W^T_{(k,\cdot)} := v^T_{w_k}$$

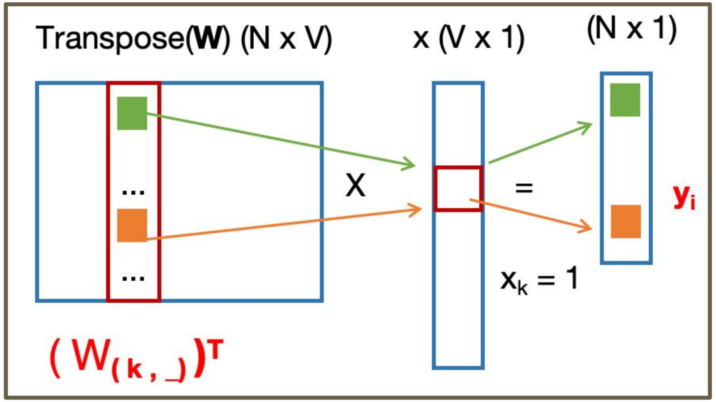
We can assign $\boldsymbol{W}_{(k,\cdot)} := \boldsymbol{v}_{wk}$, because the output vector is contributed by $\boldsymbol{word}_k$ at $\boldsymbol{x}_i$ ~ $\boldsymbol{W}_{(k,\cdot)}$

Surrounding words to Context vector

Given a context, assuming $x^k_i = 1$ and $x^{k'}_i \neq 0$ for $k \neq k'$, we set:

$$W^T \cdot x_i = W^T_{(k,\cdot)} := v_{w_k}$$

We can assign $\boldsymbol{W}_{(k,\cdot)} := \boldsymbol{v}_{w\,k}$, because the output vector is contributed by $\boldsymbol{word}_k$ at $\boldsymbol{x}_i$ ~ $\boldsymbol{W}_{(k,\cdot)}$



Therefore:

$$y = \frac{1}{2m}\left(v_{w_1} + v_{w_2} + \ldots + v_{w_{2m}}\right) = \frac{1}{2m}\left(v_1 + v_2 + \ldots + v_{2m}\right)$$

*Then, we assign **y := h** (context vector)*

Full CBOW Model

From M `

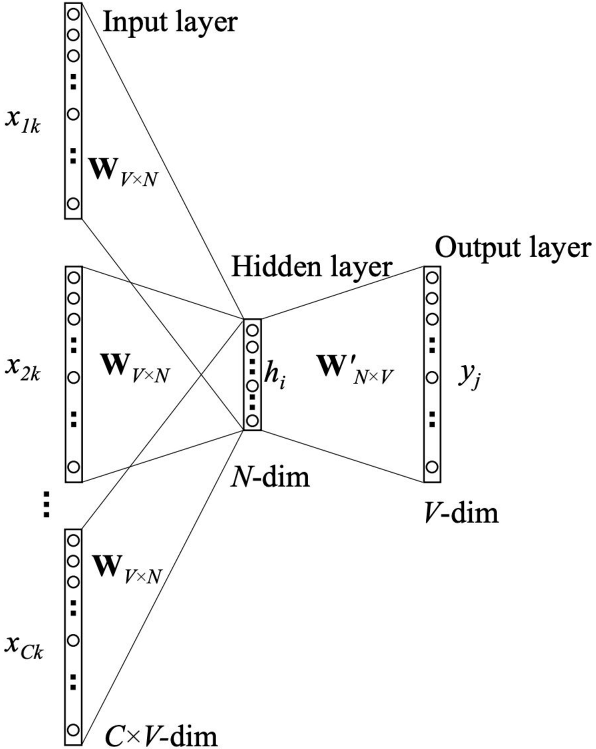
$$p(word_c \mid x = h) = y_c = \frac{exp(u_c^T \cdot h)}{\Sigma_{j=1}^{|V|} exp(u_j^T \cdot h)}$$

From M ` `

$$h = \frac{1}{2m}(v_1 + v_2 + \ldots + v_{2m})$$

Combine

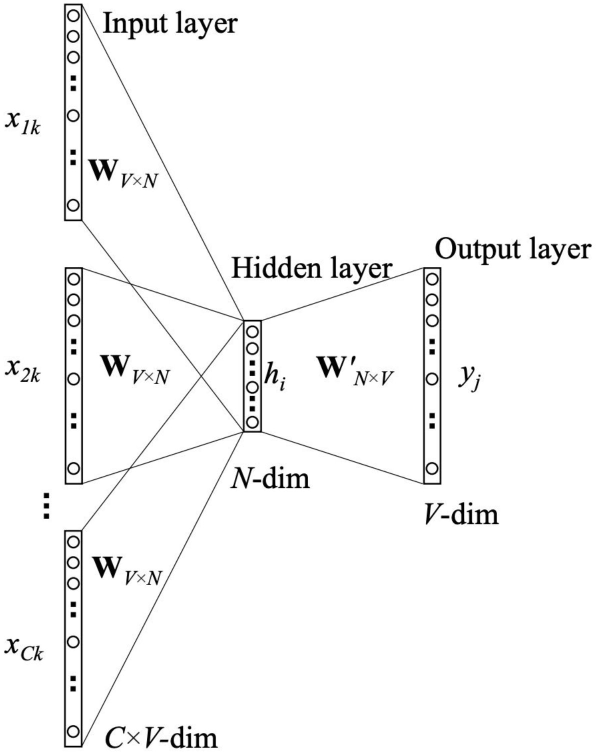
$$p(w_c \mid \mathscr{W}_o) = \frac{exp(\frac{1}{2m}u_c^T(v_{o_1} + v_{o_2} + \cdots + v_{o_{2m}}))}{\Sigma_{i=1}^{|V|} exp(\frac{1}{2m}u_i^T(v_{o_1} + v_{o_2} + \cdots + v_{o_{2m}}))} = \frac{exp(u_c^T \cdot \bar{v_o})}{\Sigma_{i=1}^{|V|} exp(u_i^T \cdot \bar{v_o})}$$



Full CBOW Model

Given a text sequence **T** for training, Following the Maximum likelihood estimation (MLE), we will train the full CBOW model by maximizing this likelihood function:

$$\prod_{t=1}^{T} p\left(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)}\right)$$

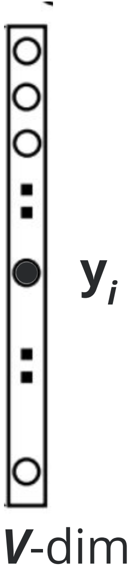


CBOW model Training

$$max\, p(w_c \mid \mathscr{W}_o) \Leftrightarrow max\, y_i \quad \Leftrightarrow max\ log\, y_i$$

$$log\, y_i = log(exp(u_c^T \cdot \bar{v}_o)) - log(\Sigma_{i=1}^{|V|} exp(u_i^T \cdot \bar{v}_o)) = u_c^T \cdot \bar{v}_o - log(\Sigma_{i=1}^{|V|} exp(u_i^T \cdot \bar{v}_o)) \ := -L$$

with y_i is the **i-th** element of the *y* vector - **i** is index of **word c** in dictionary



Thus: $L = log(\Sigma_{i=1}^{|V|} exp(u_i^T \cdot \bar{v}_o)) - u_c^T \cdot \bar{v}_o$

Loss of sequence T:

$$\text{\L} = -\Sigma_{t=1}^{T} log\, p\left(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)},\ w^{(t+1)}, \ldots, w^{(t+m)}\right) = \Sigma_{t=1}^{T} L_t$$

CBOW model Training

Through differentiation, we can obtain its gradient with respect to any context word vector $\boldsymbol{v}_{\mathbf{o}_i}$ (i=1,...,2m) as:

$$\frac{\partial L}{\partial v_{o_i}} = \frac{\partial \, log \, p(w_c \mid \mathcal{W}_o)}{\partial v_{o_i}}$$

CBOW model Training

Through differentiation, we can obtain its gradient with respect to any context word vector $\boldsymbol{v}_{\mathbf{o}\boldsymbol{i}}$ (i=1,...,2m) as:

$$\frac{\partial L}{\partial v_{o_i}} = \frac{\partial \, log\, p(w_c \mid \mathscr{W}_o)}{\partial \, v_{o_i}} = \frac{\partial \left(log(\Sigma_{i=1}^{|V|} \, exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))) - u_c^T \cdot \, \frac{1}{2m}(v_1 + .. + v_{o_i} + .. + v_{2m})\right)}{\partial \, v_{o_i}}$$

$$\boxed{= \frac{(\Sigma_{i=1}^{|V|} \, exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m})))'}{\Sigma_{j=1}^{|V|} \, exp(u_j^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))} - \frac{1}{2m} u_c}$$

CBOW model Training

Through differentiation, we can obtain its gradient with respect to any context word vector $\boldsymbol{v}_{\mathbf{o}_i}$ (i=1,...,2m) as:

$$= \frac{(\Sigma_{i=1}^{|V|} exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m})))'}{\Sigma_{j=1}^{|V|} exp(u_j^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))} - \frac{1}{2m} u_c$$

$$\boxed{= \frac{\Sigma_{i=1}^{|V|} \; (exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m})) \cdot (u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))')}{\Sigma_{j=1}^{|V|} exp(u_j^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))} - \frac{1}{2m} u_c}$$

CBOW model Training

Through differentiation, we can obtain its gradient with respect to any context word vector $\boldsymbol{v}_{\mathbf{o}\boldsymbol{i}}$ $(i{=}1,\ldots,2m)$ as:

$$= \frac{\Sigma_{i=1}^{|V|}\ \left(exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m})) \cdot (u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))'\right)}{\Sigma_{j=1}^{|V|}\ exp(u_j^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))} - \frac{1}{2m}u_c$$

$$\boxed{= \frac{\Sigma_{i=1}^{|V|}\ \left(exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m})) \cdot (u_i \cdot \frac{1}{2m})\right)}{\Sigma_{j=1}^{|V|}\ exp(u_j^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))} - \frac{1}{2m}u_c}$$

CBOW model Training

Through differentiation, we can obtain its gradient with respect to any context word vector $\boldsymbol{v}_{\mathbf{O}i}$ (i=1,...,2m) as:

$$= \frac{\Sigma_{i=1}^{|V|}\ \left(exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m})) \cdot (u_i \cdot \frac{1}{2m})\right)}{\Sigma_{j=1}^{|V|}\ exp(u_j^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))} - \frac{1}{2m} u_c$$

$$\boxed{\begin{aligned} &= \frac{1}{2m}\left(\frac{\Sigma_{i=1}^{|V|}\ \left(exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m})) \cdot u_i\ \right)}{\Sigma_{j=1}^{|V|}\ exp(u_j^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))} - u_c\right) \\ &= \frac{1}{2m}\left(\Sigma_{i=1}^{|V|}\ \frac{exp(u_i^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}) \cdot u_i}{\Sigma_{j=1}^{|V|}\ exp(u_j^T \cdot \frac{1}{2m}(v_1 + \ldots + v_{2m}))} - u_c\right) \end{aligned}}$$

CBOW model Training

Through differentiation, we can obtain its gradient with respect to any context word vector v_{Oi} (i=1,...,2m) as:

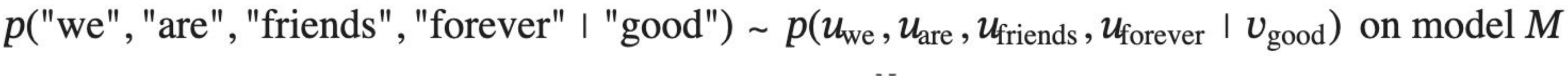
$$= \frac{1}{2m}\left(\Sigma_{i=1}^{|V|}\frac{exp(u_i^T\cdot\frac{1}{2m}(v_1+\ldots+v_{2m})\cdot u_i}{\Sigma_{j=1}^{|V|}\, exp(u_j^T\cdot\frac{1}{2m}(v_1+\ldots+v_{2m}))} - u_c\right)$$

$$\boxed{= \frac{1}{2m}\left(\Sigma_{i=1}^{|V|}\frac{exp(u_i^T\cdot\bar{v}_o)\cdot u_i}{\Sigma_{j=1}^{|V|}\, exp(u_j^T\cdot\bar{v}_o)} - u_c\right) = \frac{1}{2m}\left(\Sigma_{i=1}^{|V|}\, p(w_i \mid \mathscr{W}_o).\, u_i \; - u_c\right)}$$

Since $2m \ll |V|$ in language tasks: $O(|V|)$

Skip-Gram Model

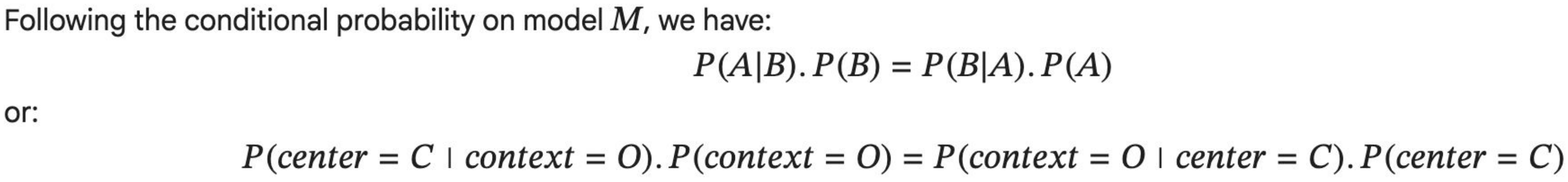
Reversely to model M:



$$p(\text{"we"}, \text{"are"}, \text{"friends"}, \text{"forever"} \mid \text{"good"}) \sim p(u_{\text{we}}, u_{\text{are}}, u_{\text{friends}}, u_{\text{forever}} \mid v_{\text{good}}) \text{ on model } M$$

Therefore, for any word with index $\boldsymbol{i}$ in the dictionary, we call vector $\boldsymbol{v_i} \in \mathbb{R}^N$ as its representation in center role and vector $\boldsymbol{u_i} \in \mathbb{R}^N$ as its representation in context role.

Why CBOW and Skip-Gram are consistent:



Following the conditional probability on model M, we have:

$$P(A|B).P(B) = P(B|A).P(A)$$

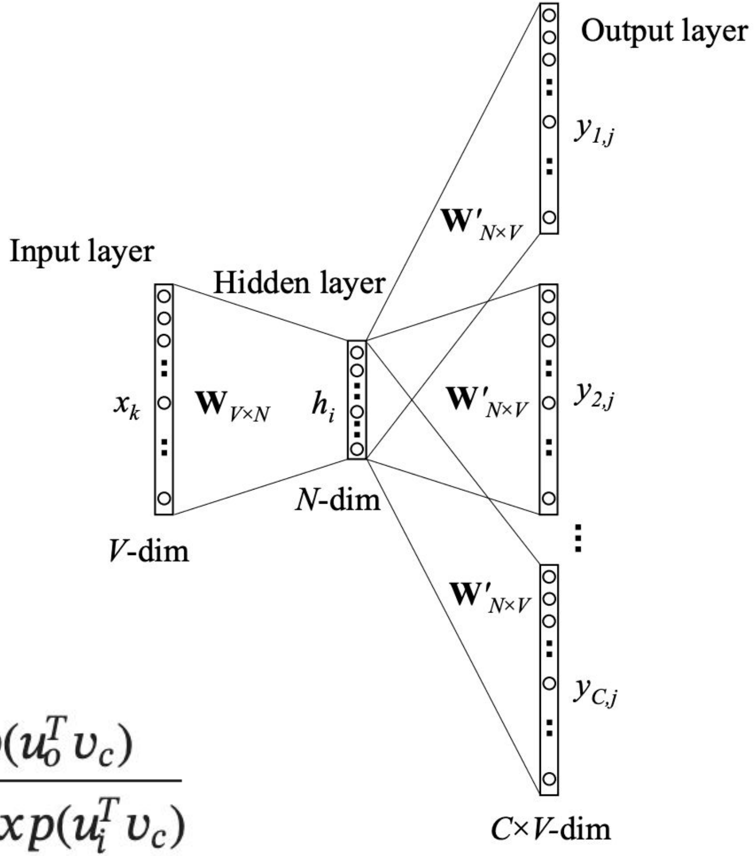
or:

$$P(center = C \mid context = O).P(context = O) = P(context = O \mid center = C).P(center = C)$$

Architecture

$W_{V\,x\,N}$ is the weights matrix whose each row at index $\boldsymbol{i}$ is the $\boldsymbol{v_i}$ vector of word $\boldsymbol{i}$ in dictionary.

$W'_{N\,x\,V}$ is the weights matrix whose each column at index $\boldsymbol{j}$ is the $\boldsymbol{u_j}$ vector of word $\boldsymbol{j}$ in dictionary.



Similar to CBOW, at output layer with softmax, we have $\boldsymbol{p(w_o \mid w_c)}$ formula as below, where $\boldsymbol{w_o}$ is the word at index $\boldsymbol{o}$ and $\boldsymbol{w_c}$ is the word at index $\boldsymbol{c}$ in the dictionary.

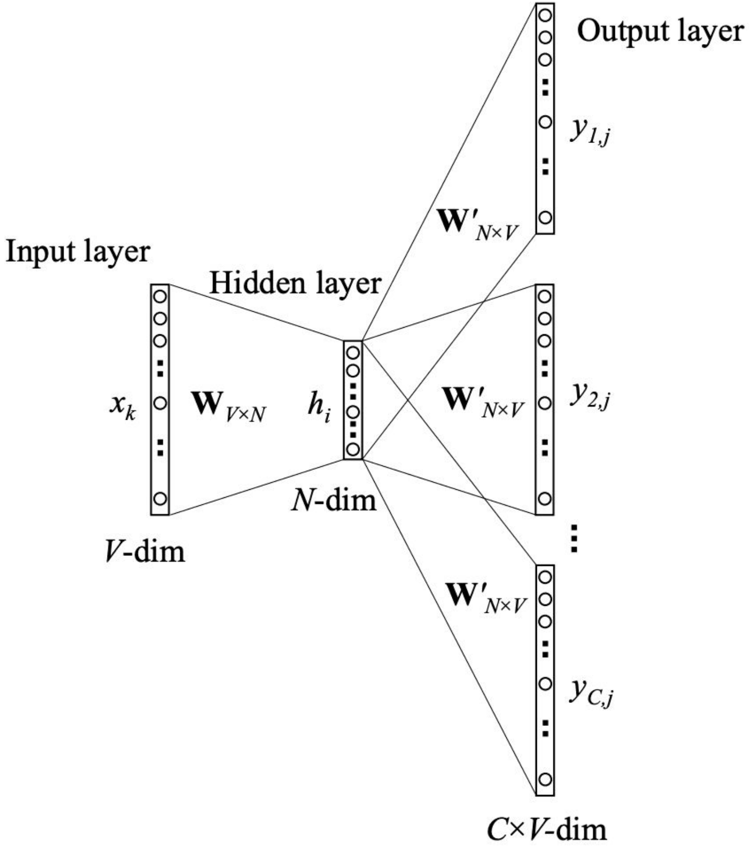
$$h = W^T_{(c,\cdot)} = v_c$$

$$p(w_o \mid w_c) = y^o_{ij} = softmax(W'^T \cdot h)^o = \frac{exp(u_o^T v_c)}{\sum_{i=1}^{|V|} exp(u_i^T v_c)}$$

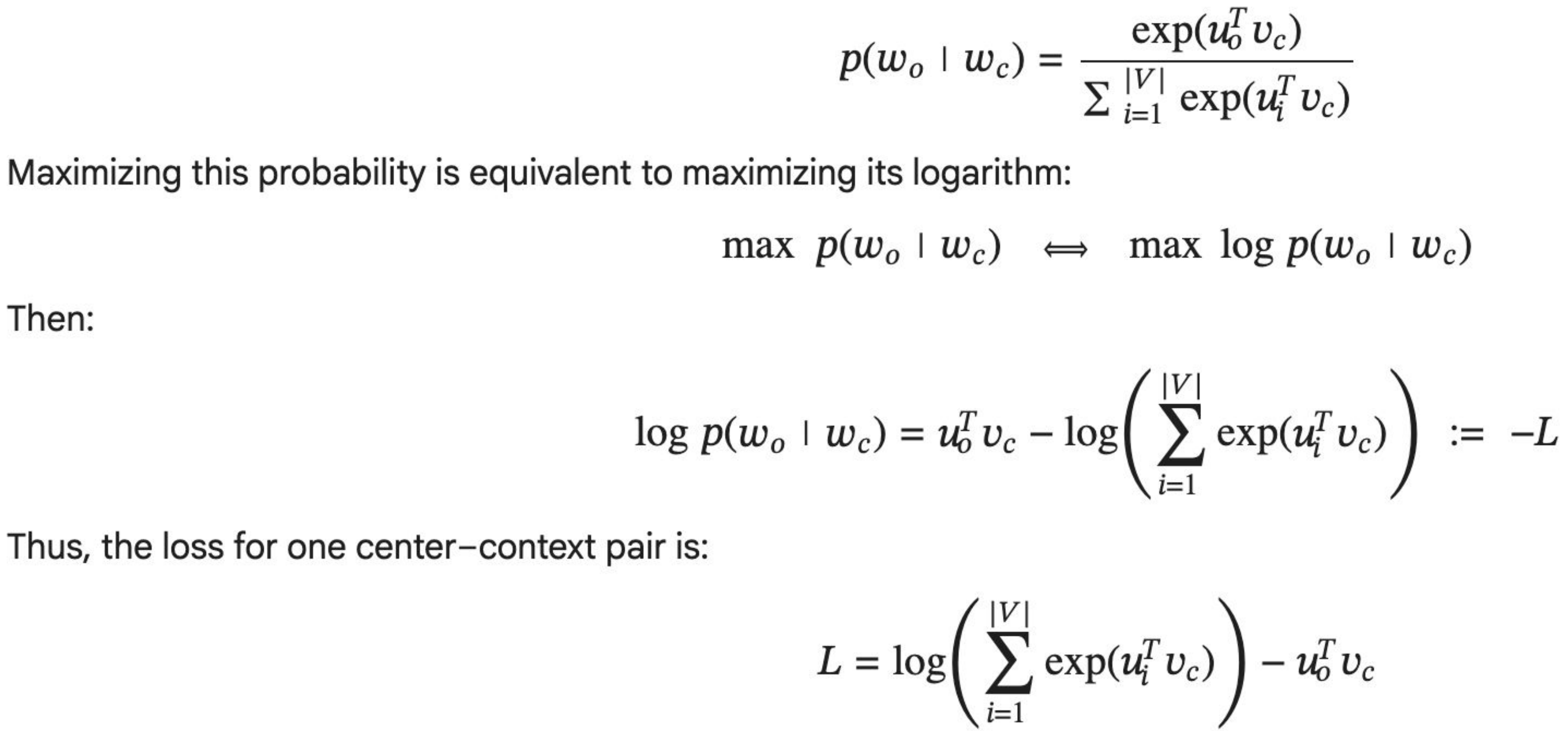
Training

Maximum likelihood estimation

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} p(w^{t+j} \mid w^{(t)})$$



Training



$$p(w_o \mid w_c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{|V|} \exp(u_i^T v_c)}$$

Maximizing this probability is equivalent to maximizing its logarithm:

$$\max\ p(w_o \mid w_c) \quad \Longleftrightarrow \quad \max\ \log p(w_o \mid w_c)$$

Then:

$$\log p(w_o \mid w_c) = u_o^T v_c - \log\left(\sum_{i=1}^{|V|} \exp(u_i^T v_c)\right) \ := \ -L$$

Thus, the loss for one center–context pair is:

$$L = \log\left(\sum_{i=1}^{|V|} \exp(u_i^T v_c)\right) - u_o^T v_c$$

Training

Differentiating with respect to v_c:

$$\frac{\partial L}{\partial v_c} = \frac{\left(\sum_{i=1}^{|V|} \exp(u_i^T v_c)\right)'}{\sum_{j=1}^{|V|} \exp(u_j^T v_c)} - u_o = \frac{\sum_{i=1}^{|V|} \exp(u_i^T v_c) \cdot (u_i^T v_c)'}{\sum_{j=1}^{|V|} \exp(u_j^T v_c)} - u_o = \frac{\sum_{i=1}^{|V|} \exp(u_i^T v_c) \cdot u_i}{\sum_{j=1}^{|V|} \exp(u_j^T v_c)} - u_o$$

Recognizing the softmax probability:

$$p(w_i \mid w_c) = \frac{\exp(u_i^T v_c)}{\sum_{j=1}^{|V|} \exp(u_j^T v_c)}$$

we obtain:

$$\frac{\partial L}{\partial v_c} = \sum_{i=1}^{|V|} p(w_i \mid w_c)\, u_i \; - \; u_o$$

Complexity:

$$\boxed{O(|V|)}$$

Approximate Training

Why approximate training?

Fully Softmax

$$O(|\mathcal{V}|)$$

For a typical Word2Vec with a vocabulary size of $|V| = 3{,}000{,}000$:

- $\approx 3{,}000{,}000$ dot products per update
- $\approx 3{,}000{,}000$ gradient updates

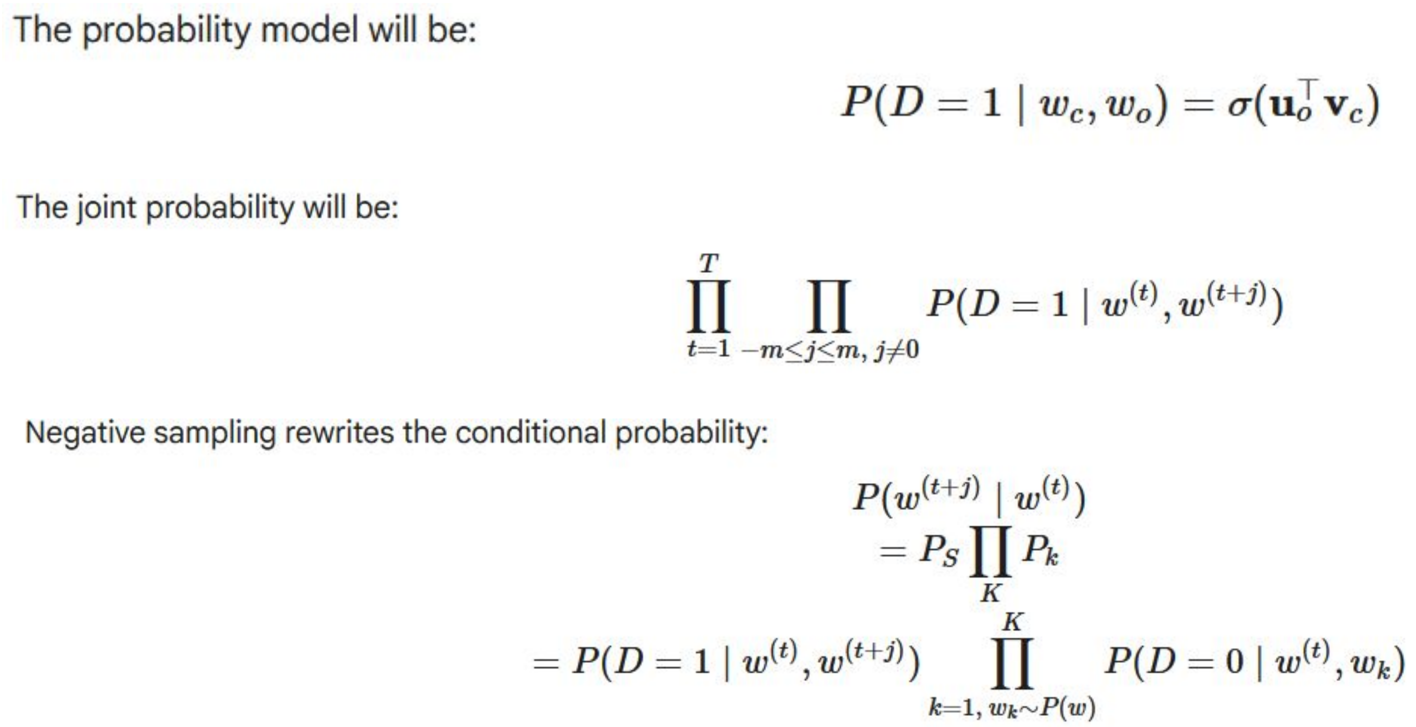
Negative Sampling

$O(k)$ where typically $k = 5$ to 10 15.

Hierarchical Softmax

$O(\log_2 |V|)$ $O(\log_2 |3,000,000|) \approx 22$

Negative Sampling in Skip-Gram Model



The probability model will be:

$$P(D = 1 \mid w_c, w_o) = \sigma(\mathbf{u}_o^\top \mathbf{v}_c)$$

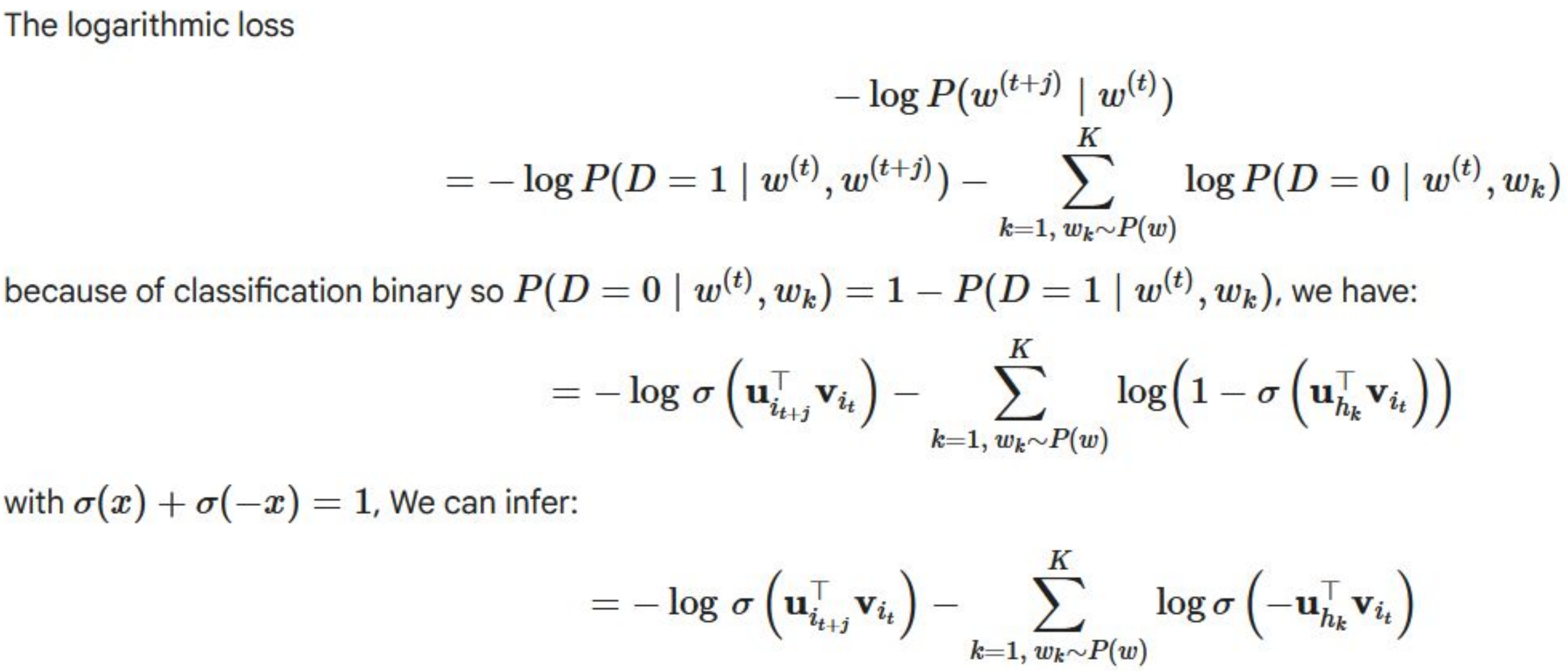
The joint probability will be:

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} P(D = 1 \mid w^{(t)}, w^{(t+j)})$$

Negative sampling rewrites the conditional probability:

$$P(w^{(t+j)} \mid w^{(t)})$$
$$= P_S \prod_K P_k$$
$$= P(D = 1 \mid w^{(t)}, w^{(t+j)}) \prod_{k=1,\ w_k \sim P(w)}^{K} P(D = 0 \mid w^{(t)}, w_k)$$

Negative Sampling in Skip-Gram Model



The logarithmic loss

$$-\log P(w^{(t+j)} \mid w^{(t)})$$

$$= -\log P(D=1 \mid w^{(t)}, w^{(t+j)}) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log P(D=0 \mid w^{(t)}, w_k)$$

because of classification binary so $P(D=0 \mid w^{(t)}, w_k) = 1 - P(D=1 \mid w^{(t)}, w_k)$, we have:

$$= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^\top \mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log\left(1 - \sigma\left(\mathbf{u}_{h_k}^\top \mathbf{v}_{i_t}\right)\right)$$

with $\sigma(x) + \sigma(-x) = 1$, We can infer:

$$= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^\top \mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log \sigma\left(-\mathbf{u}_{h_k}^\top \mathbf{v}_{i_t}\right)$$

Negative Sampling in Continuous Bag-of-words Model

The probability model will be:

$$P(D=1 \mid w_c, w_{o_1}, \ldots, w_{o_{2m}}) = \sigma(\mathbf{u}_o^\top \bar{\mathbf{v}}_o)$$

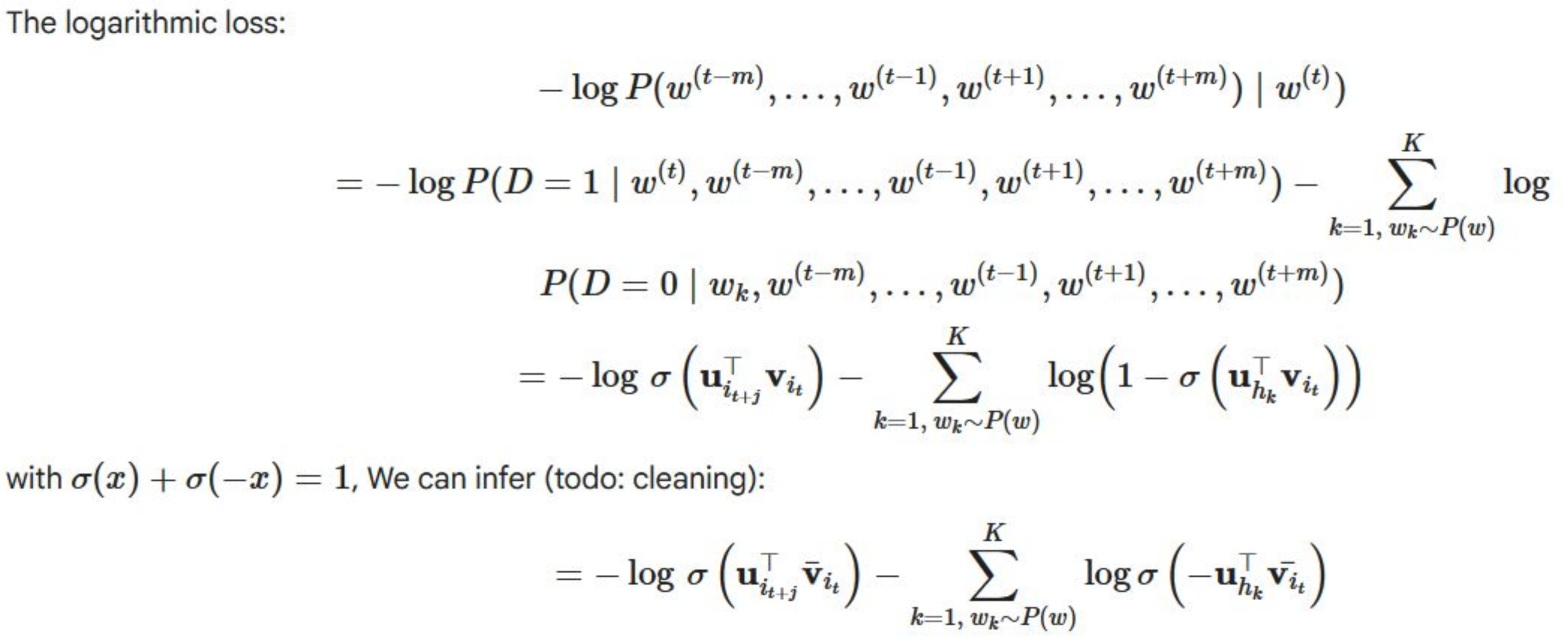
The joint probability will be:

$$\prod_{t=1}^{T} P(D=1 \mid w^{(t)}, w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$$

$$P(w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)} \mid w^{(t)}) \\ = P_S \prod_{K} P_k$$

$$= P(D=1 \mid w^{(t)}, w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)}) \prod_{k=1,\; w_k \sim P(w)}^{K} P(D=0 \mid w_k, w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$$

Negative Sampling in Continuous Bag-of-words Model



The logarithmic loss:

$$-\log P(w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)} \mid w^{(t)})$$

$$= -\log P(D=1 \mid w^{(t)}, w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)}) - \sum_{k=1,\, w_k \sim P(w)}^{K} \log$$

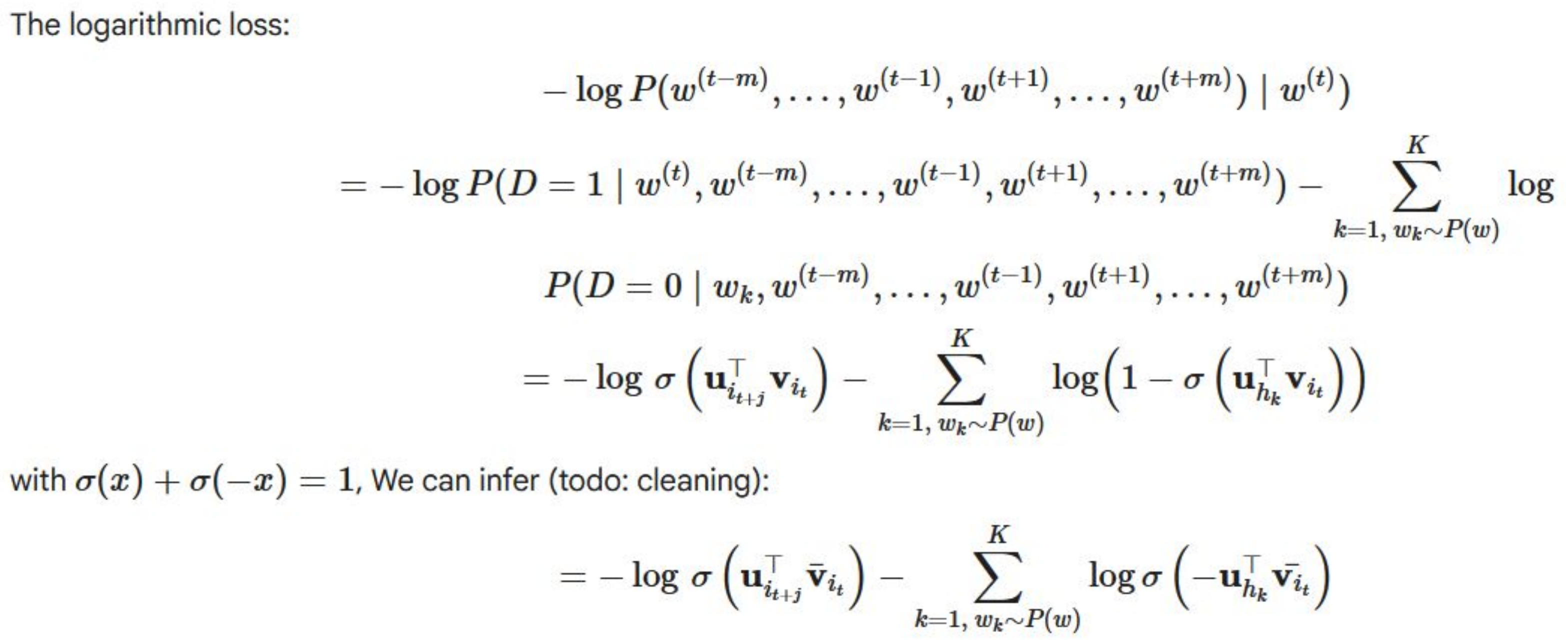
$$P(D=0 \mid w_k, w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$$

$$= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^\top \mathbf{v}_{i_t}\right) - \sum_{k=1,\, w_k \sim P(w)}^{K} \log\left(1 - \sigma\left(\mathbf{u}_{h_k}^\top \mathbf{v}_{i_t}\right)\right)$$

with $\sigma(x) + \sigma(-x) = 1$, We can infer (todo: cleaning):

$$= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^\top \bar{\mathbf{v}}_{i_t}\right) - \sum_{k=1,\, w_k \sim P(w)}^{K} \log \sigma\left(-\mathbf{u}_{h_k}^\top \bar{\mathbf{v}}_{i_t}\right)$$

Negative Sampling in Continuous Bag-of-words Model



The logarithmic loss:

$$-\log P(w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)} \mid w^{(t)})$$

$$= -\log P(D=1 \mid w^{(t)}, w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)}) - \sum_{k=1,\, w_k \sim P(w)}^{K} \log$$

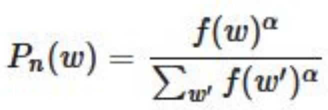
$$P(D=0 \mid w_k, w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$$

$$= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^{\top} \mathbf{v}_{i_t}\right) - \sum_{k=1,\, w_k \sim P(w)}^{K} \log\left(1 - \sigma\left(\mathbf{u}_{h_k}^{\top} \mathbf{v}_{i_t}\right)\right)$$

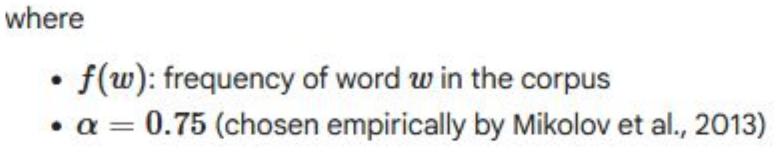
with $\sigma(x) + \sigma(-x) = 1$, We can infer (todo: cleaning):

$$= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^{\top} \bar{\mathbf{v}}_{i_t}\right) - \sum_{k=1,\, w_k \sim P(w)}^{K} \log \sigma\left(-\mathbf{u}_{h_k}^{\top} \bar{\mathbf{v}}_{i_t}\right)$$

How can we sample noise words in negative sampling?



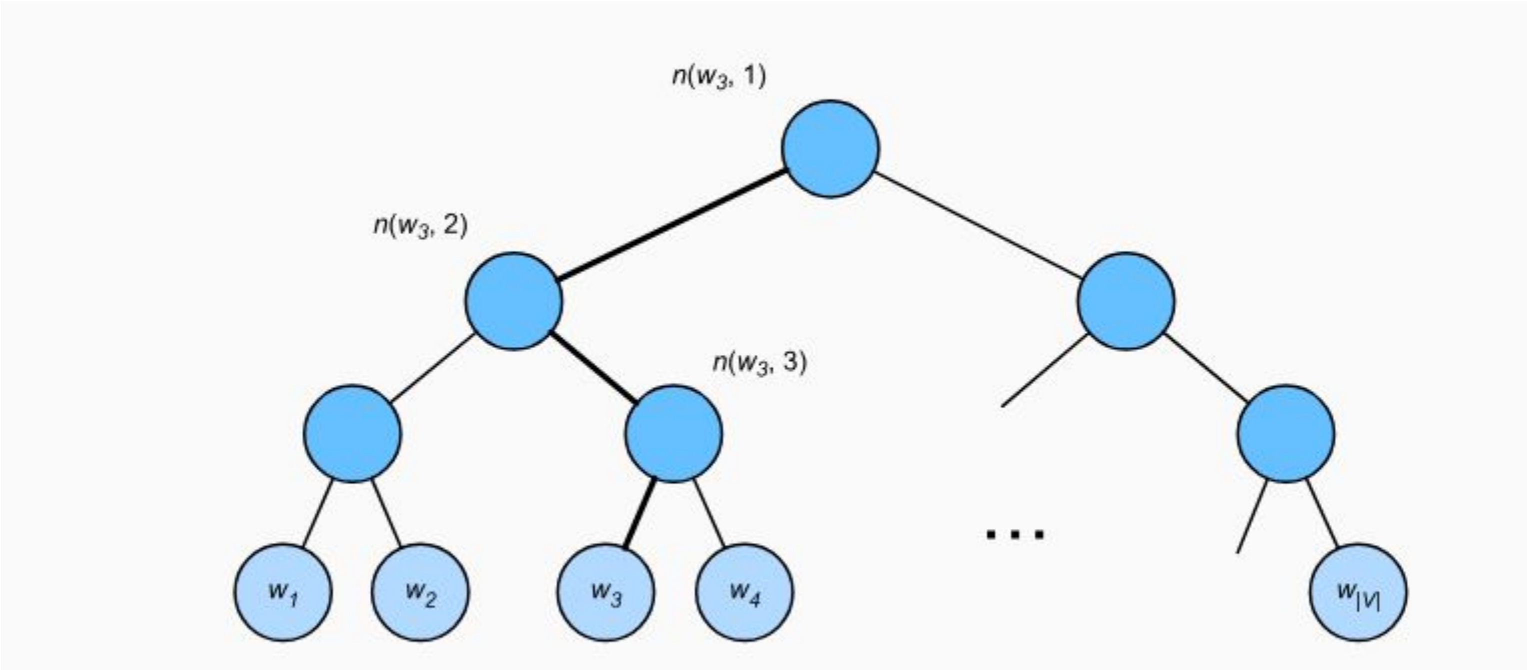
$$P_n(w) = \frac{f(w)^\alpha}{\sum_{w'} f(w')^\alpha}$$



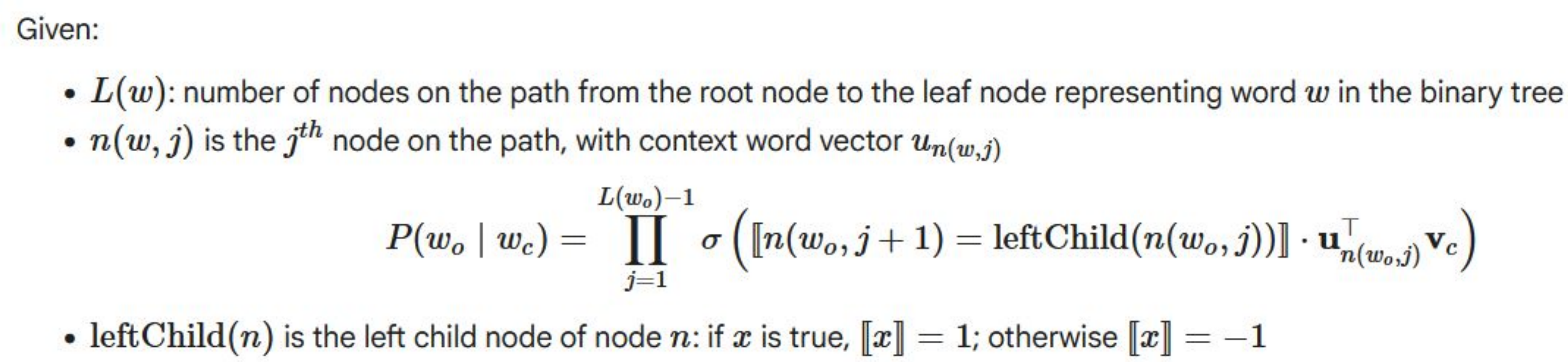
where

- $f(w)$: frequency of word w in the corpus
- $\alpha = 0.75$ (chosen empirically by Mikolov et al., 2013)

Hierarchical Softmax in Skip Gram Model



Hierarchical Softmax in Skip Gram Model



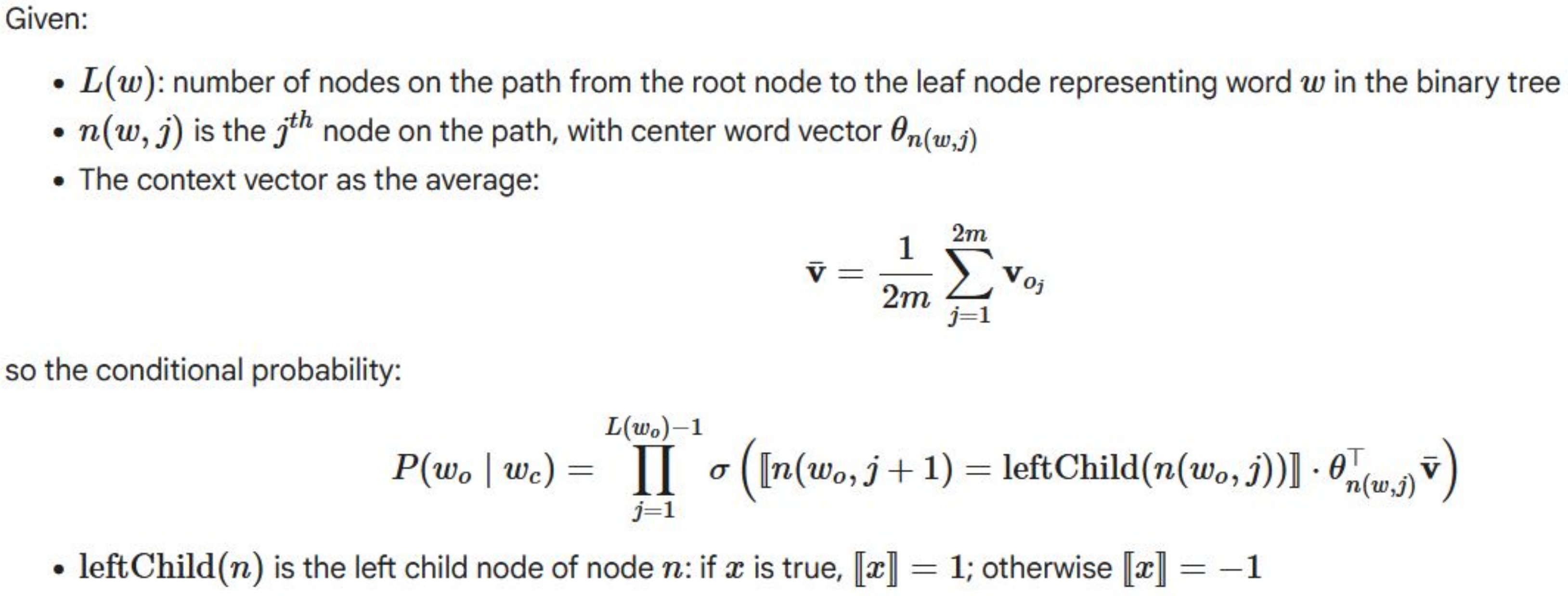
Given:

- $L(w)$: number of nodes on the path from the root node to the leaf node representing word w in the binary tree
- $n(w, j)$ is the j^{th} node on the path, with context word vector $u_{n(w,j)}$

$$P(w_o \mid w_c) = \prod_{j=1}^{L(w_o)-1} \sigma\left([\![n(w_o, j+1) = \text{leftChild}(n(w_o, j))]\!] \cdot \mathbf{u}_{n(w_o,j)}^\top \mathbf{v}_c \right)$$

- $\text{leftChild}(n)$ is the left child node of node n: if x is true, $[\![x]\!] = 1$; otherwise $[\![x]\!] = -1$

Hierarchical Softmax in Continuous Bag-of-words Model



Given:

- $L(w)$: number of nodes on the path from the root node to the leaf node representing word w in the binary tree
- $n(w, j)$ is the j^{th} node on the path, with center word vector $\theta_{n(w,j)}$
- The context vector as the average:

$$\bar{\mathbf{v}} = \frac{1}{2m} \sum_{j=1}^{2m} \mathbf{v}_{o_j}$$

so the conditional probability:

$$P(w_o \mid w_c) = \prod_{j=1}^{L(w_o)-1} \sigma\left([\![n(w_o, j+1) = \text{leftChild}(n(w_o, j))]\!] \cdot \theta_{n(w,j)}^\top \bar{\mathbf{v}} \right)$$

- $\text{leftChild}(n)$ is the left child node of node n: if x is true, $[\![x]\!] = 1$; otherwise $[\![x]\!] = -1$

Exercise

Verify that (15.2.9) holds

$$\sum_{w \in \mathcal{V}} P(w \mid w_c) = 1.$$

At an internal node n, we have two children:

- Left child L
- Right child R

and $t_n = u_n^\top v_c$

Then:

- Probability of going left:

$$p_L = \sigma(t_n)$$

- Probability of going right:

$$p_R = \sigma(-t_n)$$

and

$$\sigma(t_n) + \sigma(-t_n) = 1$$

The probability mass of a subtree equals the probability of reaching its root

Thanks for watching and listening!