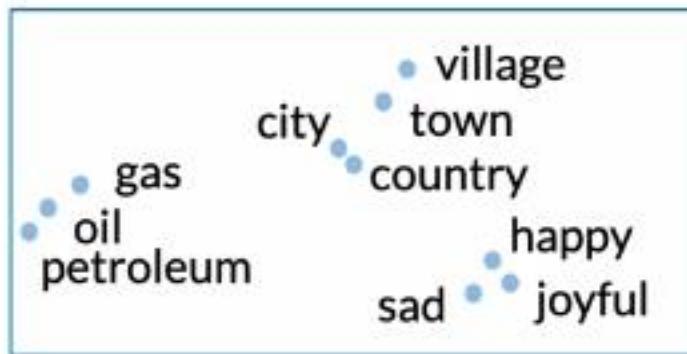


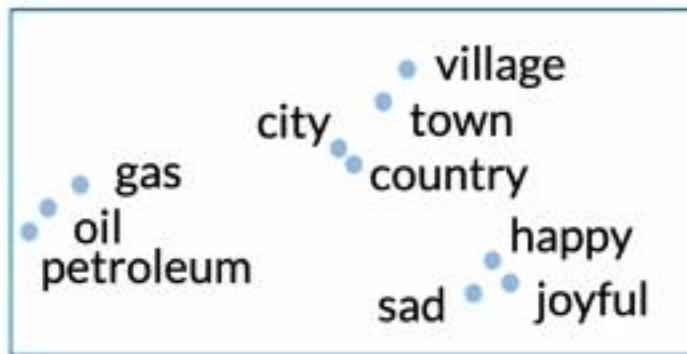
Some basic applications of word embeddings

Some basic applications of word embeddings



Semantic analogies
and similarity

Some basic applications of word embeddings



Semantic analogies
and similarity



Sentiment analysis

Some basic applications of word embeddings



Semantic analogies
and similarity



Sentiment analysis



Classification of
customer feedback

Advanced applications of word embeddings



Machine translation

Advanced applications of word embeddings



Machine translation



Information extraction

Advanced applications of word embeddings



Machine translation



Information extraction



Question answering

Learning objectives

- Identify the key concepts of word representations

Learning objectives

- Identify the key concepts of word representations
- Generate word embeddings

Learning objectives

- Identify the key concepts of word representations
- Generate word embeddings
- Prepare text for machine learning

Learning objectives

- Identify the key concepts of word representations
- Generate word embeddings
- Prepare text for machine learning
- Implement the continuous bag-of-words model

Learning objectives

Prerequisite: neural networks

- Identify the key concepts of word representations
- Generate word embeddings
- Prepare text for machine learning
- Implement the continuous bag-of-words model

Integers

Word	Number
a	1
able	2
about	3
...	...
hand	615
...	...
happy	621
...	...
zebra	1000

Integers

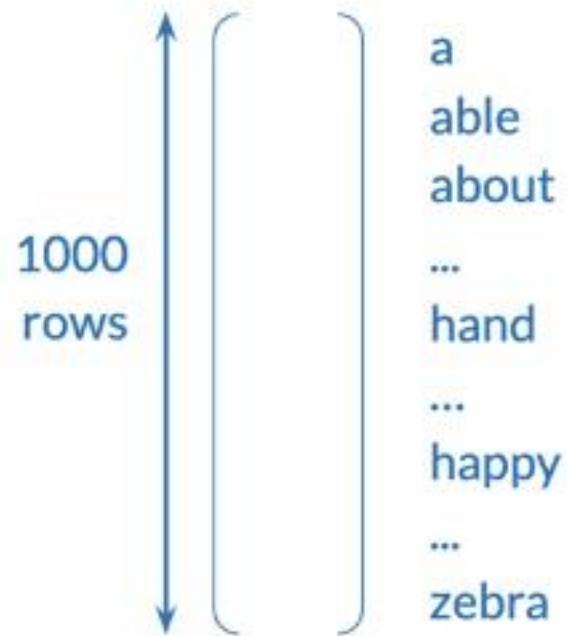
- + Simple
- Ordering: little semantic sense

Integers

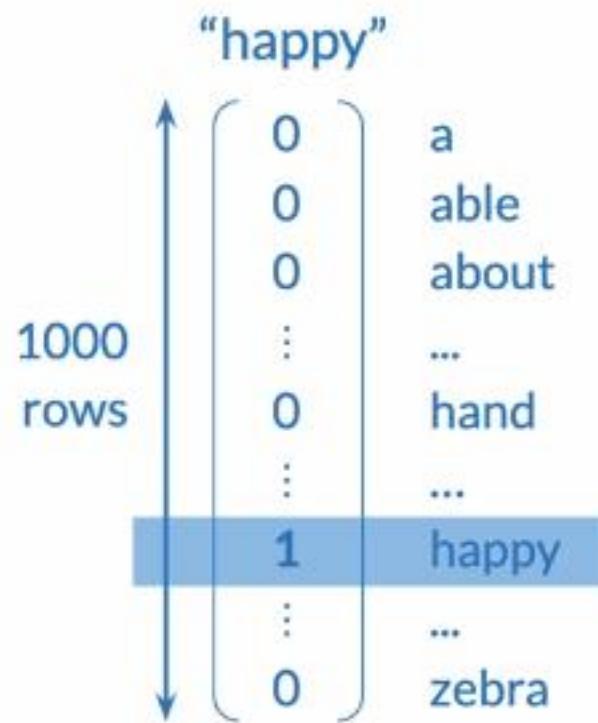
- + Simple
- Ordering: little semantic sense

hand < happy < zebra
615 ?! 621 ?! 1000

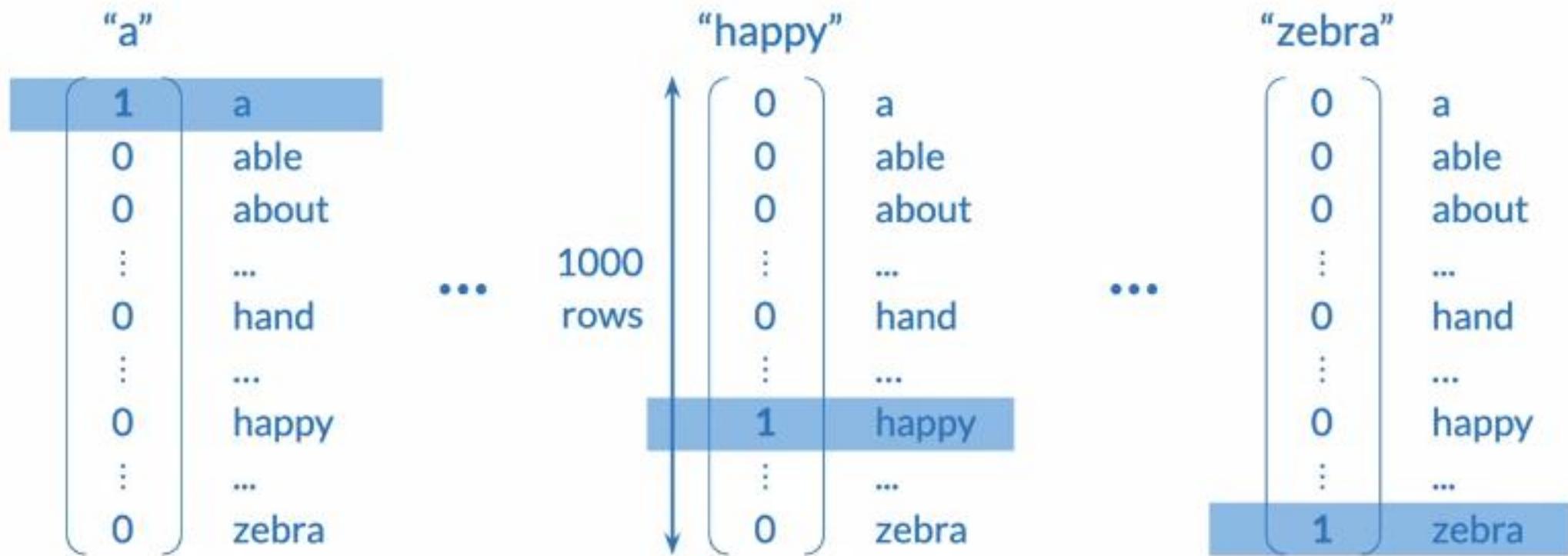
One-hot vectors



One-hot vectors



One-hot vectors



One-hot vectors

Word	Number
a	1
able	2
about	3
...	...
hand	615
...	...
happy	621
...	...
zebra	1000

“happy”

$$\begin{matrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \\ \vdots & \vdots \\ 615 & 0 \\ \vdots & \vdots \\ 621 & 1 \\ \vdots & \vdots \\ 1000 & 0 \end{matrix}$$

a
able
about
...
hand
...
happy
...
zebra

One-hot vectors

Word	Number		"happy"	
a	1		1	0
able	2		2	0
about	3		3	0
...
hand	615		615	0
...
happy	621	→	621	1
...
zebra	1000		1000	0

One-hot vectors

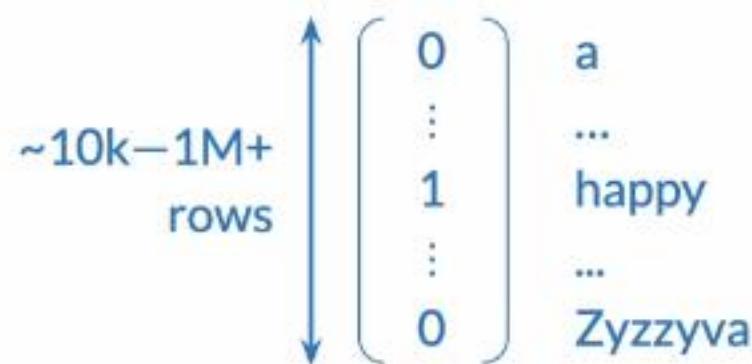
- + Simple
- + No implied ordering

One-hot vectors

- + Simple
- + No implied ordering
- Huge vectors

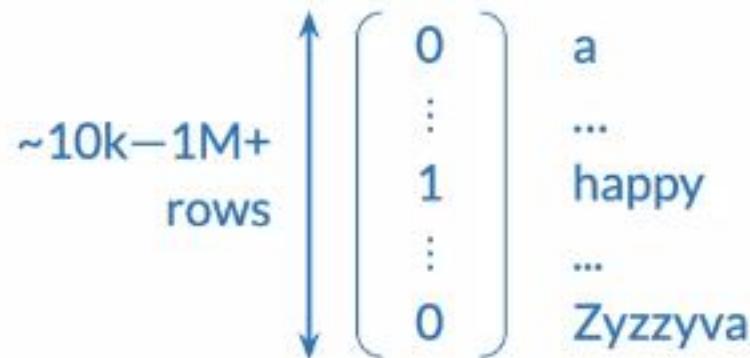
One-hot vectors

- + Simple
- + No implied ordering
- Huge vectors



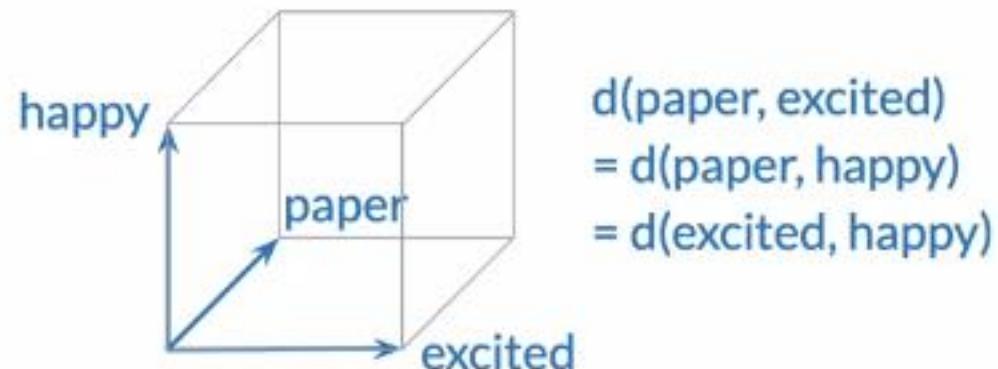
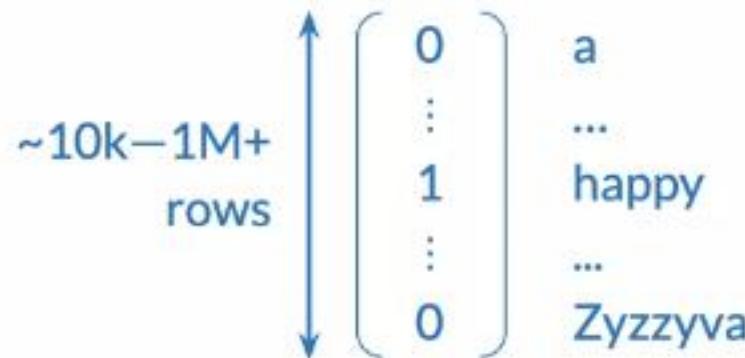
One-hot vectors

- + Simple
- + No implied ordering
- Huge vectors
- No embedded meaning



One-hot vectors

- + Simple
- + No implied ordering
- Huge vectors
- No embedded meaning

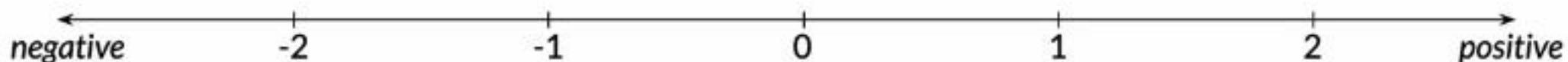




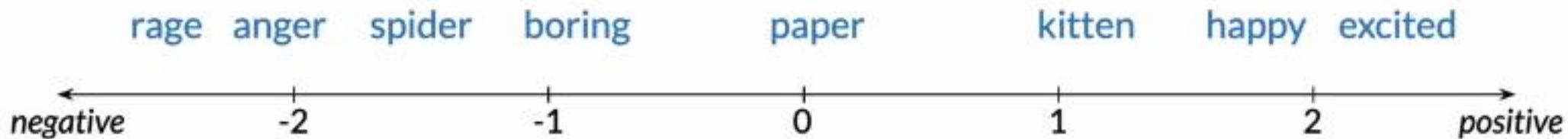
deeplearning.ai

Word Embeddings

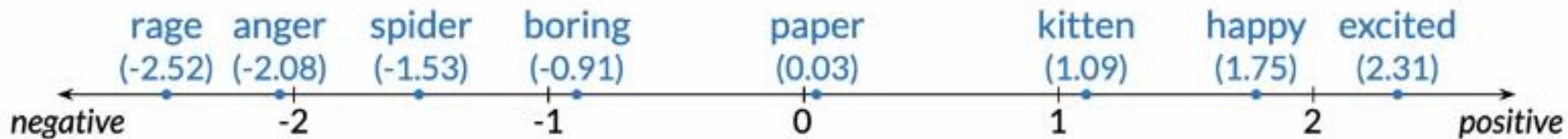
Meaning as vectors



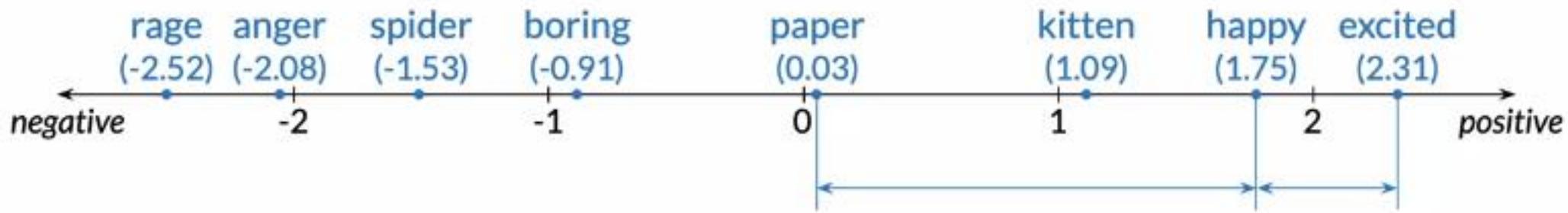
Meaning as vectors



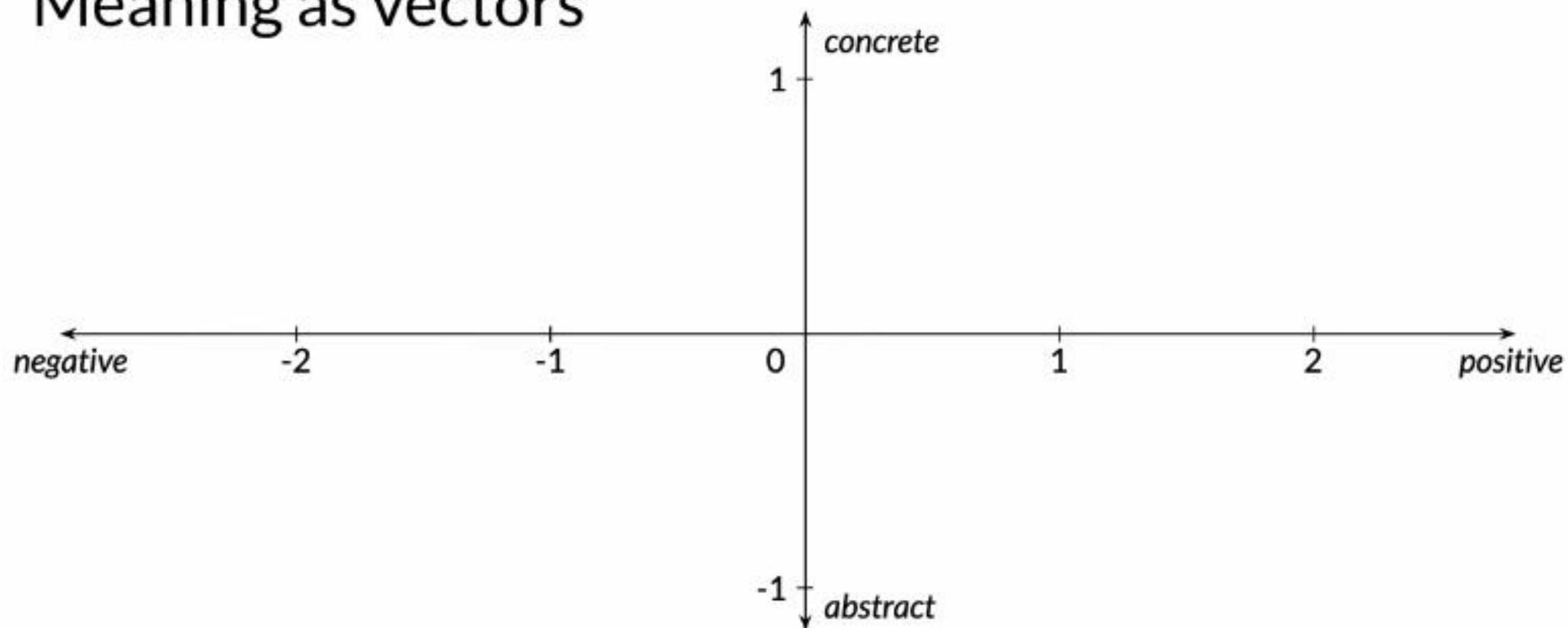
Meaning as vectors



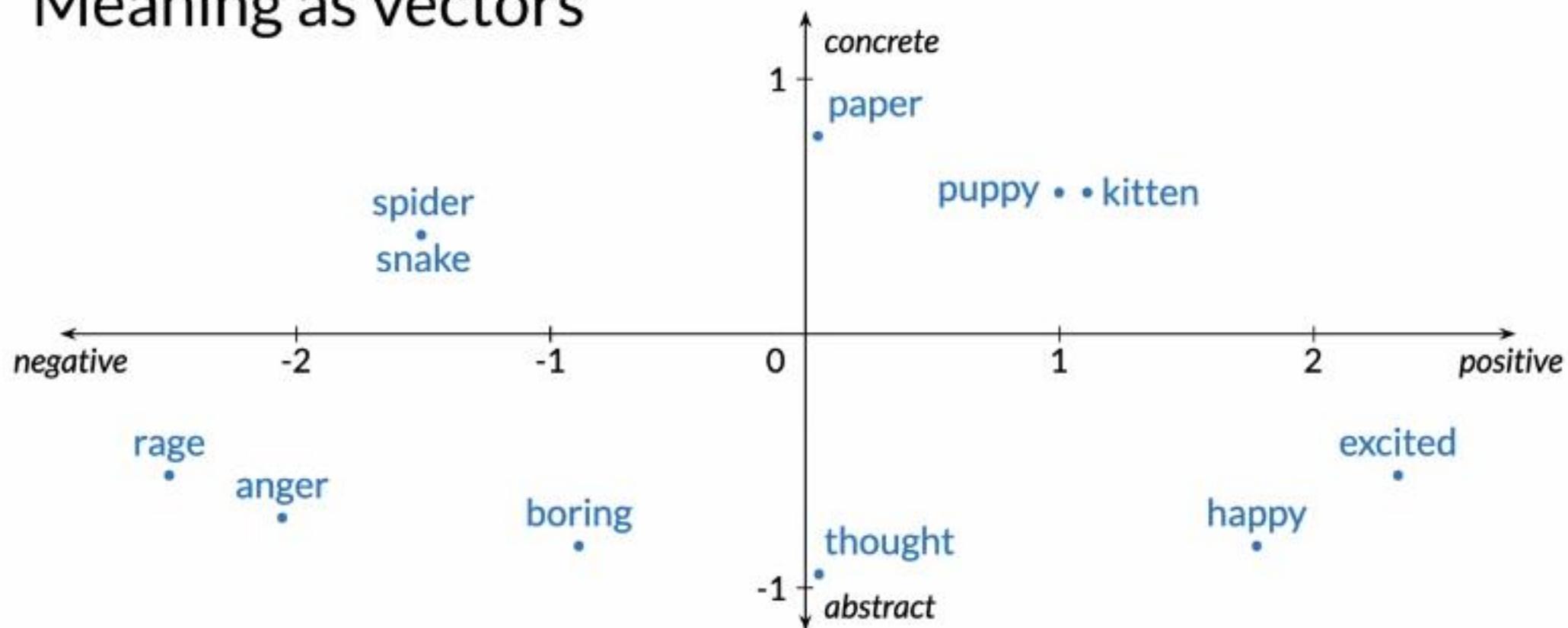
Meaning as vectors



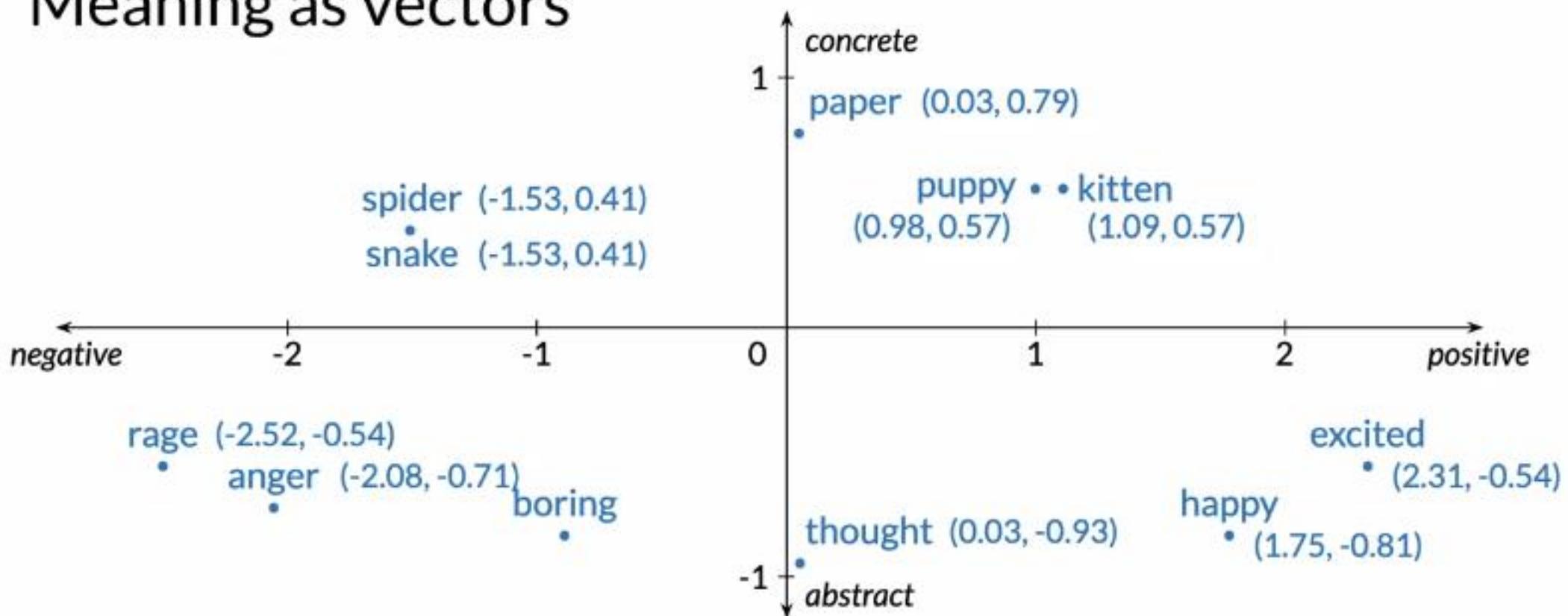
Meaning as vectors



Meaning as vectors



Meaning as vectors



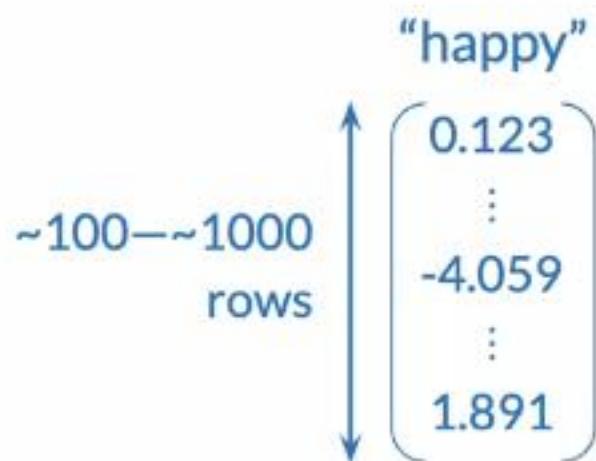
Word embedding vectors

Word embedding vectors

- + Low dimension

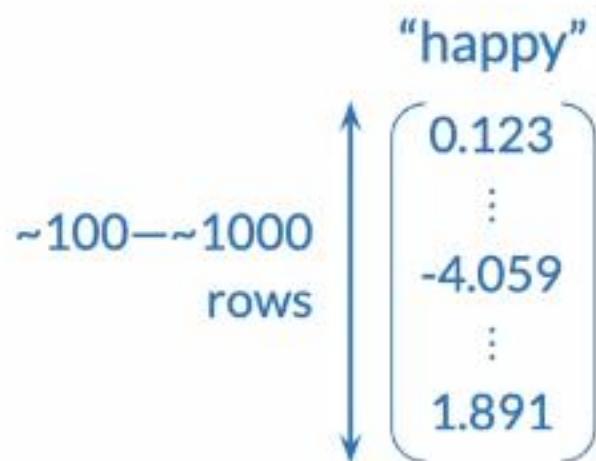
Word embedding vectors

- + Low dimension



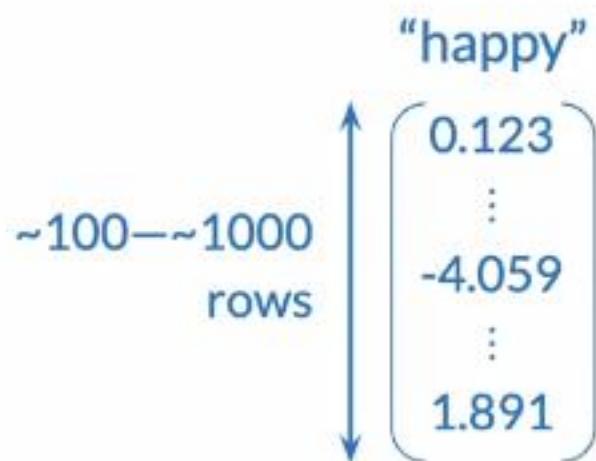
Word embedding vectors

- + Low dimension
- + Embed meaning



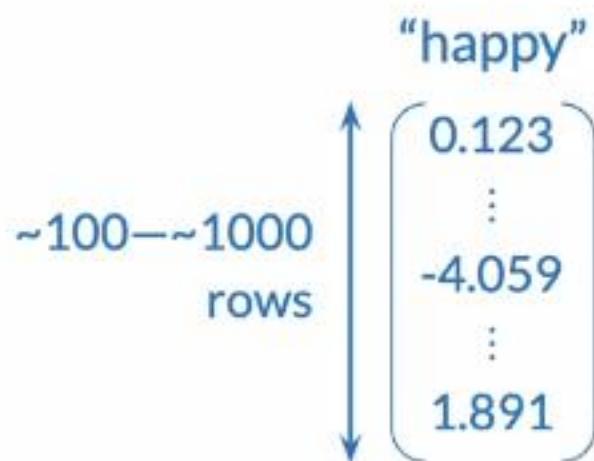
Word embedding vectors

- + Low dimension
- + Embed meaning
 - e.g. semantic distance



Word embedding vectors

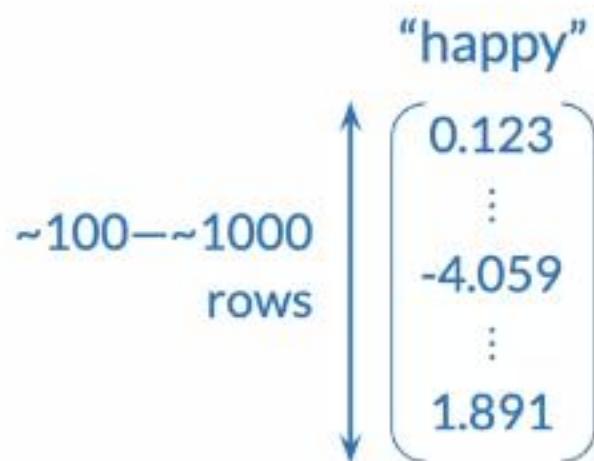
- + Low dimension
- + Embed meaning
 - e.g. semantic distance



forest = tree forest ≠ ticket

Word embedding vectors

- + Low dimension
- + Embed meaning
 - e.g. semantic distance



forest = tree forest \neq ticket

- e.g. analogies

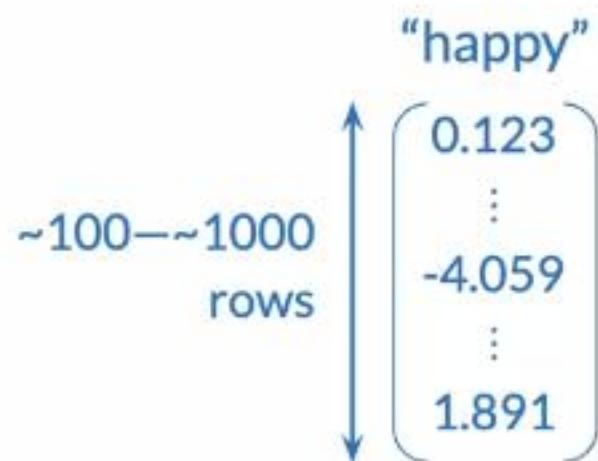
Word embedding vectors

- + Low dimension
- + Embed meaning
 - o e.g. semantic distance

forest \approx tree forest \neq ticket

- o e.g. analogies

Paris:France :: Rome:?



Terminology

integers

one-hot vectors

word embedding vectors

Terminology

integers

word vectors

one-hot vectors

word embedding vectors

Terminology

integers

word vectors

one-hot vectors

word embedding vectors

“word vectors”

word embeddings

Summary

- Words as integers
- Words as vectors
 - One-hot vectors
 - Word embedding vectors
- Benefits of word embeddings for NLP

Word embedding process

Corpus

Embedding method

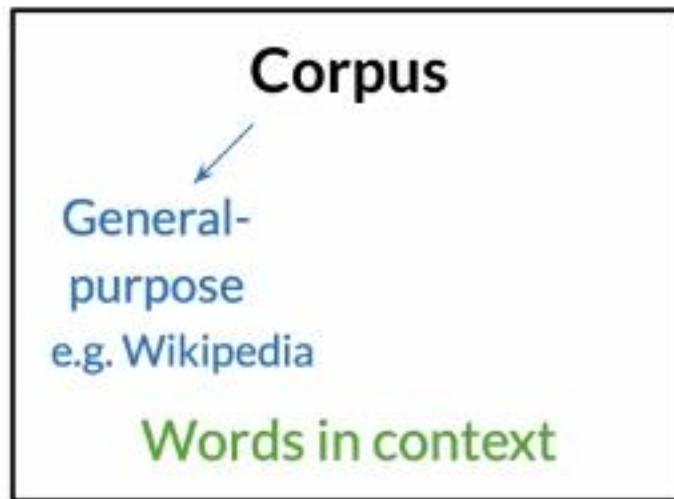
Word embedding process

Corpus

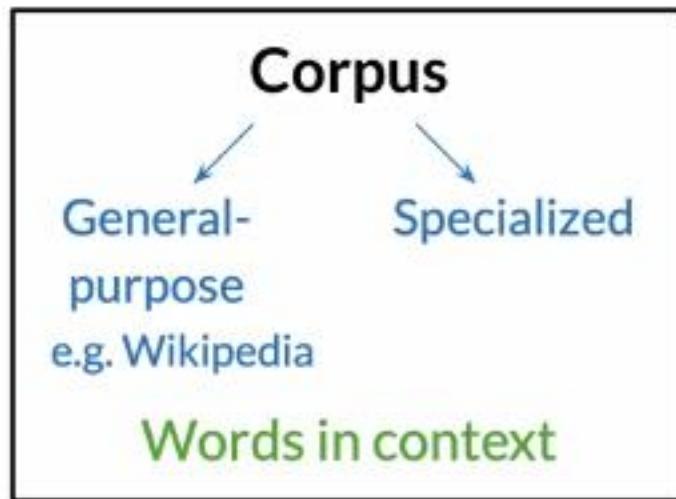
Words in context

Embedding method

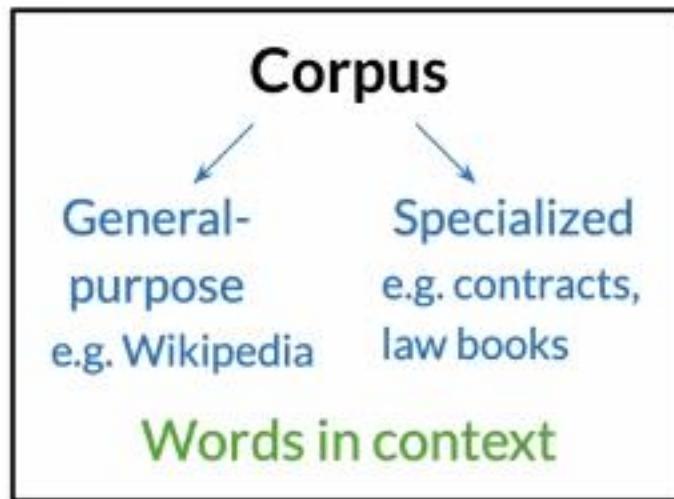
Word embedding process



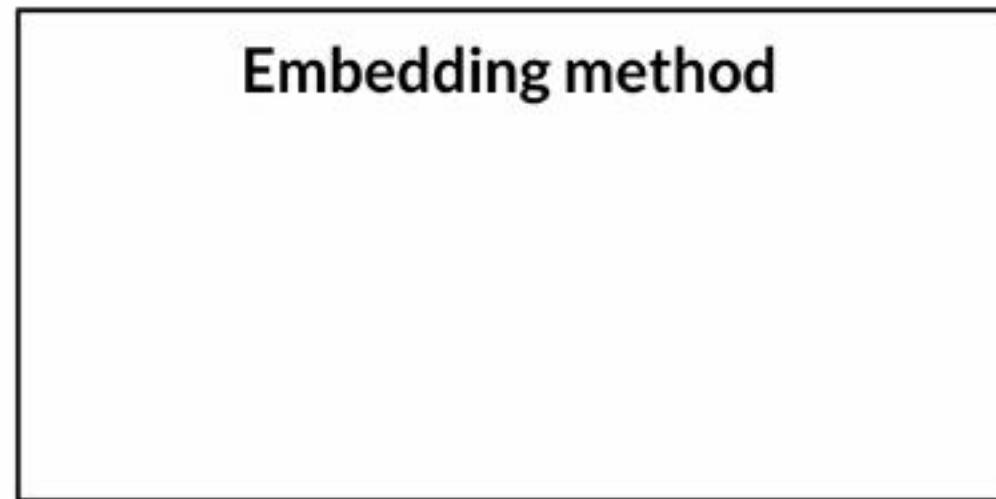
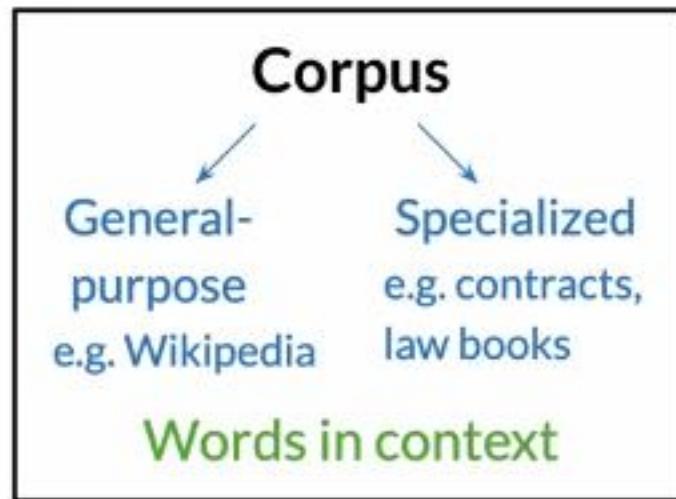
Word embedding process



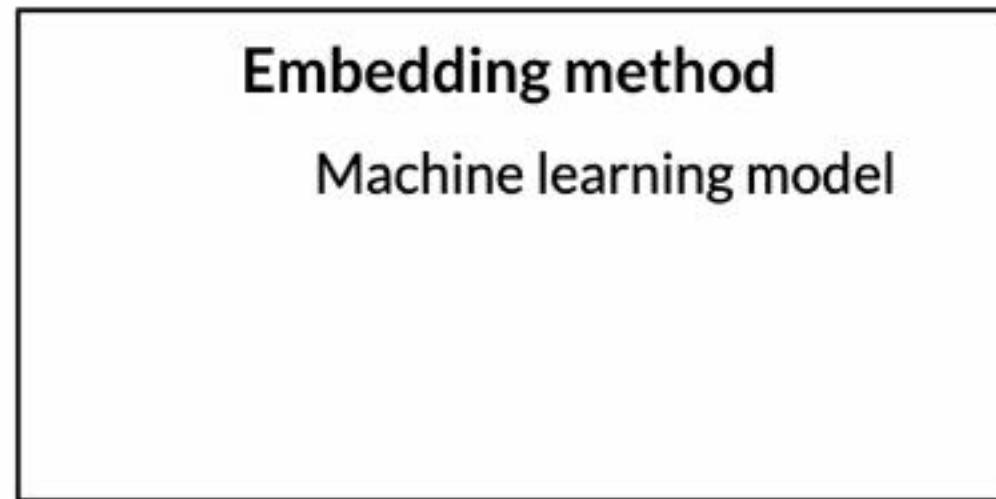
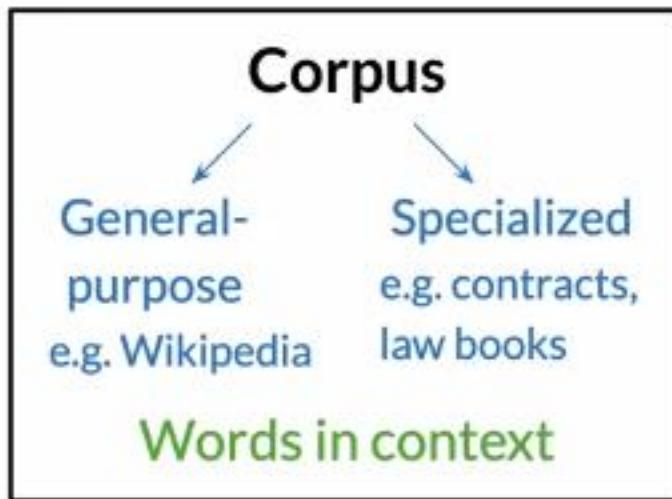
Word embedding process



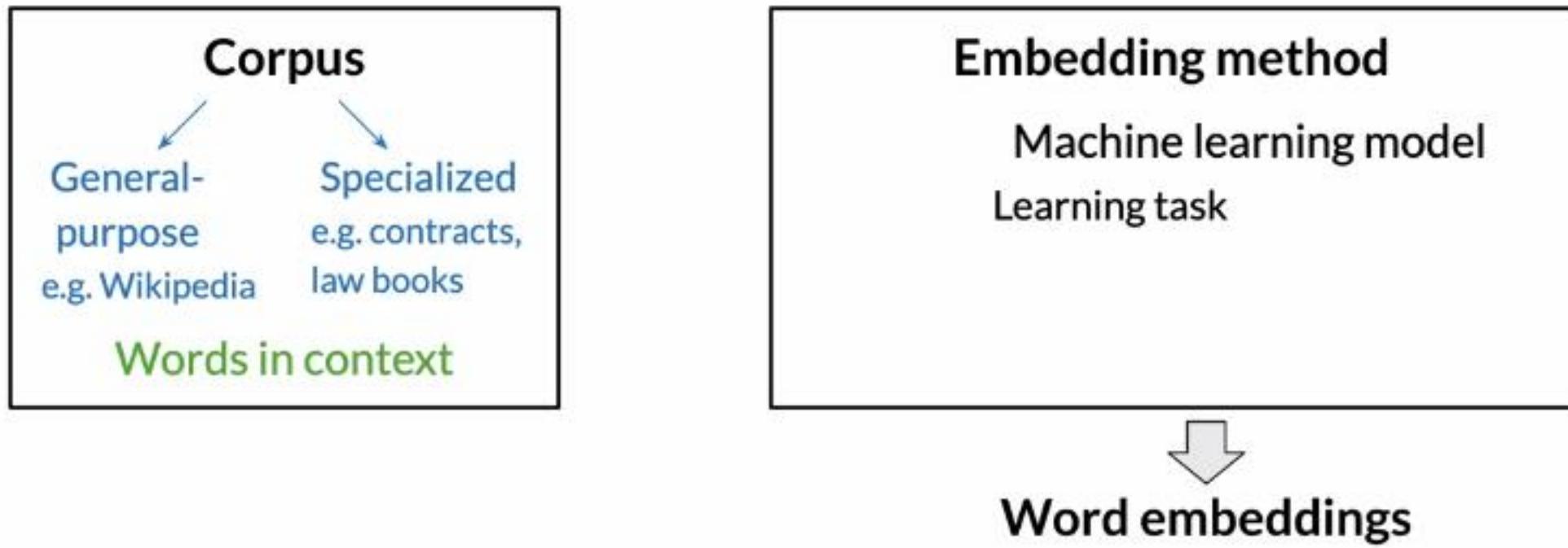
Word embedding process



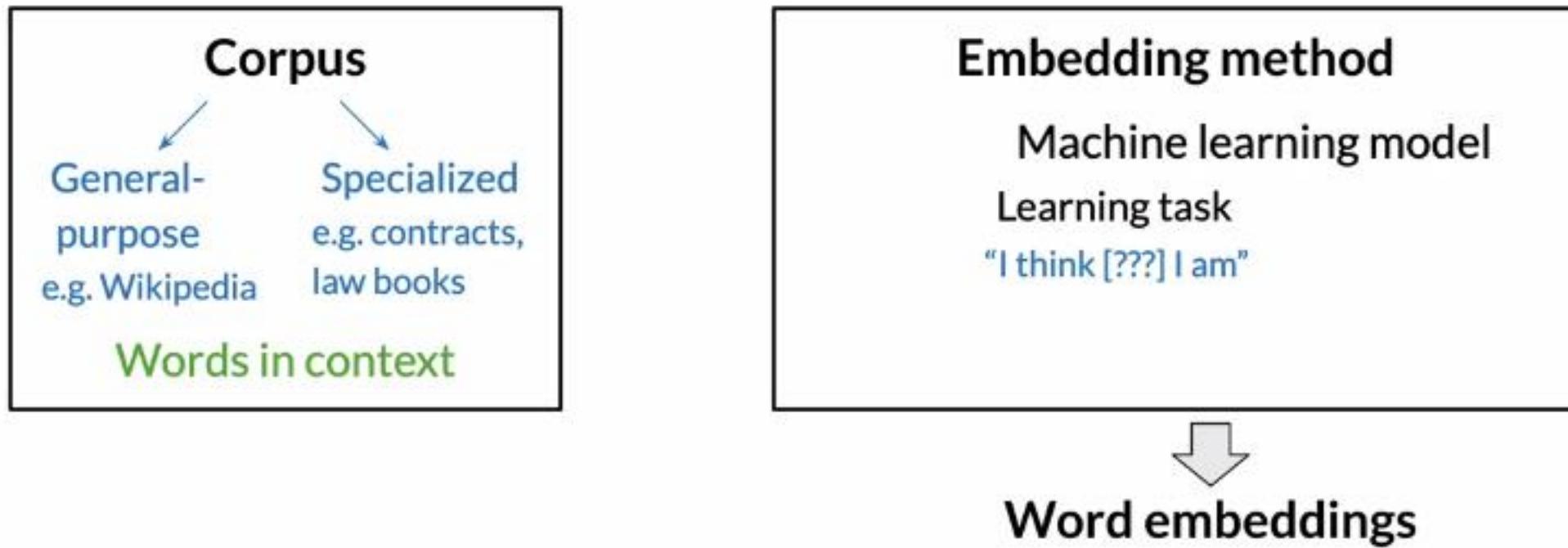
Word embedding process



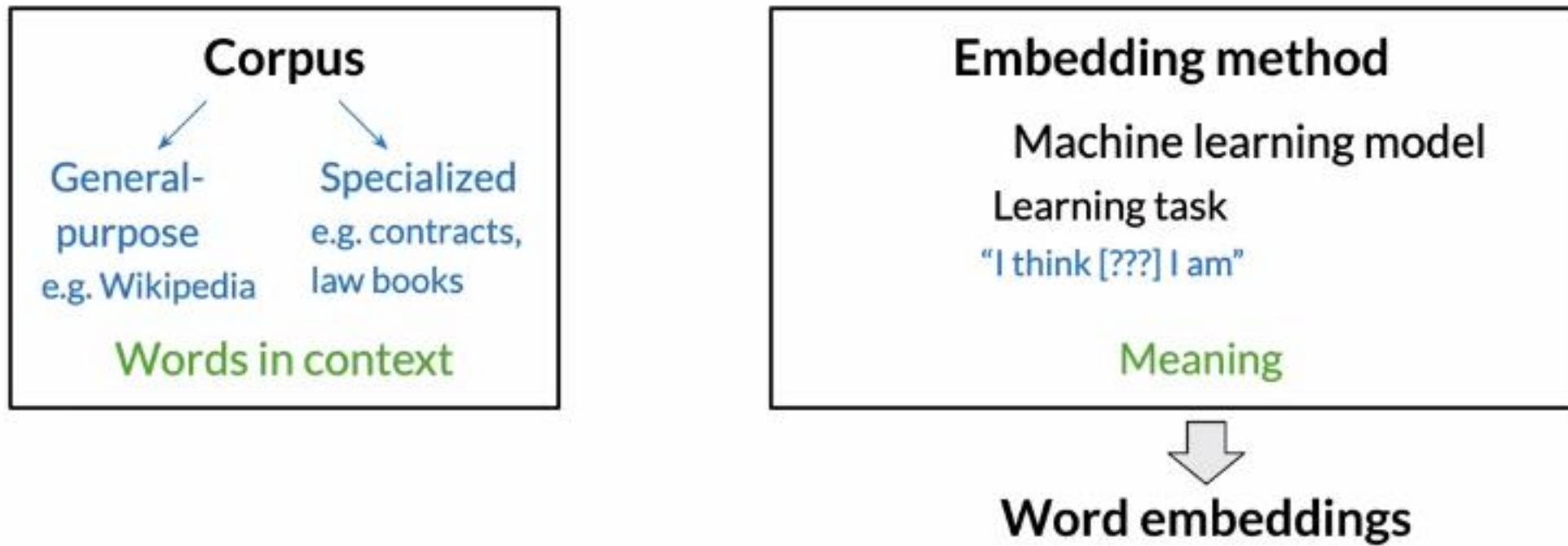
Word embedding process



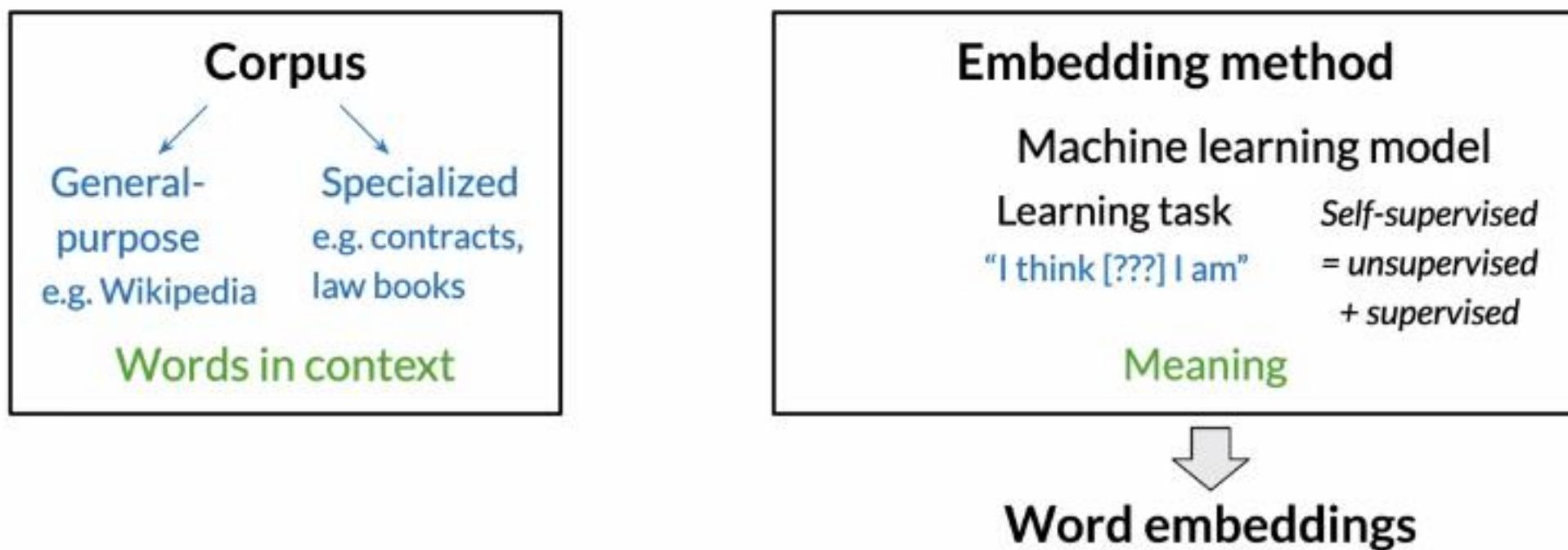
Word embedding process



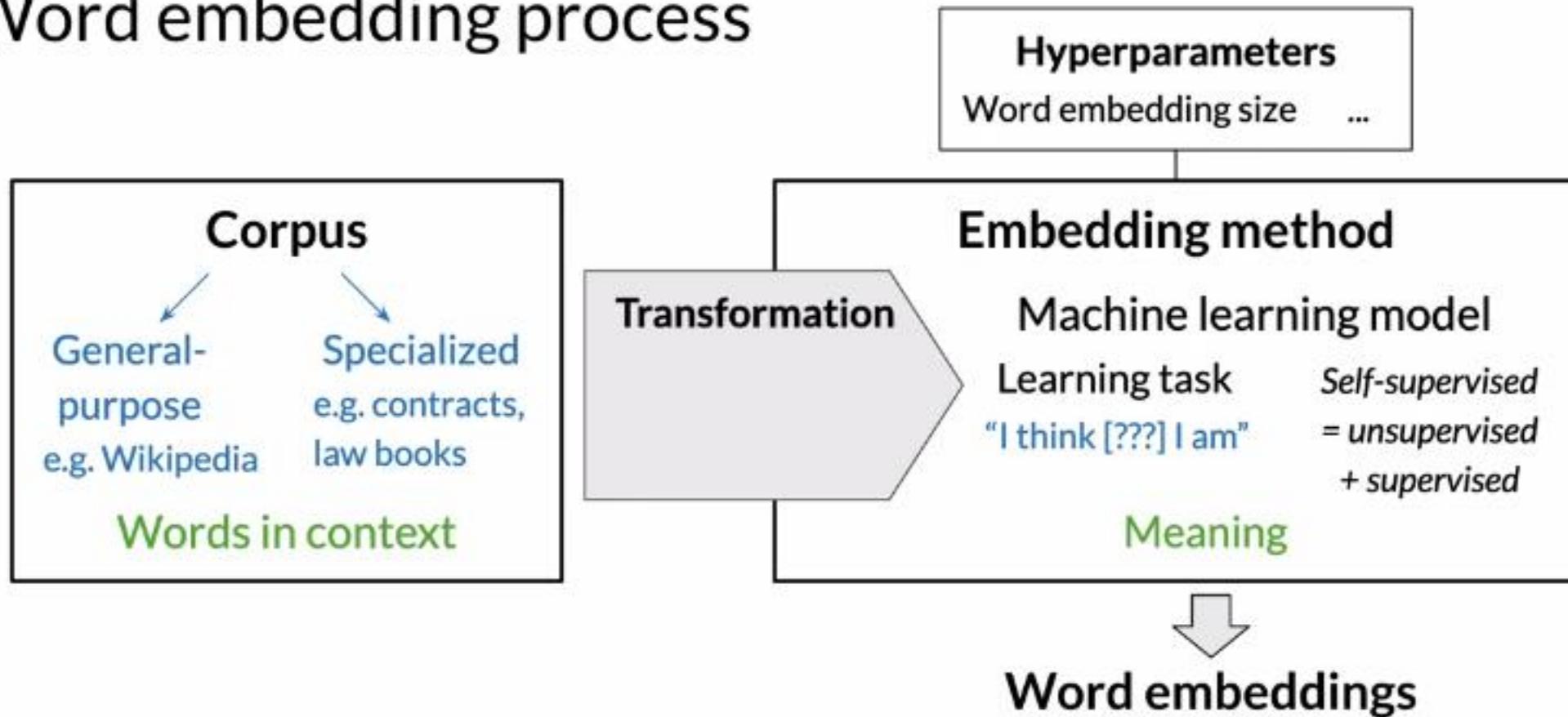
Word embedding process



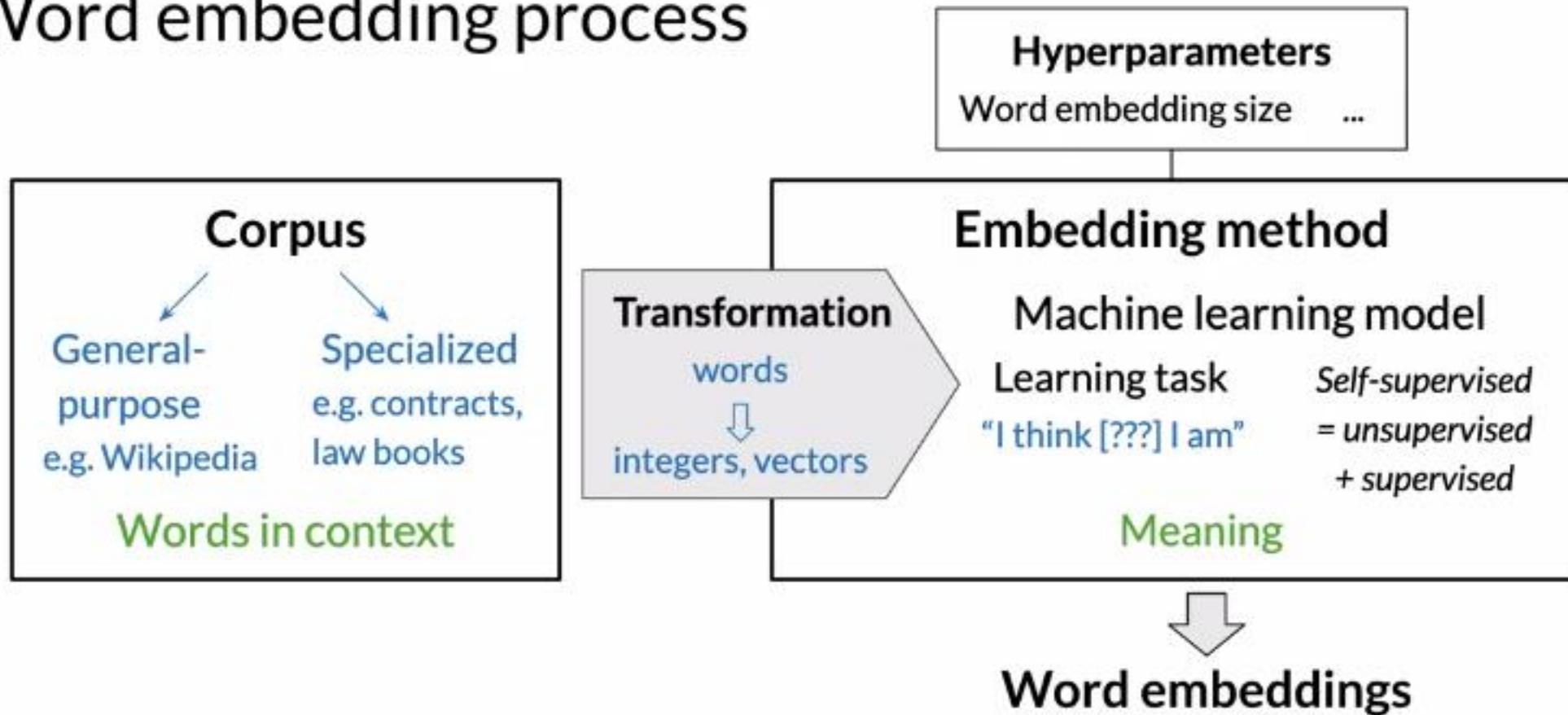
Word embedding process



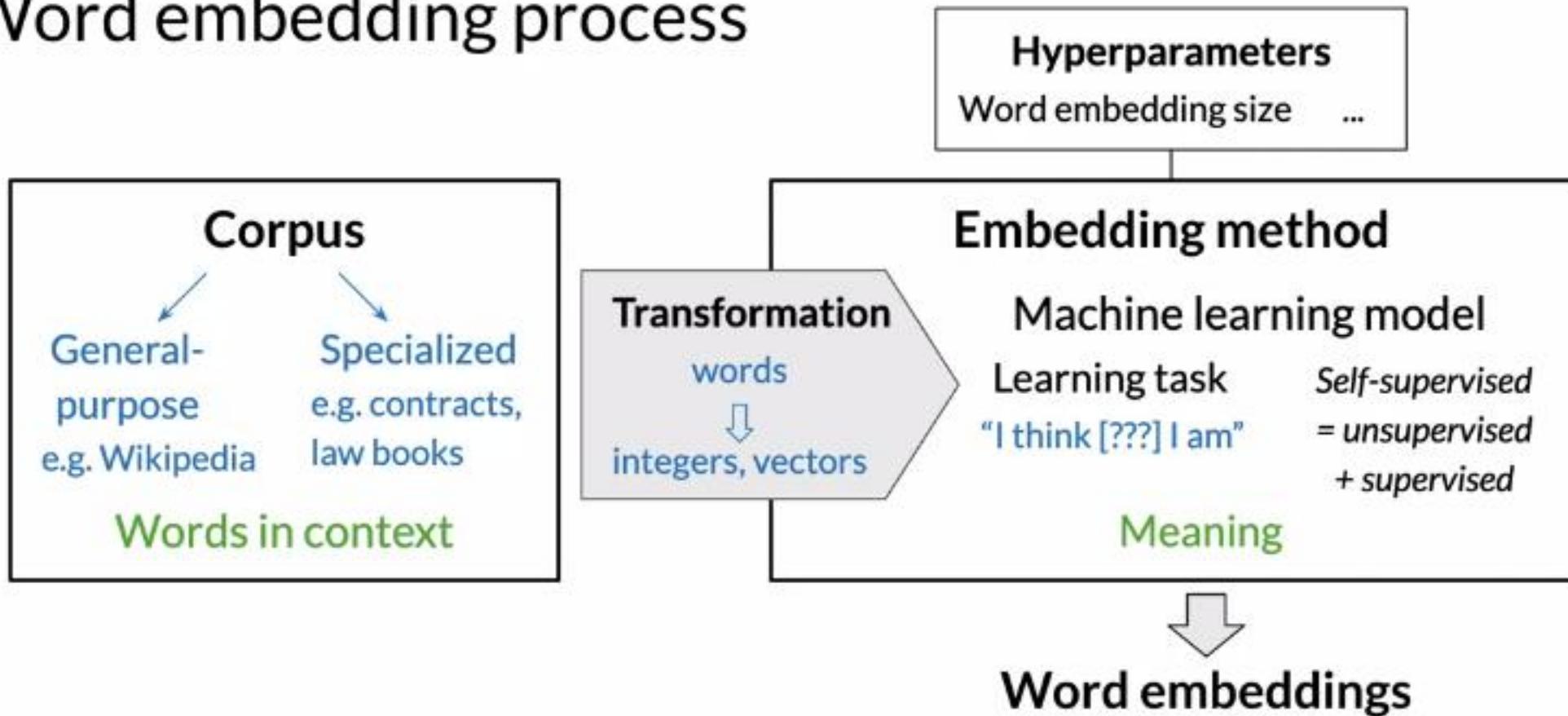
Word embedding process



Word embedding process



Word embedding process





deeplearning.ai

Word Embedding Methods

Basic word embedding methods

- word2vec (Google, 2013)
 - Continuous bag-of-words (CBOW)

Basic word embedding methods

- word2vec (Google, 2013)
 - Continuous bag-of-words (CBOW)
 - Continuous skip-gram / Skip-gram with negative sampling (SGNS)

Basic word embedding methods

- word2vec (Google, 2013)
 - Continuous bag-of-words (CBOW)
 - Continuous skip-gram / Skip-gram with negative sampling (SGNS)
- Global Vectors (GloVe) (Stanford, 2014)

Basic word embedding methods

- word2vec (Google, 2013)
 - Continuous bag-of-words (CBOW)
 - Continuous skip-gram / Skip-gram with negative sampling (SGNS)
- Global Vectors (GloVe) (Stanford, 2014)
- fastText (Facebook, 2016)
 - Supports out-of-vocabulary (OOV) words

Advanced word embedding methods

Deep learning, contextual embeddings

Advanced word embedding methods

Deep learning, contextual embeddings

- BERT (Google, 2018)

Advanced word embedding methods

Deep learning, contextual embeddings

- BERT (Google, 2018)
- ELMo (Allen Institute for AI, 2018)

Advanced word embedding methods

Deep learning, contextual embeddings

- BERT (Google, 2018)
- ELMo (Allen Institute for AI, 2018)
- GPT-2 (OpenAI, 2018)

Advanced word embedding methods

Deep learning, contextual embeddings

- BERT (Google, 2018)
 - ELMo (Allen Institute for AI, 2018)
 - GPT-2 (OpenAI, 2018)
- 
- Tunable pre-trained
models available

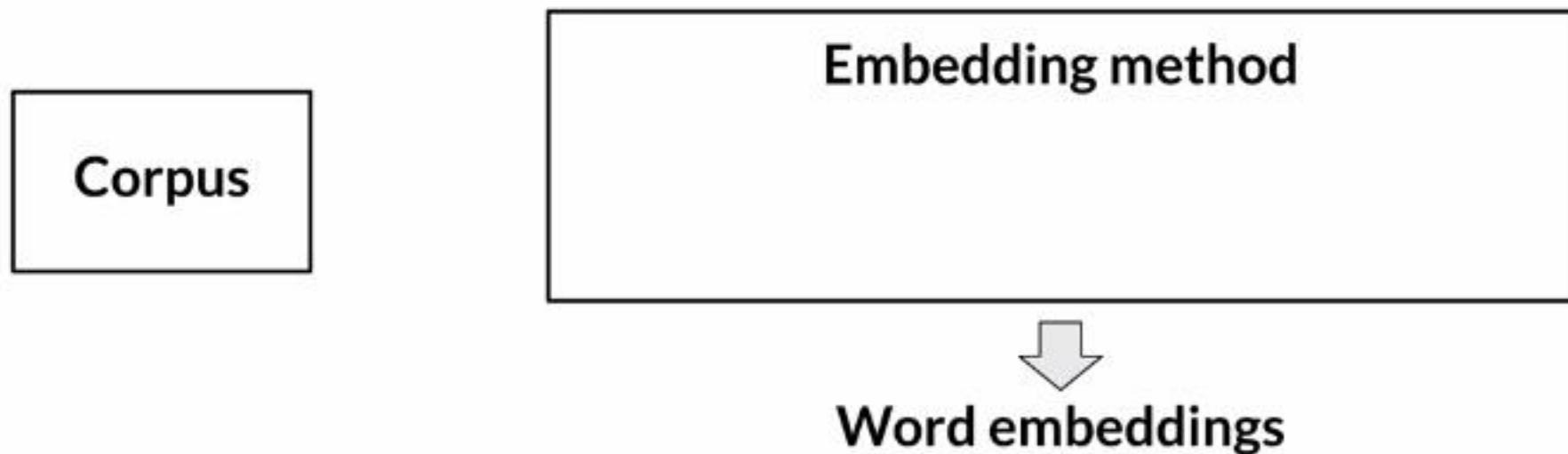
Continuous bag-of-words word embedding process

Continuous bag-of-words word embedding process

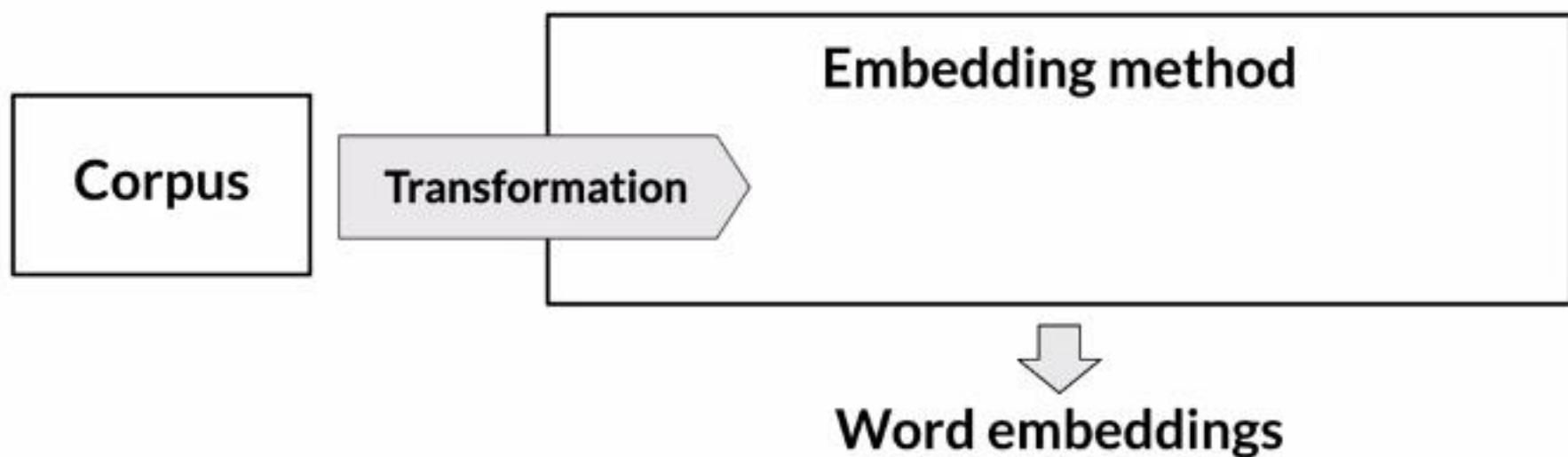
Corpus

Embedding method

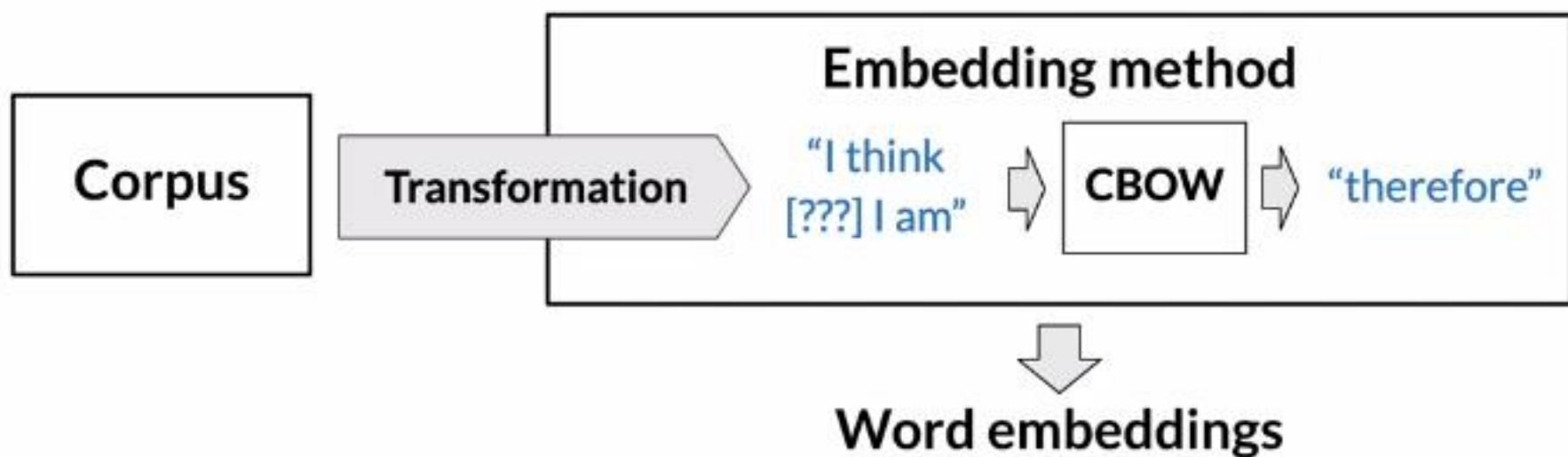
Continuous bag-of-words word embedding process



Continuous bag-of-words word embedding process



Continuous bag-of-words word embedding process



Corpus

Transformation CBOW

Center word prediction: rationale

Center word prediction: rationale

The little _____ ? _____ is barking

Center word prediction: rationale

The little _____ is barking



?

dog

puppy

hound

terrier

...

Corpus

Transformation

CBOW

Creating a training example

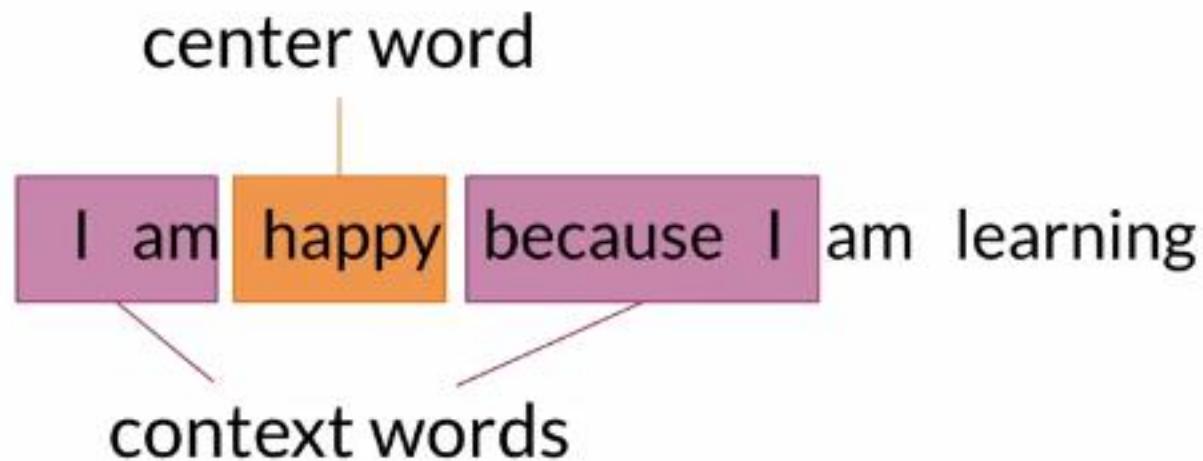
I am happy because I am learning

Creating a training example

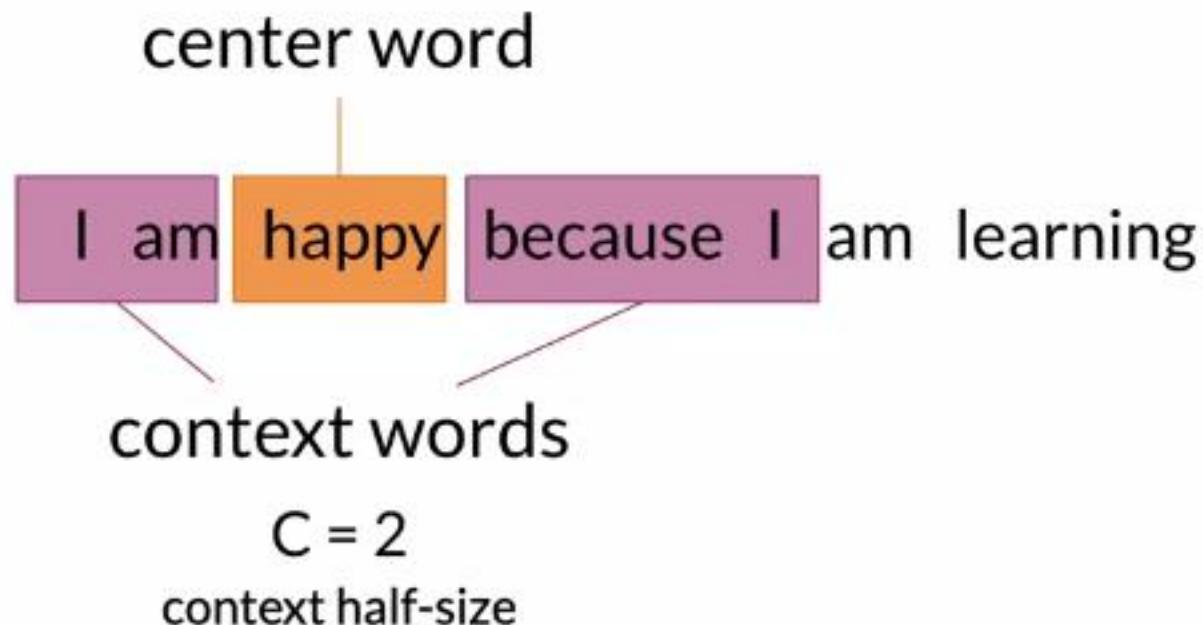
center word

I am **happy** because I am learning

Creating a training example



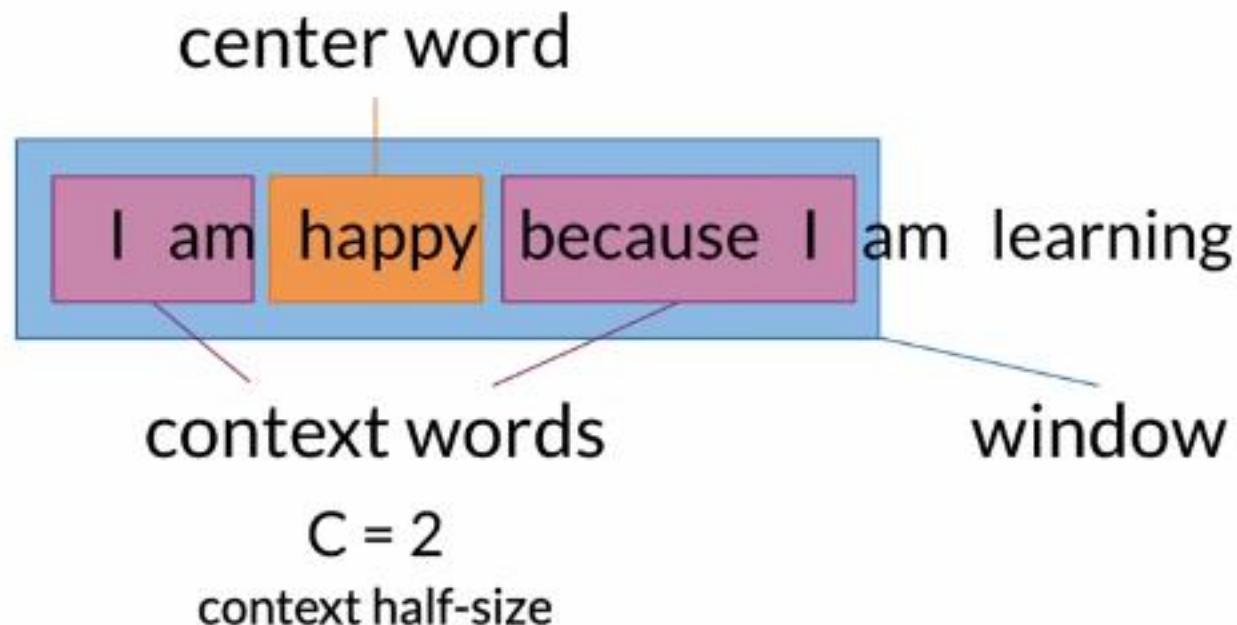
Creating a training example



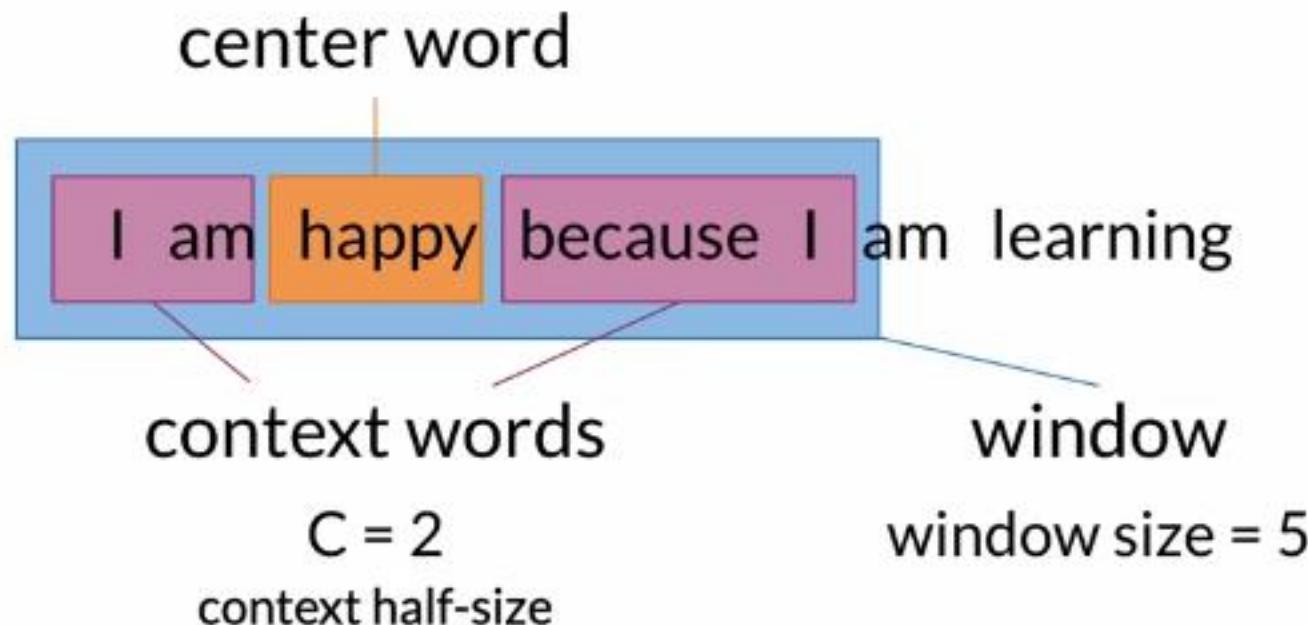
Corpus

Transformation CBOW

Creating a training example



Creating a training example

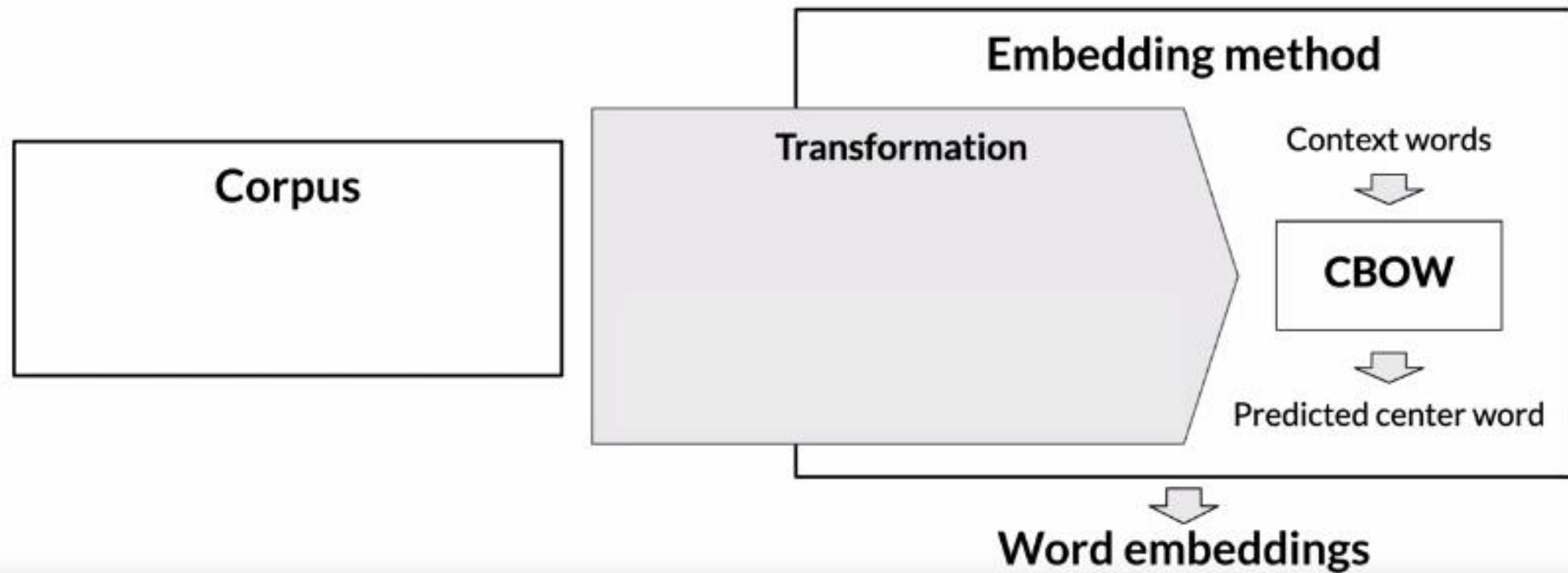


Corpus

Transformation

CBOW

From corpus to training

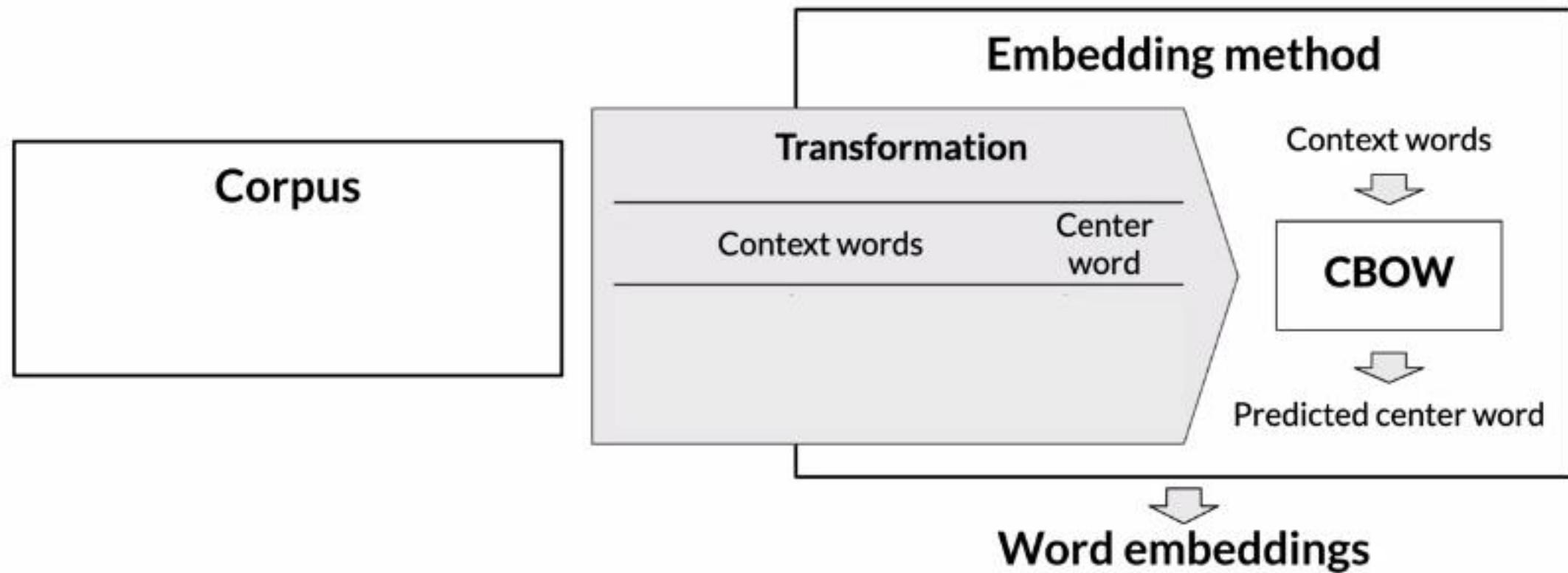


Corpus

Transformation

CBOW

From corpus to training

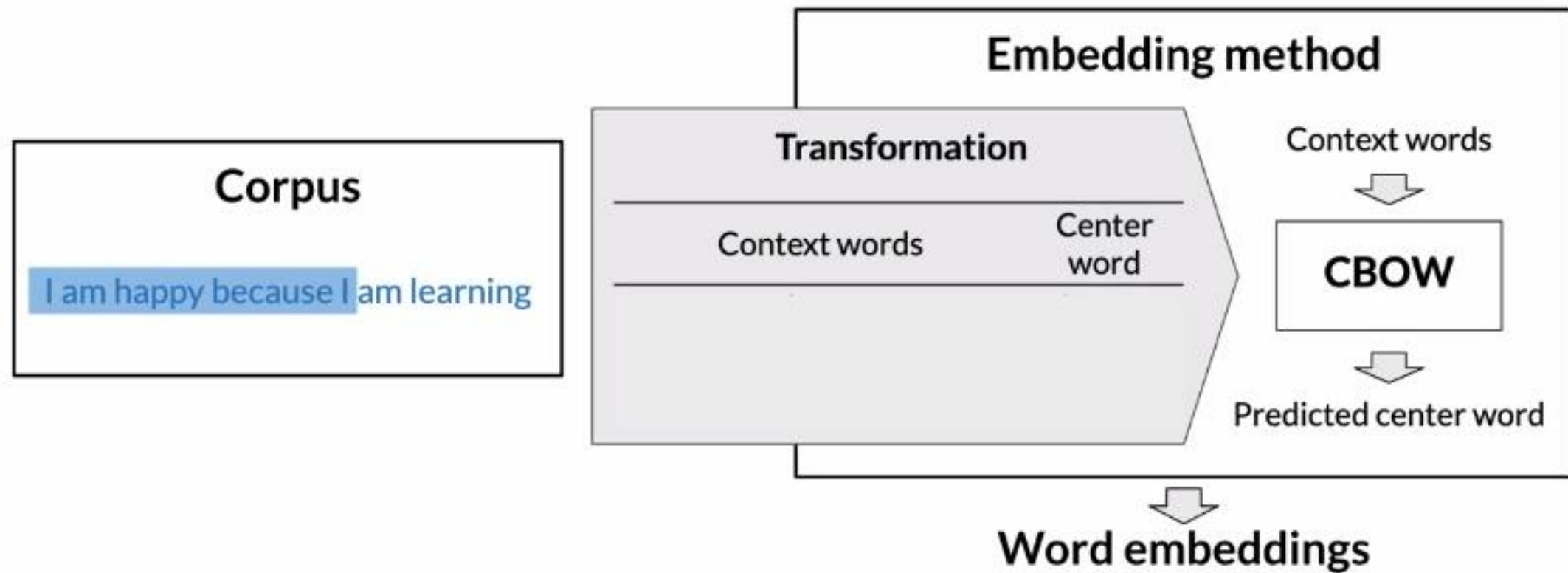


Corpus

Transformation

CBOW

From corpus to training

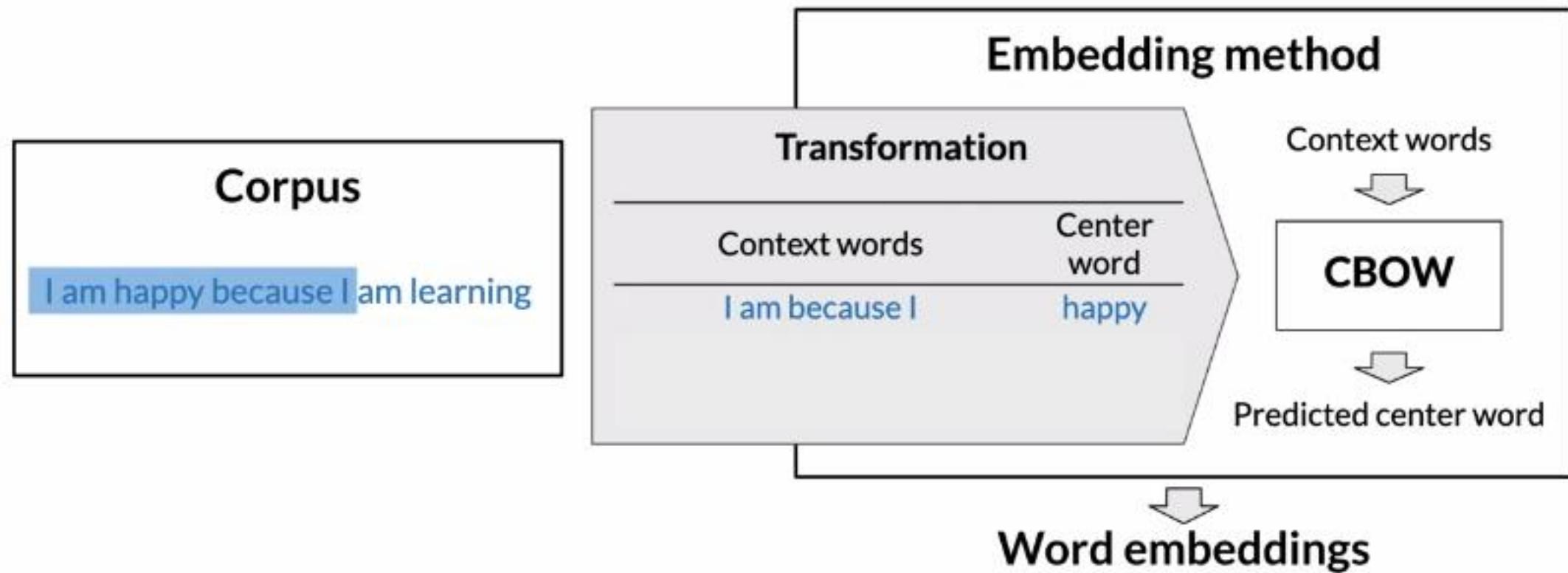


Corpus

Transformation

CBOW

From corpus to training

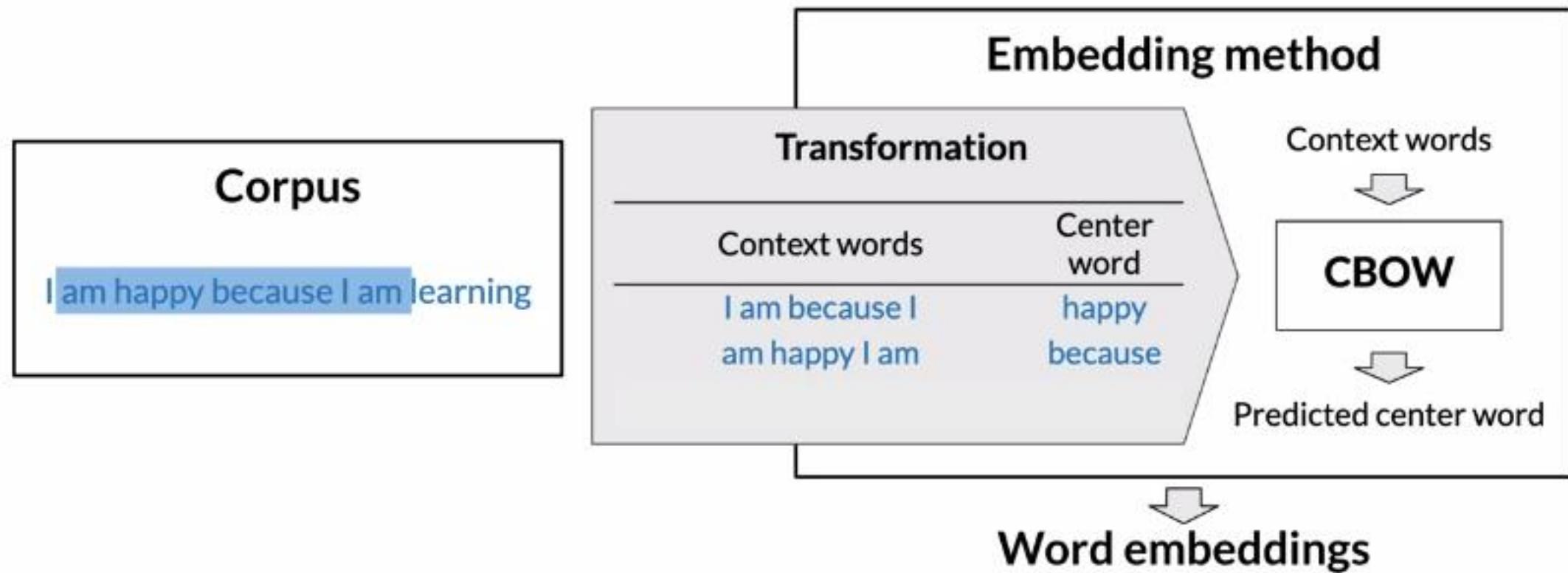


Corpus

Transformation

CBOW

From corpus to training

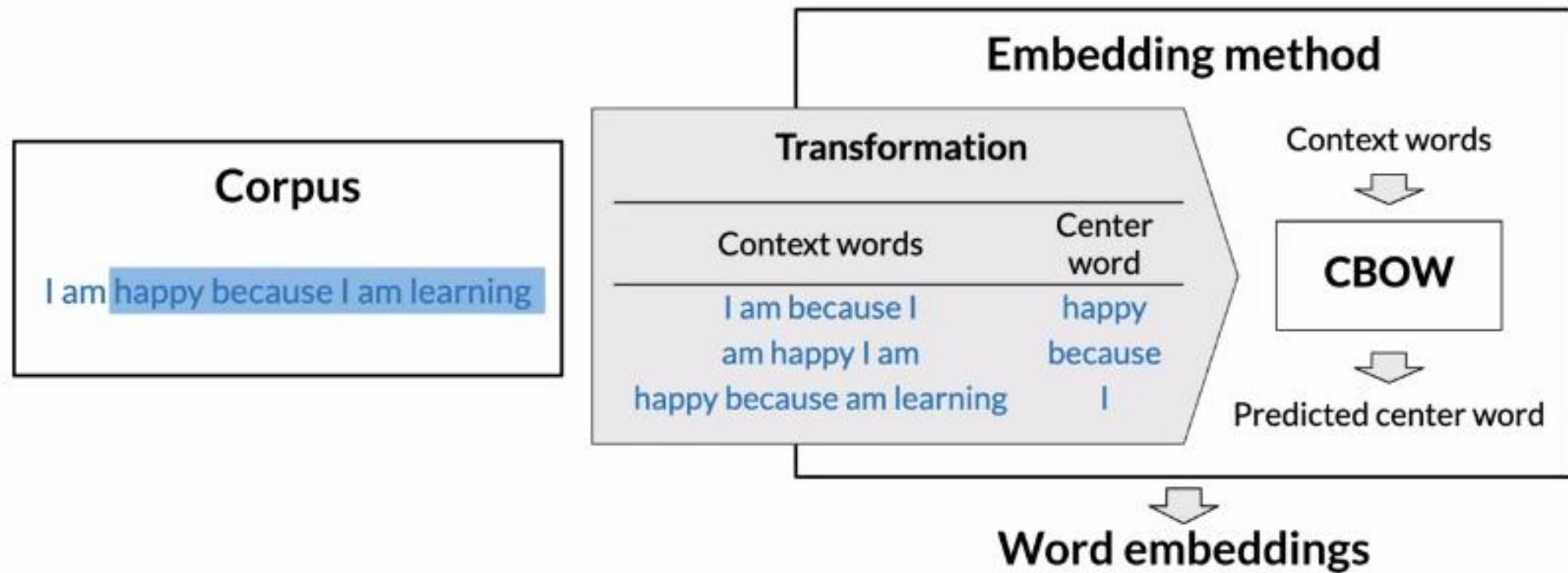


Corpus

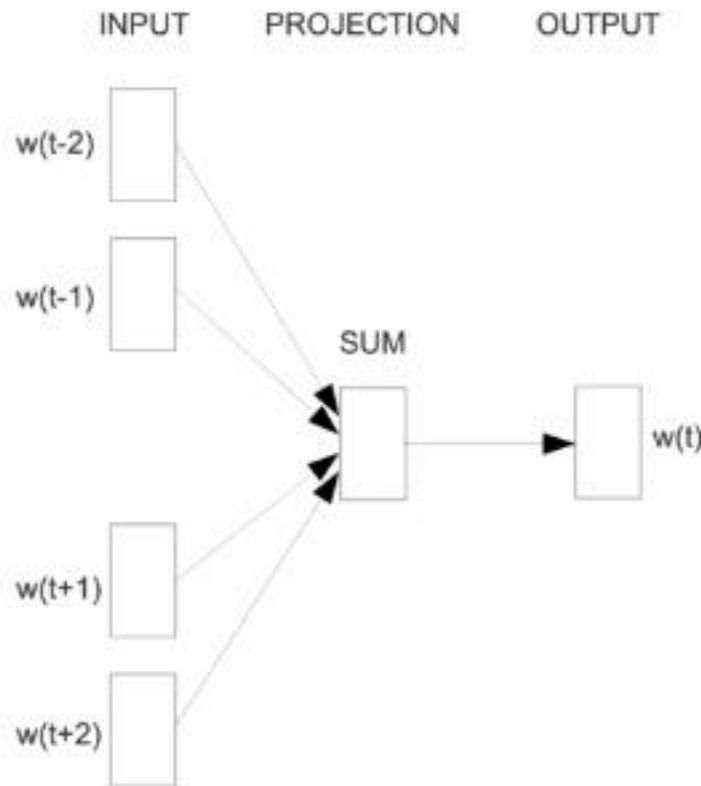
Transformation

CBOW

From corpus to training



CBOW in a nutshell



Source: Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013).
[Efficient Estimation of Word Representations in Vector Space](#)



deeplearning.ai

Cleaning and Tokenization

Cleaning and tokenization matters

Cleaning and tokenization matters

- Letter case

Cleaning and tokenization matters

- Letter case “The” == “the” == “THE”

Cleaning and tokenization matters

- Letter case

“The” == “the” == “THE” → lowercase / uppercase

Cleaning and tokenization matters

- Letter case “The” == “the” == “THE” → lowercase / upper case
- Punctuation

Cleaning and tokenization matters

- Letter case “The” == “the” == “THE” → lowercase / uppercase
- Punctuation , ! . ? → .

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " " « » „ „ → Ø

Cleaning and tokenization matters

- Letter case “The” == “the” == “THE” → lowercase / uppercase
- Punctuation , ! . ? → . “ “ ‘ ’ “ ” → Ø ... !! ??? → .

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " " « » „ „ → Ø ... !! ??? → .
- Numbers

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " " « » „ „ → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " " « » ‘ ’ ” → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " " « » „ „ → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210 → as is / <NUMBER>

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " " « » ' " → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210 → as is / <NUMBER>
- Special characters

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " " « » „ „ → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210 → as is / <NUMBER>
- Special characters ⌂ \$ € § ¶ **

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " “ ” « » ‘ ’ ” → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210 → as is / <NUMBER>
- Special characters ₣ \$ € § ¶ ** → Ø

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " " « » ' " → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210 → as is / <NUMBER>
- Special characters ⌂ \$ € § ¶ ** → Ø
- Special words

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " “ ” « » ‘ ’ ” → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210 → as is / <NUMBER>
- Special characters ⌂ \$ € § ¶ ** → Ø
- Special words 😊 #nlp

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → lowercase / uppercase
- Punctuation , ! . ? → . " “ ” « » ‘ ’ ” → Ø ... !! ??? → .
- Numbers 1 2 3 5 8 → Ø 3.14159 90210 → as is / <NUMBER>
- Special characters ⌧ \$ € § ¶ ** → Ø
- Special words 😊 #nlp → :happy: #nlp

Example in Python: corpus

Who ❤️ "word embeddings" in 2020? I do!!!

Example in Python: corpus

Who ❤️ "word embeddings" in 2020? I do!!!

emoji

punctuation

number

Example in Python: libraries

```
# pip install nltk
# pip install emoji

import nltk
from nltk.tokenize import word_tokenize
import emoji

nltk.download('punkt') # download pre-trained Punkt tokenizer for English
```

Example in Python: code

```
corpus = 'Who ❤️ "word embeddings" in 2020? I do!!!'  
  
data = re.sub(r'[,!?;-]+', '.', corpus)
```

Example in Python: code

```
corpus = 'Who ❤️ "word embeddings" in 2020? I do!!!'
```

```
data = re.sub(r'[,!?;-]+', '.', corpus)
```

→ Who ❤️ "word embeddings" in 2020. I do.

Example in Python: code

```
corpus = 'Who ❤️ "word embeddings" in 2020? I do!!!'

data = re.sub(r'[,!?;-]+', '.', corpus)
data = nltk.word_tokenize(data) # tokenize string to words

→ ['Who', '❤️', '', 'word', 'embeddings', "", 'in', '2020', '.', 'I',
'do', '.']
```

Example in Python: code

```
corpus = 'Who ❤️ "word embeddings" in 2020? I do!!!'

data = re.sub(r'[,!?;-]+', '.', corpus)
data = nltk.word_tokenize(data) # tokenize string to words
data = [ ch.lower() for ch in data
         if ch.isalpha()
         or ch == '.'
         or emoji.get_emoji_regexp().search(ch)
     ]
```

Example in Python: code

```
corpus = 'Who ❤️ "word embeddings" in 2020? I do!!!'

data = re.sub(r'[,!?;-]+', '.', corpus)
data = nltk.word_tokenize(data) # tokenize string to words
data = [ ch.lower() for ch in data
         if ch.isalpha()
         or ch == '.'
         or emoji.get_emoji_regexp().search(ch)
     ]
→ ['who', '❤️', 'word', 'embeddings', 'in', '.', 'i', 'do', '.']
```



deeplearning.ai

Sliding Window of Words in Python

Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```

Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```

Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```

Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```



Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```

I	am	happy	because	I	am	learning
0	1	2	3	4	5	6

Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```

I	am	happy	because	I	am	learning
0	1	2	3	4	5	6

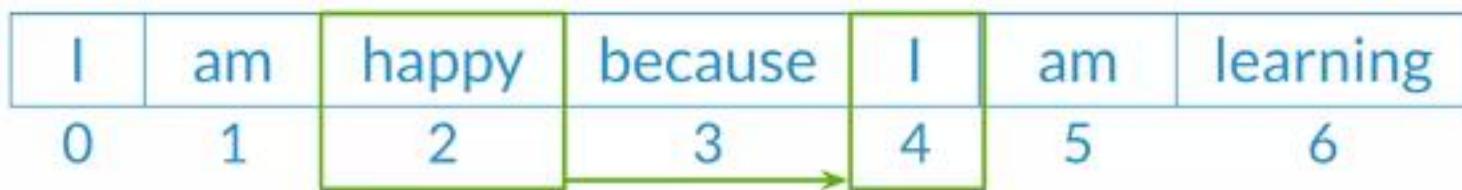
Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```

I	am	happy	because	I	am	learning
0	1	2	3	4	5	6

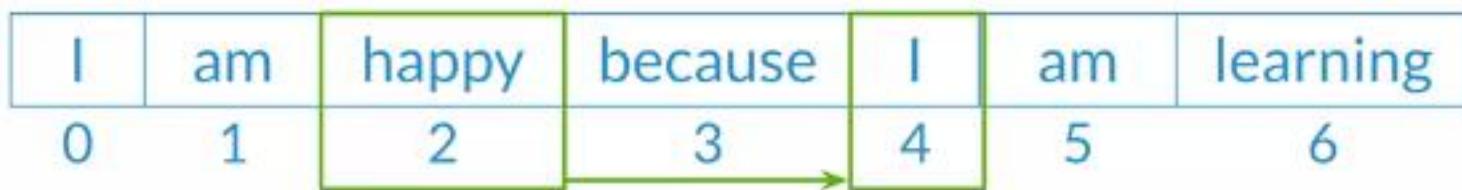
Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```



Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```



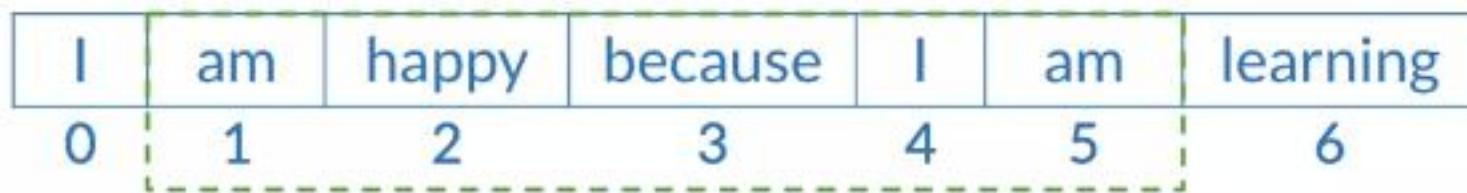
Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```

I	am	happy	because	I	am	learning
0	1	2	3	4	5	6

Sliding window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```



Sliding window of words in Python

```
def get_windows(words, C):
    ...
    yield context_words, center_word

for x, y in get_windows(
        ['i', 'am', 'happy', 'because', 'i', 'am', 'learning'],
        2
    ):
    print(f'{x}\t{y}')
```

Sliding window of words in Python

```
def get_windows(words, C):
    ...
    yield context_words, center_word

for x, y in get_windows(
        ['i', 'am', 'happy', 'because', 'i', 'am', 'learning'],
        2
    ):
    print(f'{x}\t{y}')
```

Sliding window of words in Python

```
for x, y in get_windows(  
    ['i', 'am', 'happy', 'because', 'i', 'am', 'learning'],  
    2  
):  
    print(f'{x}\t{y}')
```

```
→ ['I', 'am', 'because', 'I']      happy  
['am', 'happy', 'I', 'am']        because  
['happy', 'because', 'am', 'learning'] I
```



deeplearning.ai

Transforming Words into Vectors

Transforming center words into vectors

Corpus I am happy because I am learning

Transforming center words into vectors

Corpus I am happy because I am learning

Vocabulary am, because, happy, I, learning

Transforming center words into vectors

Corpus I am happy because I am learning

Vocabulary am, because, happy, I, learning

One-hot
vector

am
because
happy
I
learning



Transforming center words into vectors

Corpus I am happy because I am learning

Vocabulary am, because, happy, I, learning

One-hot
vector

	am
am	1
because	0
happy	0
I	0
learning	0

Transforming center words into vectors

Corpus I am happy because I am learning

Vocabulary am, because, happy, I, learning

One-hot vector	am	because
am	1	0
because	0	1
happy	0	0
I	0	0
learning	0	0

Transforming center words into vectors

Corpus I am happy because I am learning

Vocabulary am, because, happy, I, learning

One-hot vector	am	because	happy	I	learning
am	$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$
because	$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$
happy	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$
I	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$
learning	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

Transforming context words into vectors

Average of individual one-hot vectors

Transforming context words into vectors

Average of individual one-hot vectors

I am because |

Transforming context words into vectors

Average of individual one-hot vectors

	I	am	because	I
am	0			
because	0			
happy	0			
I	1			
learning	0			

Transforming context words into vectors

Average of individual one-hot vectors

	I	am	because	I
am	0			
because	0			
happy	0			
I	1			
learning	0			

Transforming context words into vectors

Average of individual one-hot vectors

	I	am	because	I
am	0	1	0	0
because	0	0	1	0
happy	0	0	0	0
I	1	0	0	1
learning	0	0	0	0

Transforming context words into vectors

Average of individual one-hot vectors

	I	am	because	I
am	0	1	0	0
because	0	0	1	0
happy	0	0	0	0
I	1	0	0	1
learning	0	0	0	0

Transforming context words into vectors

Average of individual one-hot vectors

$$\left[\begin{array}{c} I \\ am \\ because \\ happy \\ I \\ learning \end{array} \right] = \frac{1}{4} \left(\left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{array} \right] + \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \right] + \left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{array} \right] + \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{array} \right] \right)$$

Transforming context words into vectors

Average of individual one-hot vectors

$$\left[\begin{array}{c} I \\ am \\ because \\ happy \\ I \\ learning \end{array} \right] = \frac{1}{4} \left(\left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{array} \right] + \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \right] + \left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{array} \right] + \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{array} \right] \right)$$

Final prepared training set

Context words	Context words vector	Center word	Center word vector
<i>I am because I</i>	[0.25; 0.25; 0; 0.5; 0]	<i>happy</i>	[0; 0; 1; 0; 0]

Final prepared training set

Context words	Context words vector	Center word	Center word vector
<i>I am because I</i>	[0.25; 0.25; 0; 0.5; 0]	<i>happy</i>	[0; 0; 1; 0; 0]
<i>am happy I am</i>	[0.5; 0; 0.25; 0.25; 0]	<i>because</i>	[0; 1; 0; 0; 0]
<i>happy because am learning</i>	[0.25; 0.25; 0.25; 0; 0.25]	<i>I</i>	[0; 0; 0; 1; 0]

Architecture of the CBOW model

Architecture of the CBOW model

Input layer



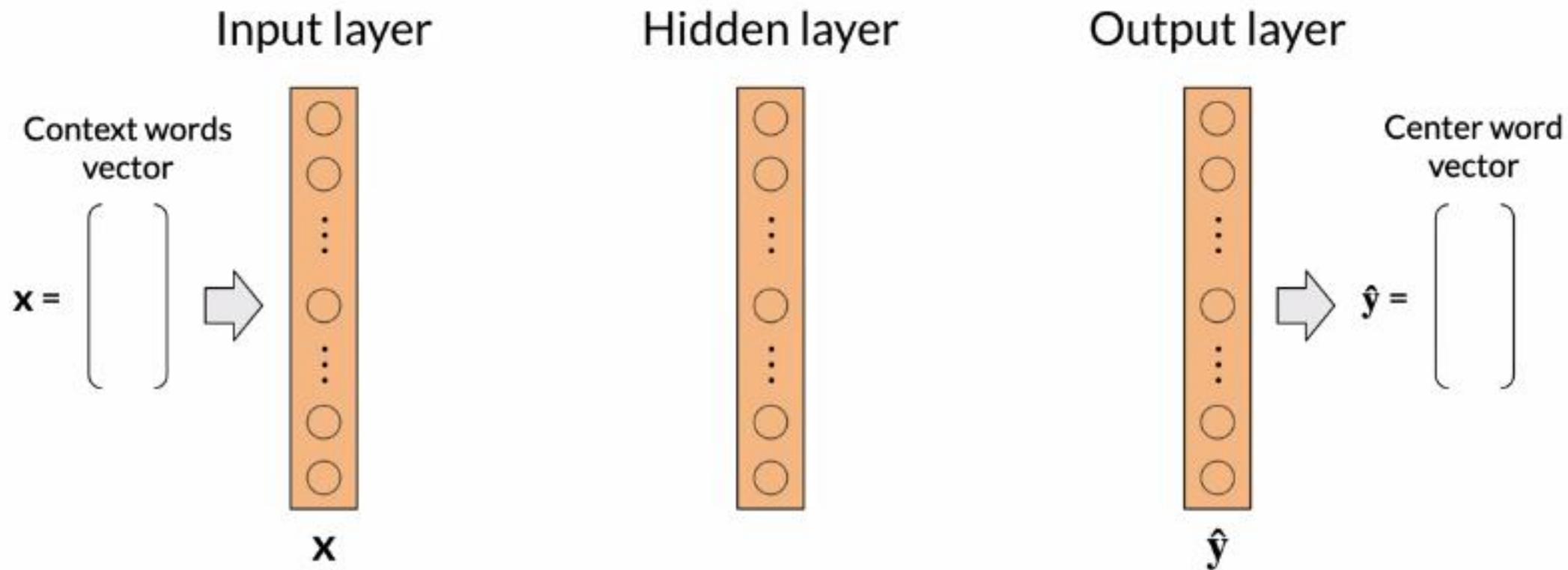
Hidden layer



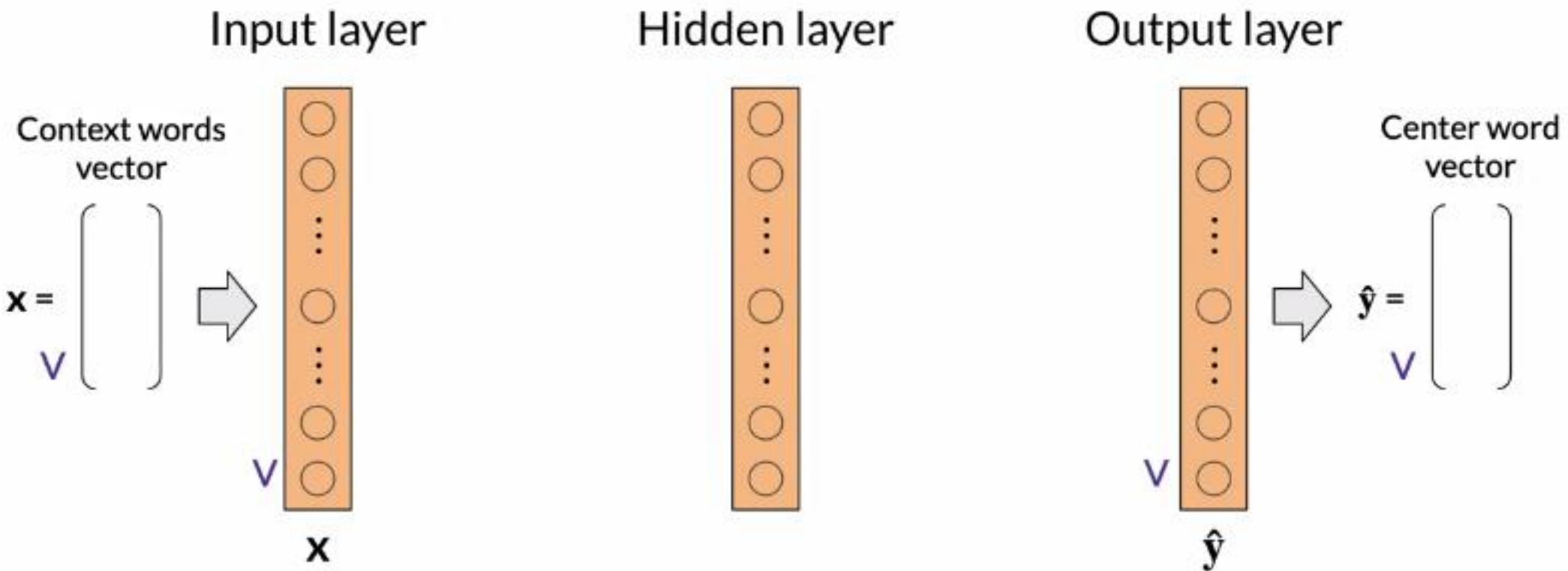
Output layer



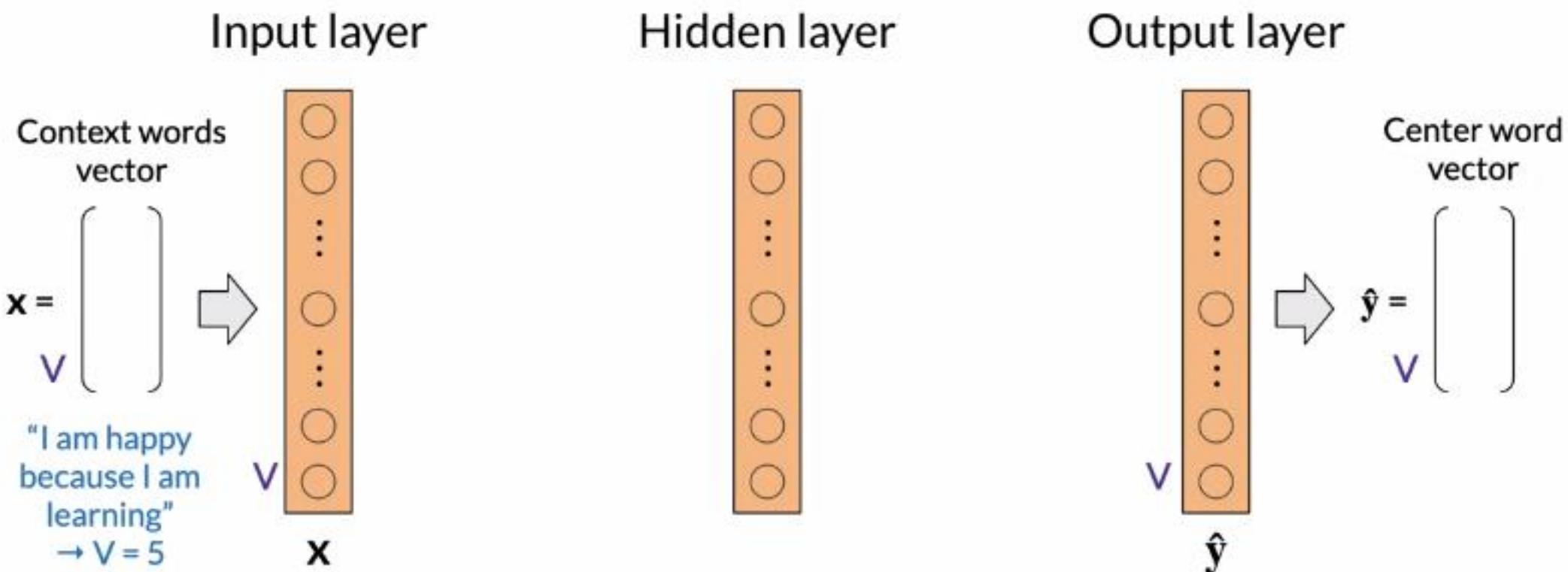
Architecture of the CBOW model



Architecture of the CBOW model



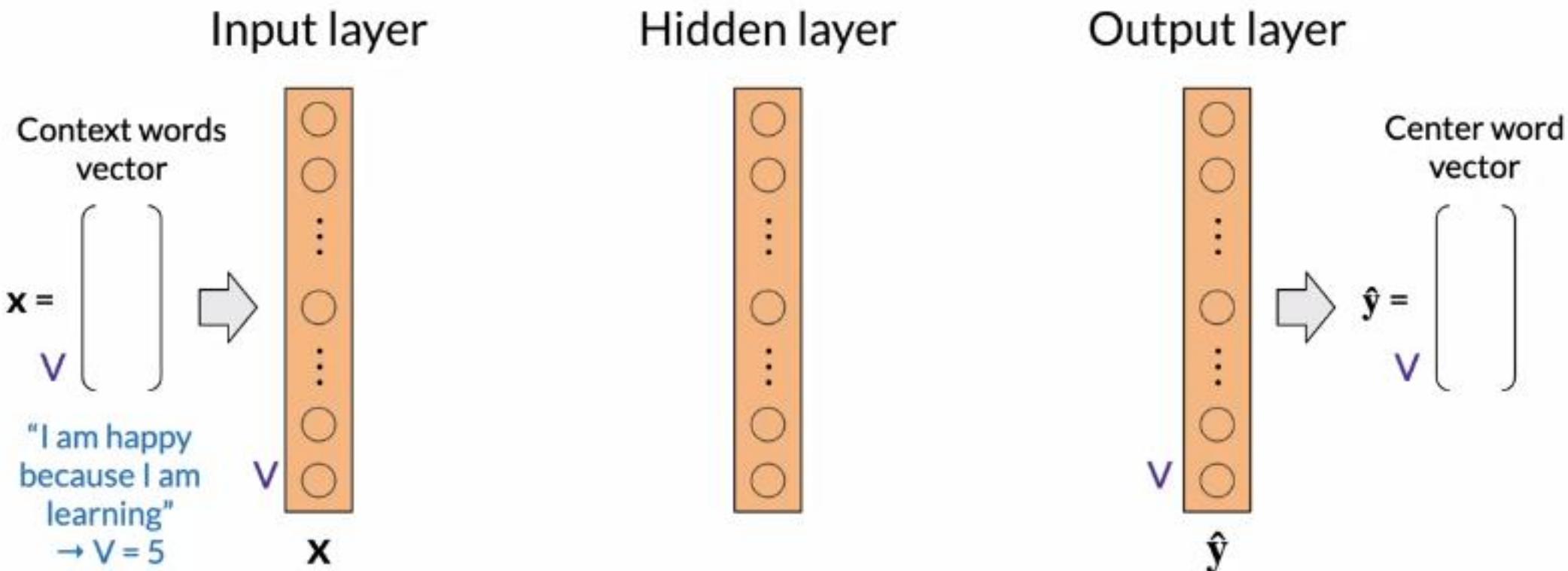
Architecture of the CBOW model



Architecture of the CBOW model

Hyperparameters

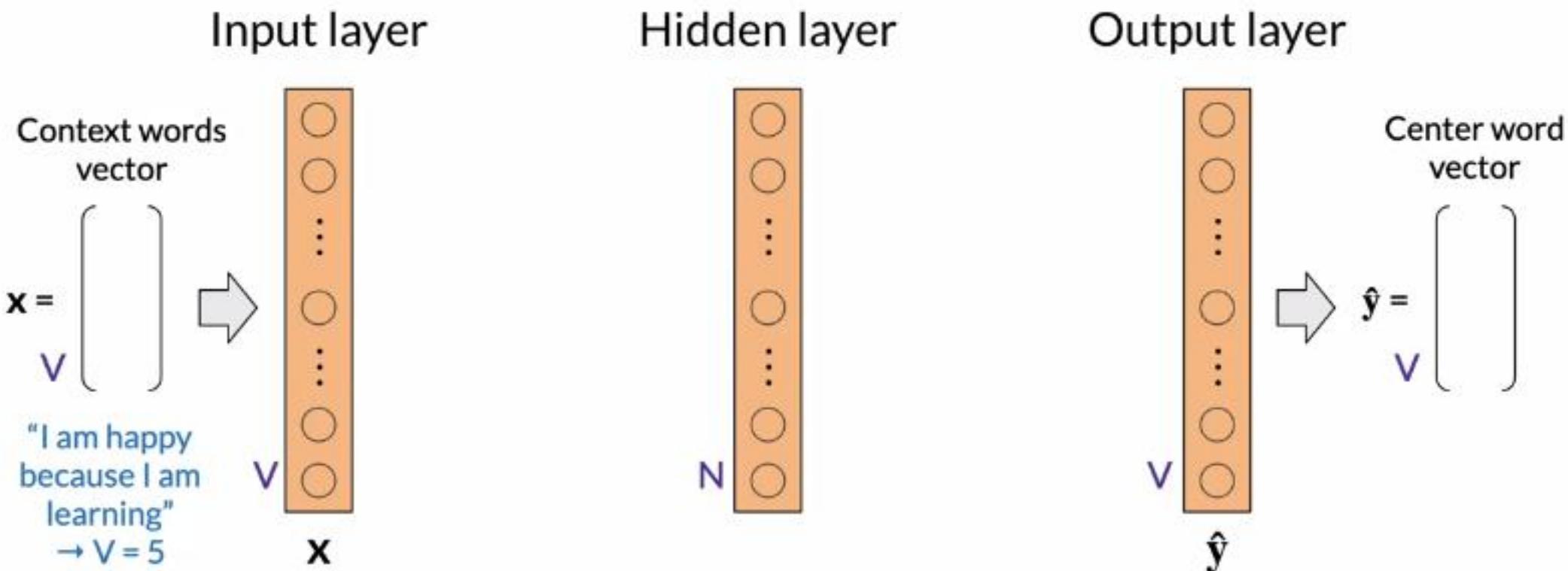
Word embedding size ...



Hyperparameters

N: Word embedding size ...

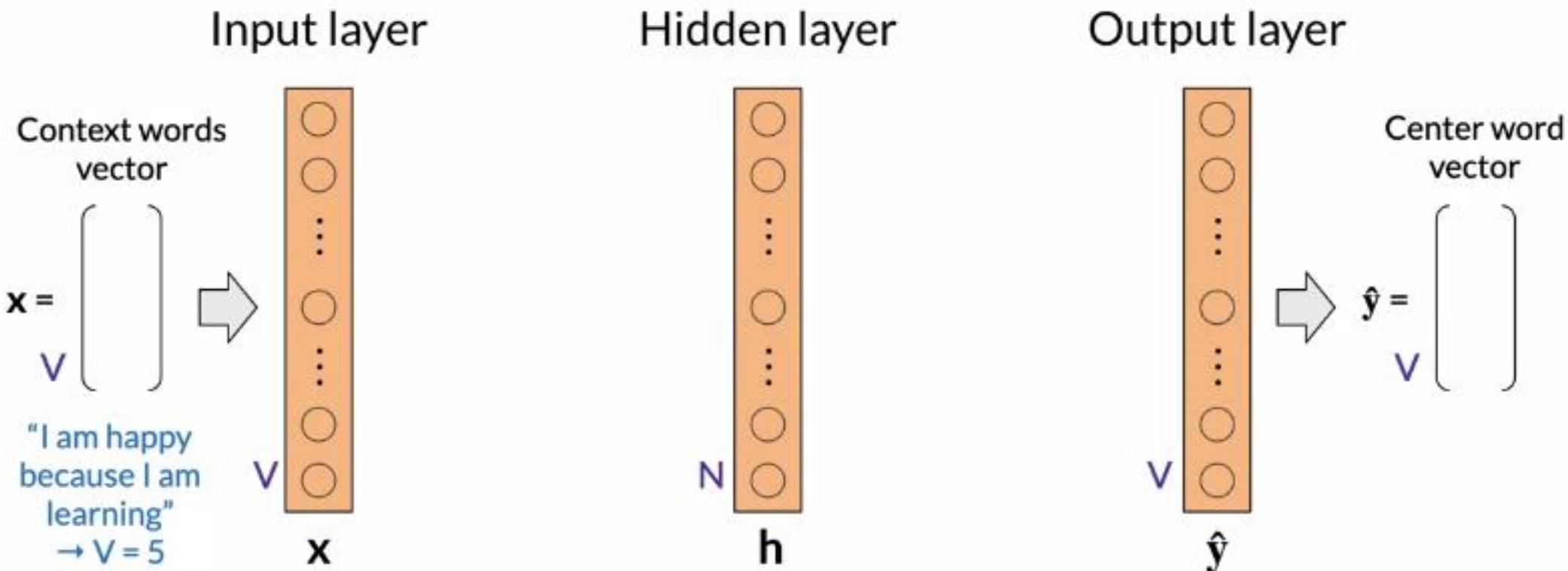
Architecture of the CBOW model



Hyperparameters

N: Word embedding size ...

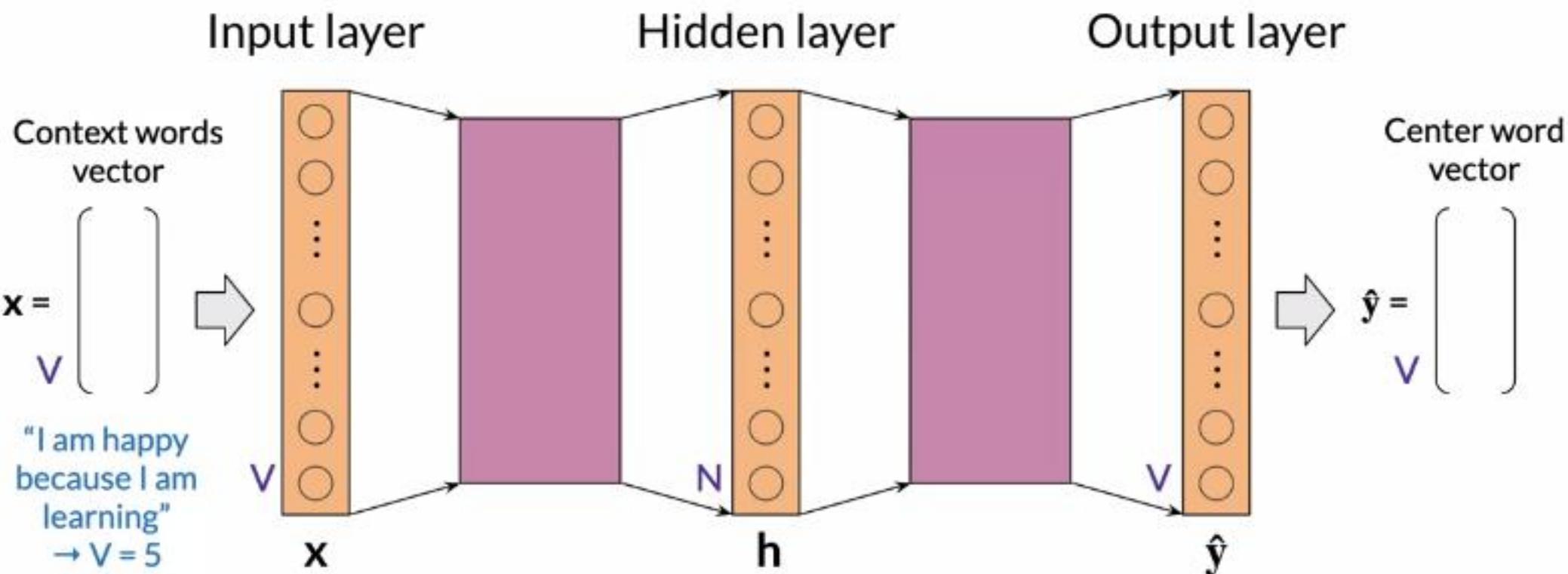
Architecture of the CBOW model



Hyperparameters

N: Word embedding size ...

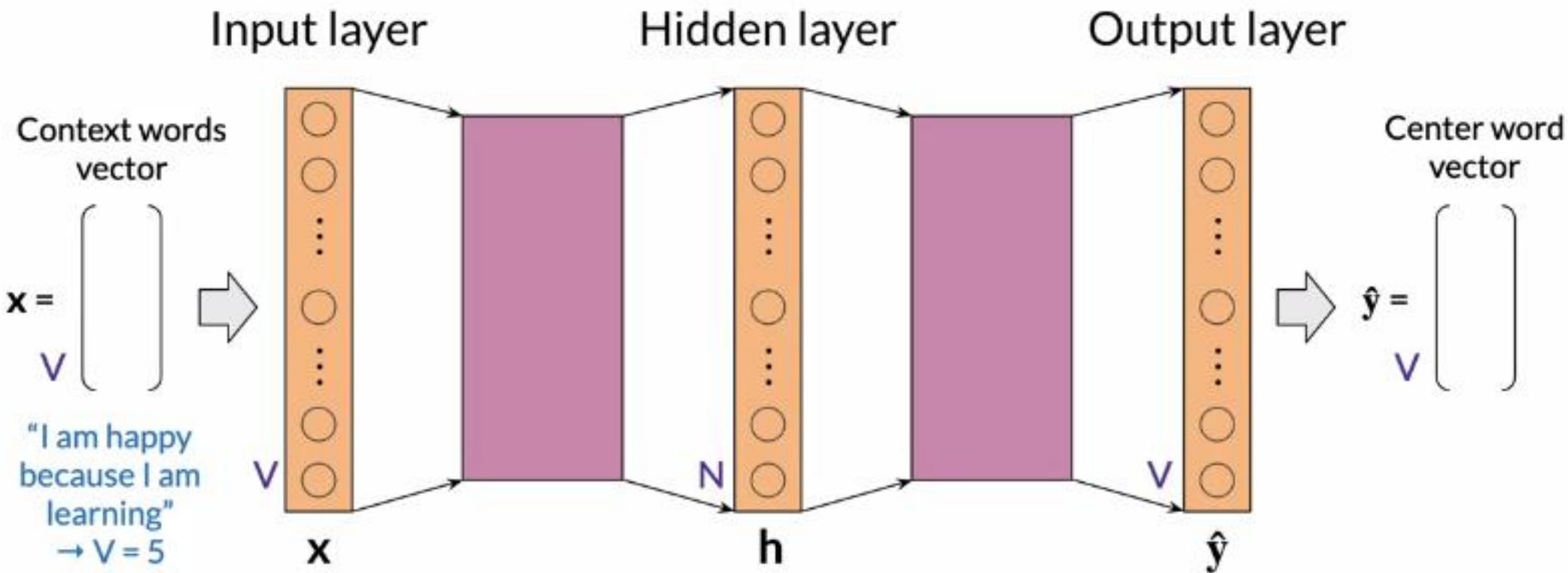
Architecture of the CBOW model



Architecture of the CBOW model

Hyperparameters

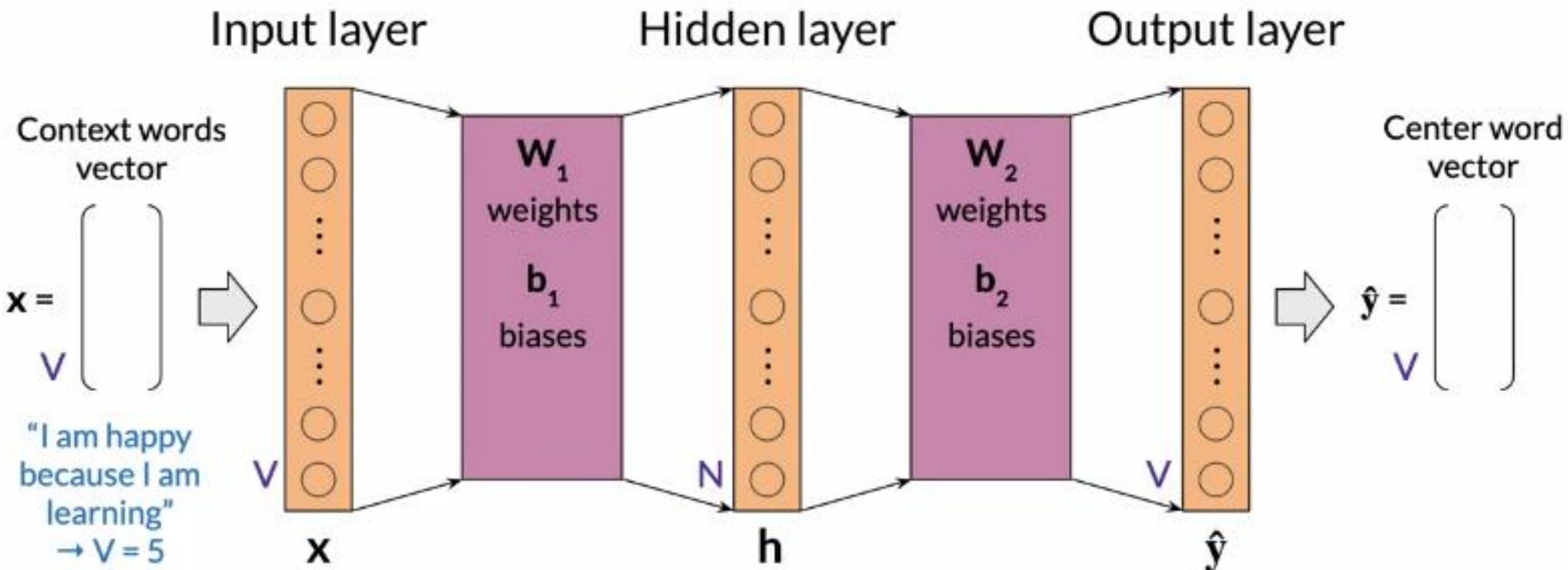
N: Word embedding size ...



Architecture of the CBOW model

Hyperparameters

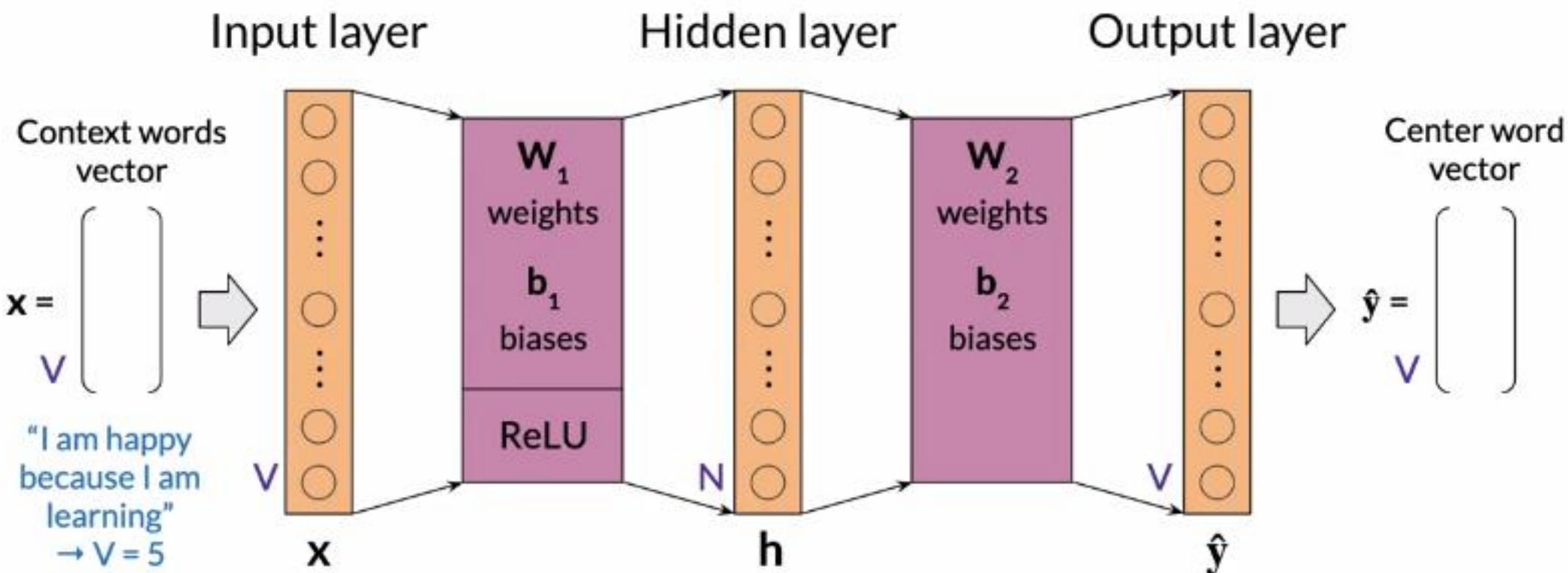
N: Word embedding size ...



Hyperparameters

N: Word embedding size ...

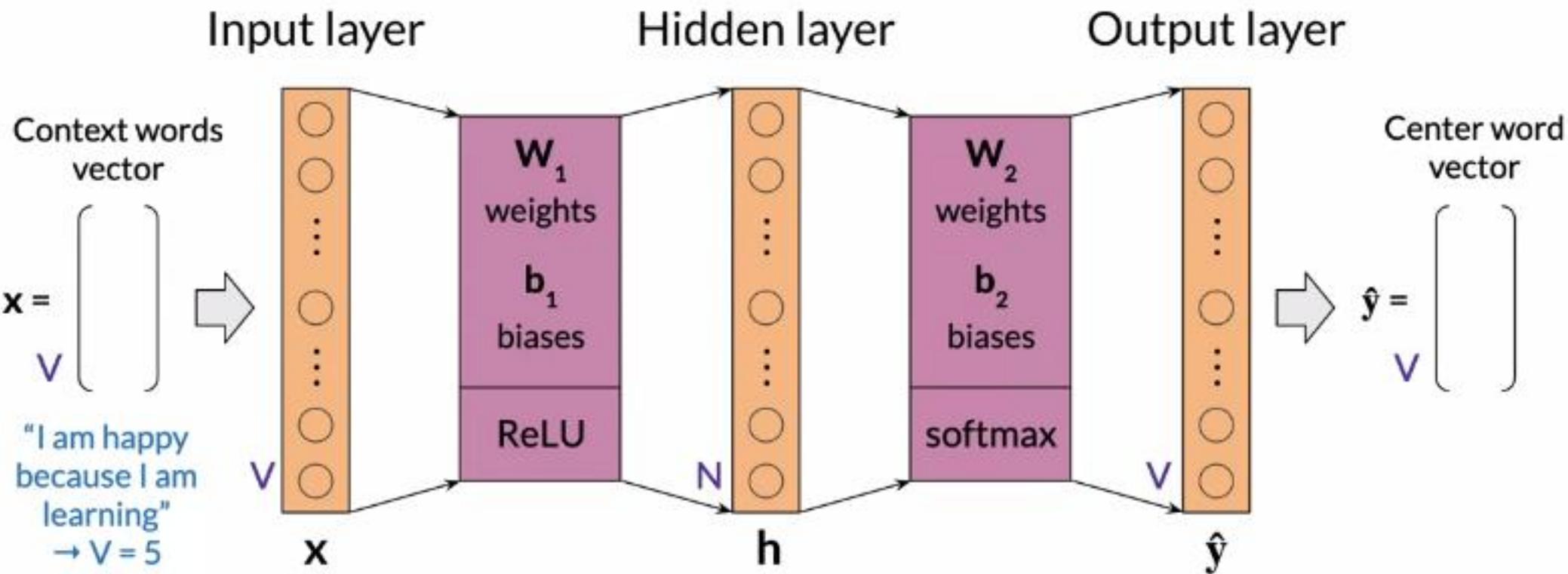
Architecture of the CBOW model



Architecture of the CBOW model

Hyperparameters

N: Word embedding size ...



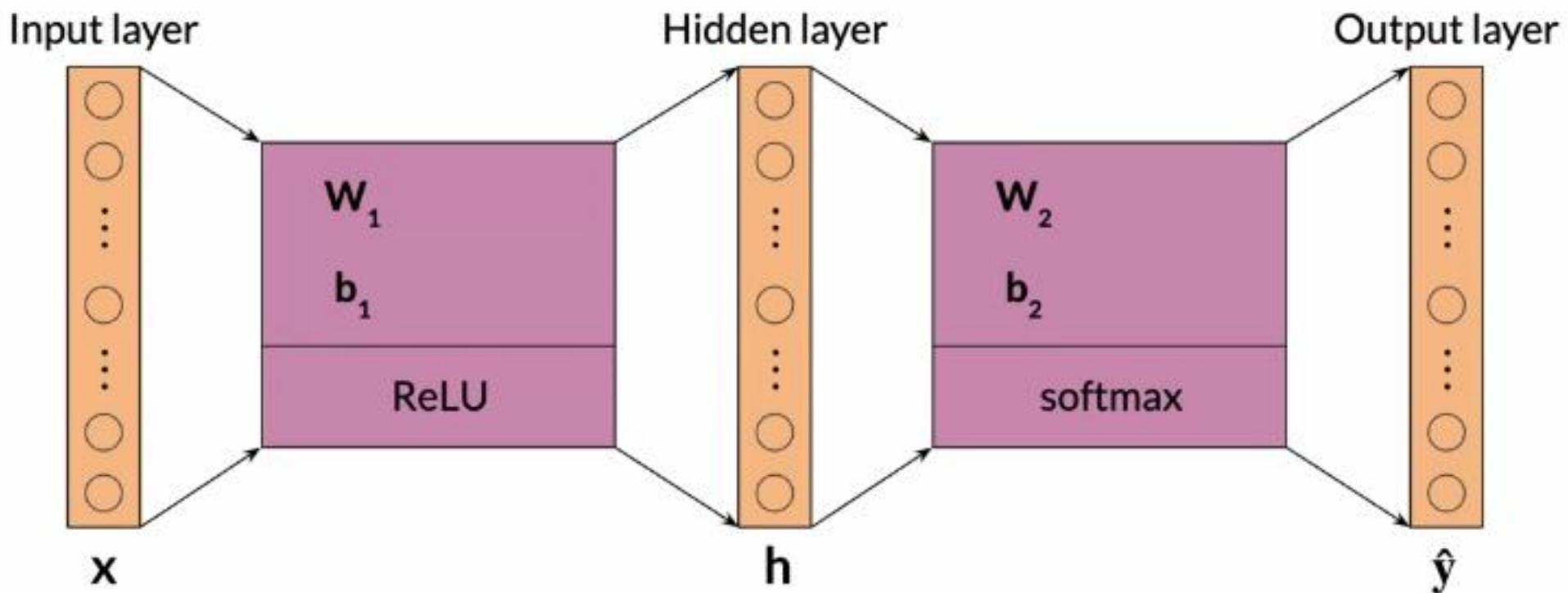


deeplearning.ai

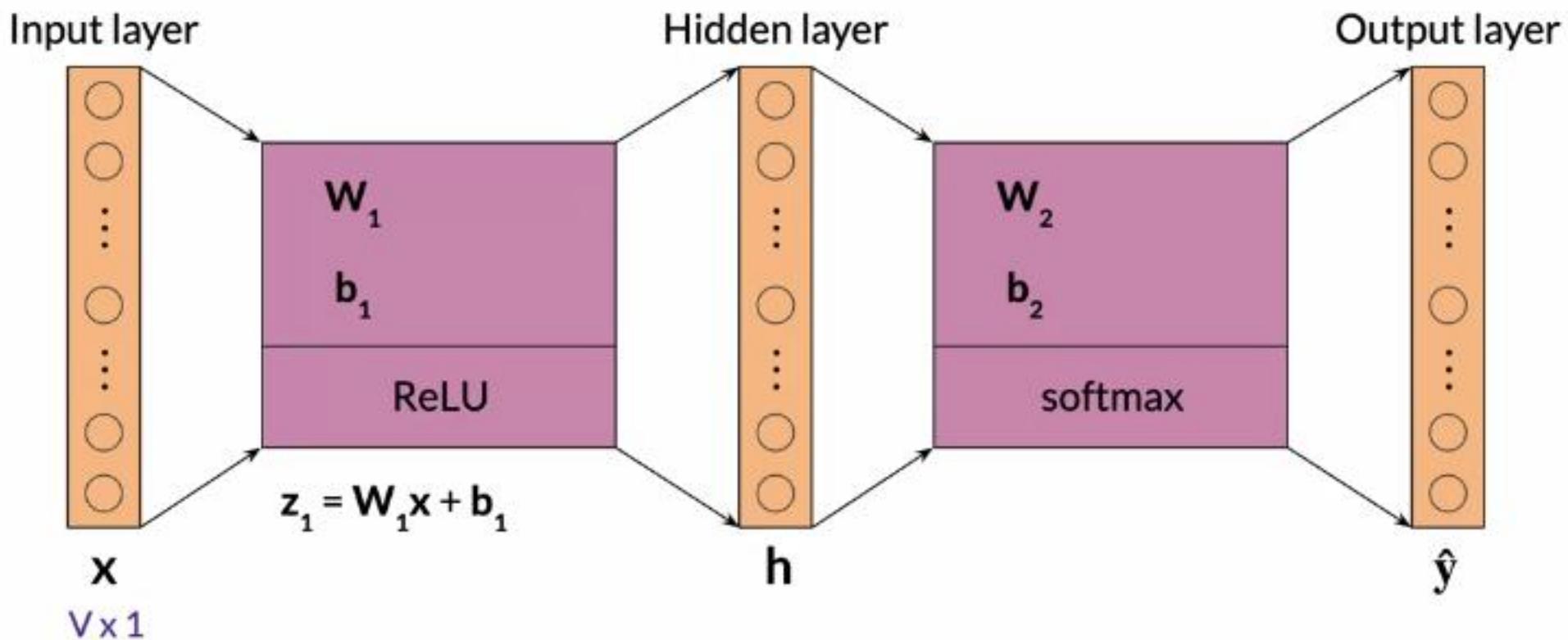
Architecture of the CBOW Model:

Dimensions

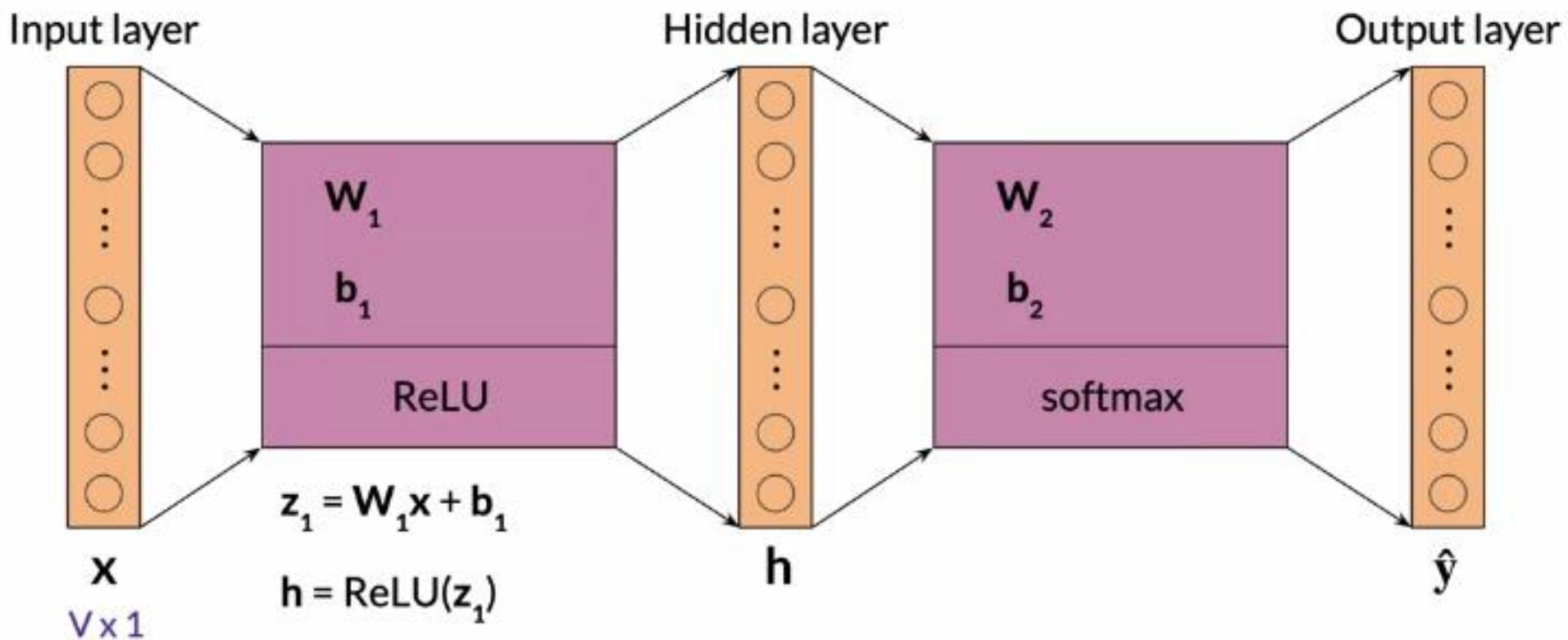
Dimensions (single input)



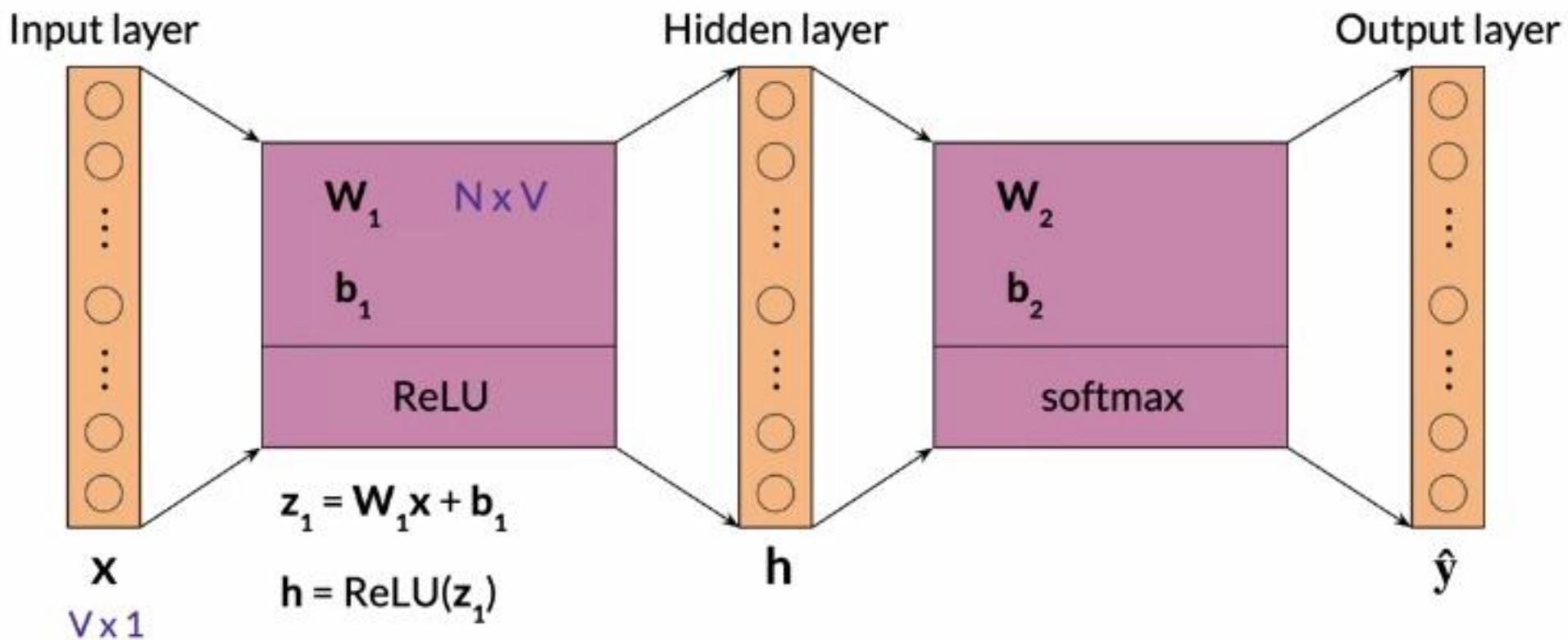
Dimensions (single input)



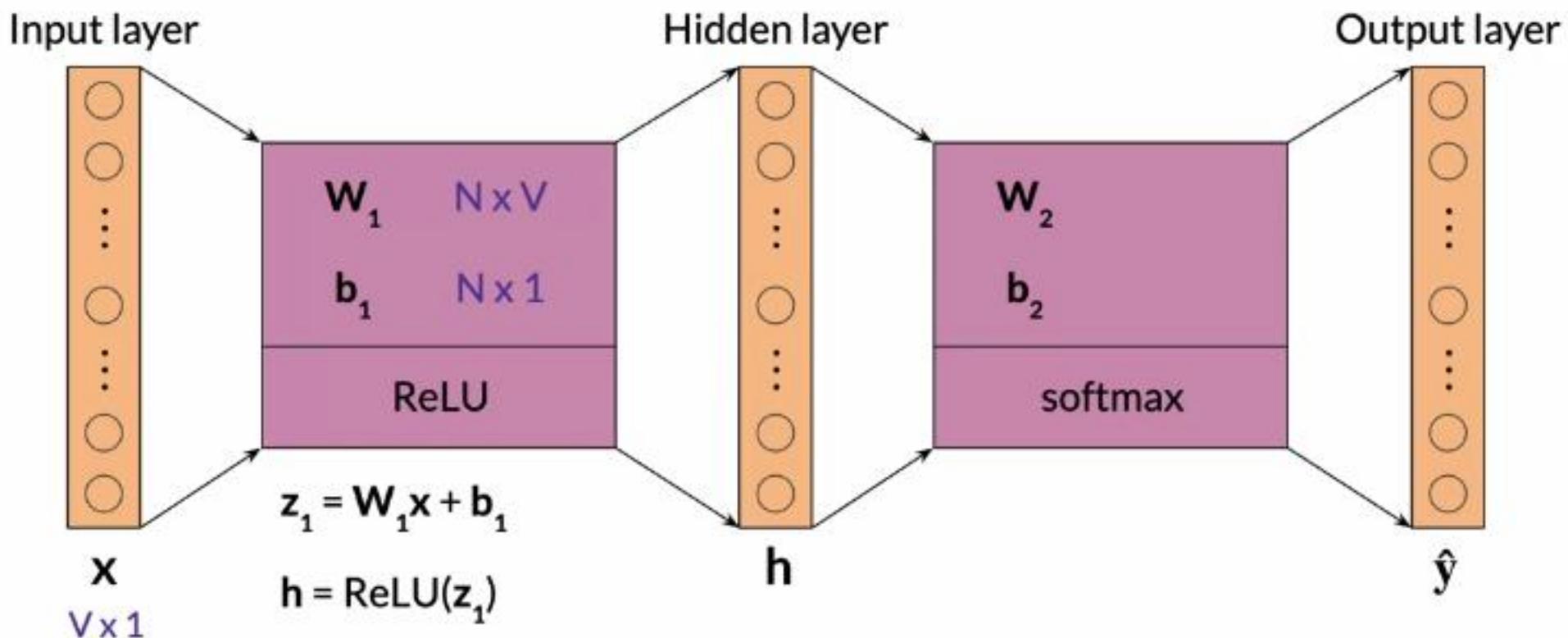
Dimensions (single input)



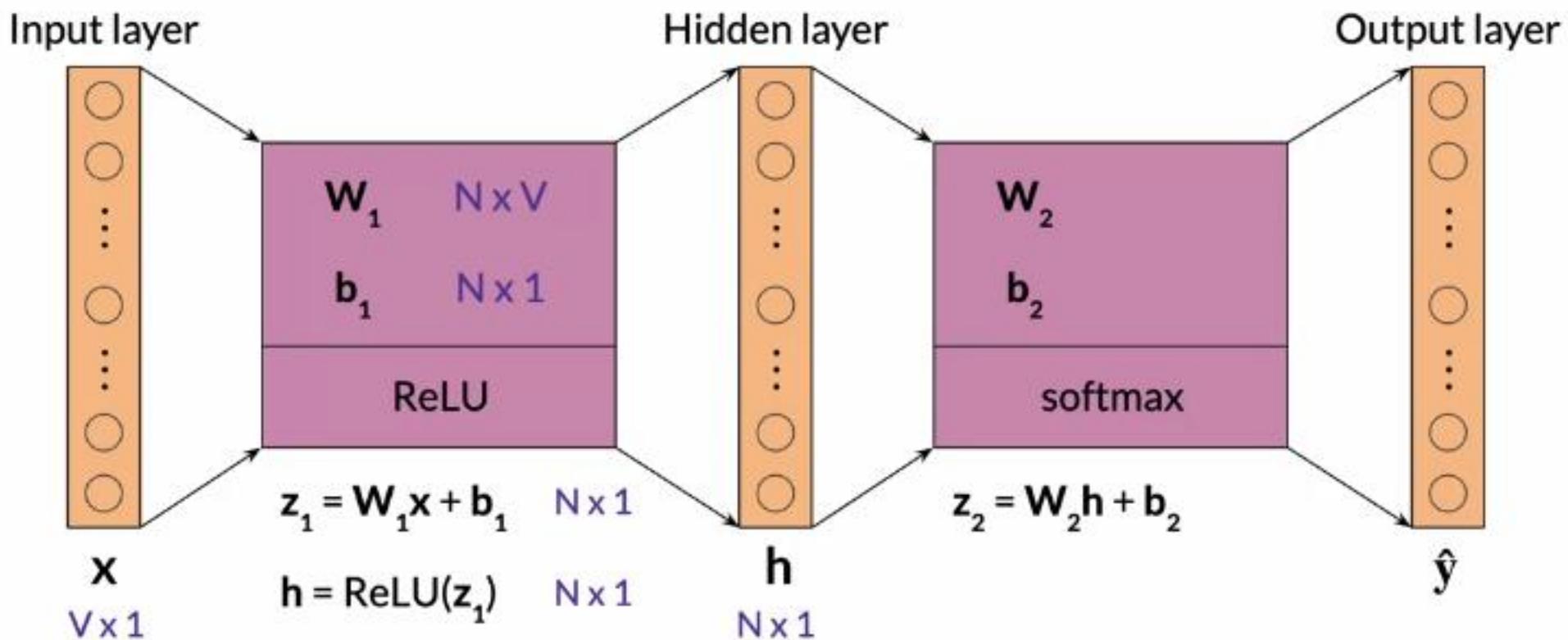
Dimensions (single input)



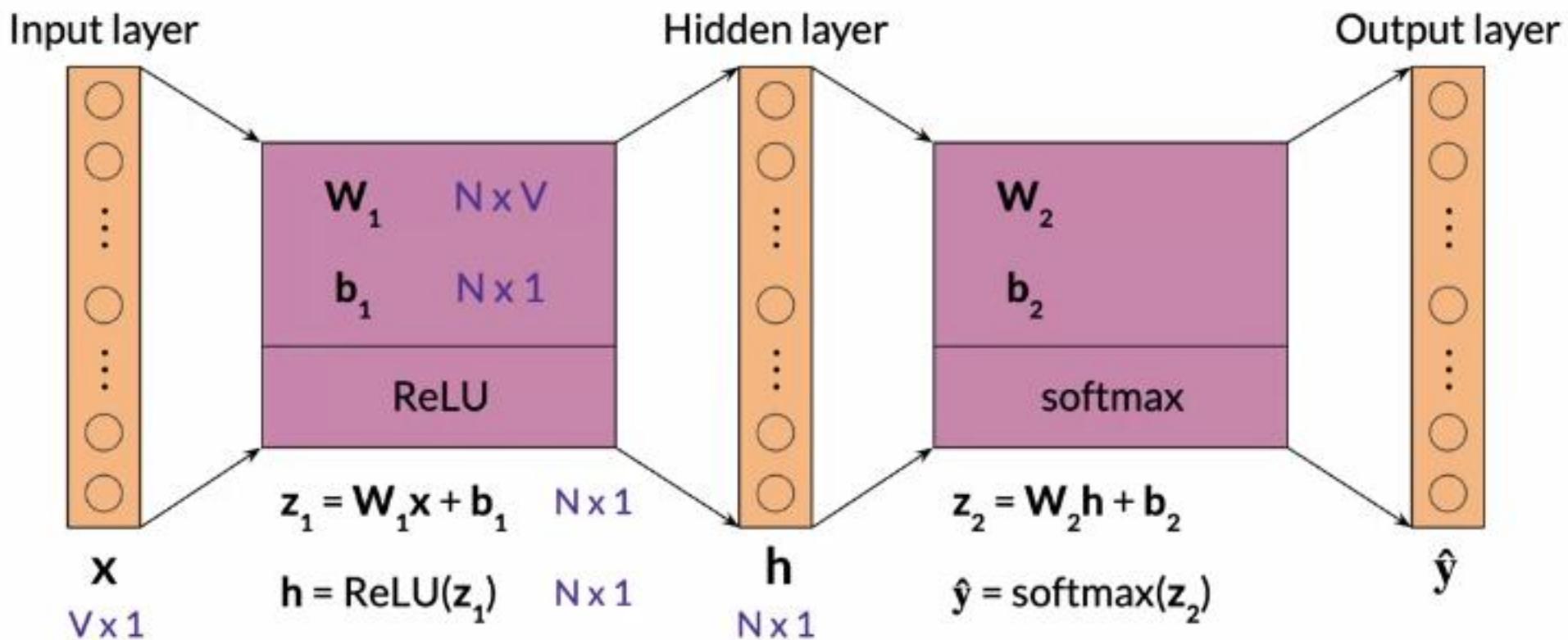
Dimensions (single input)



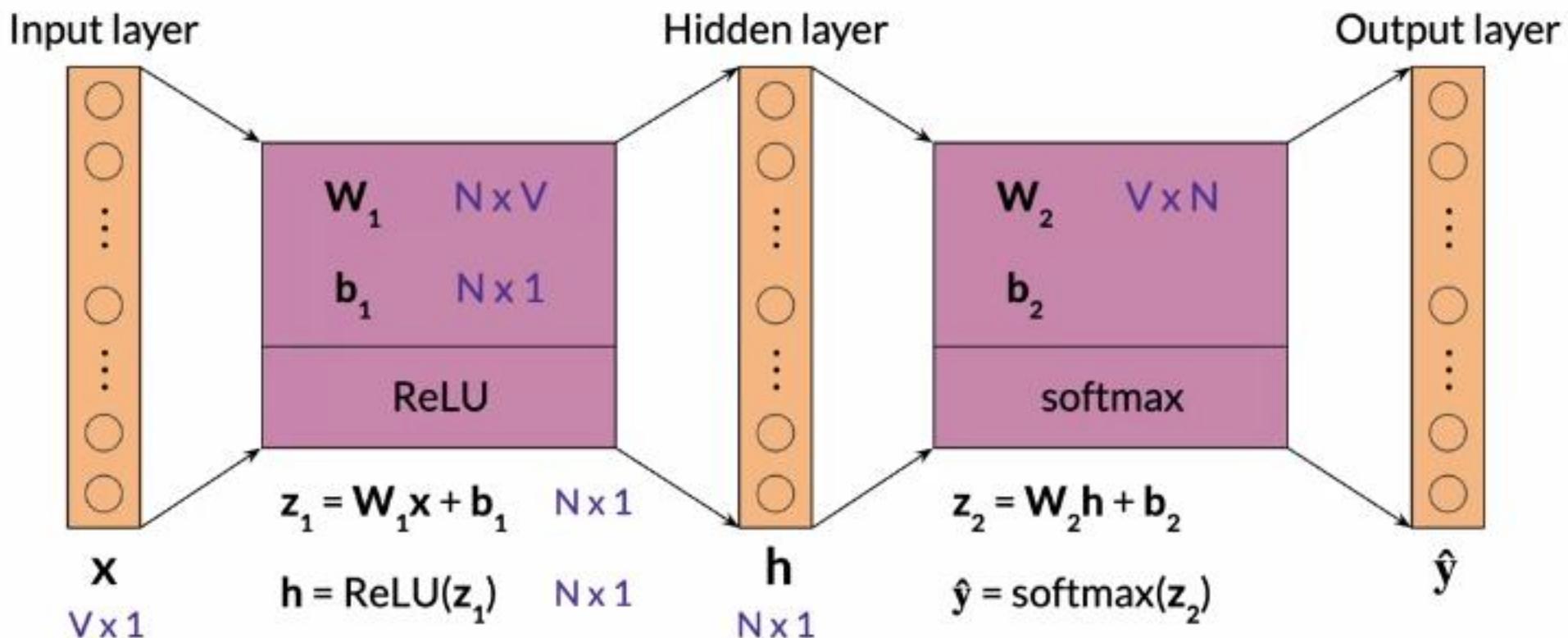
Dimensions (single input)



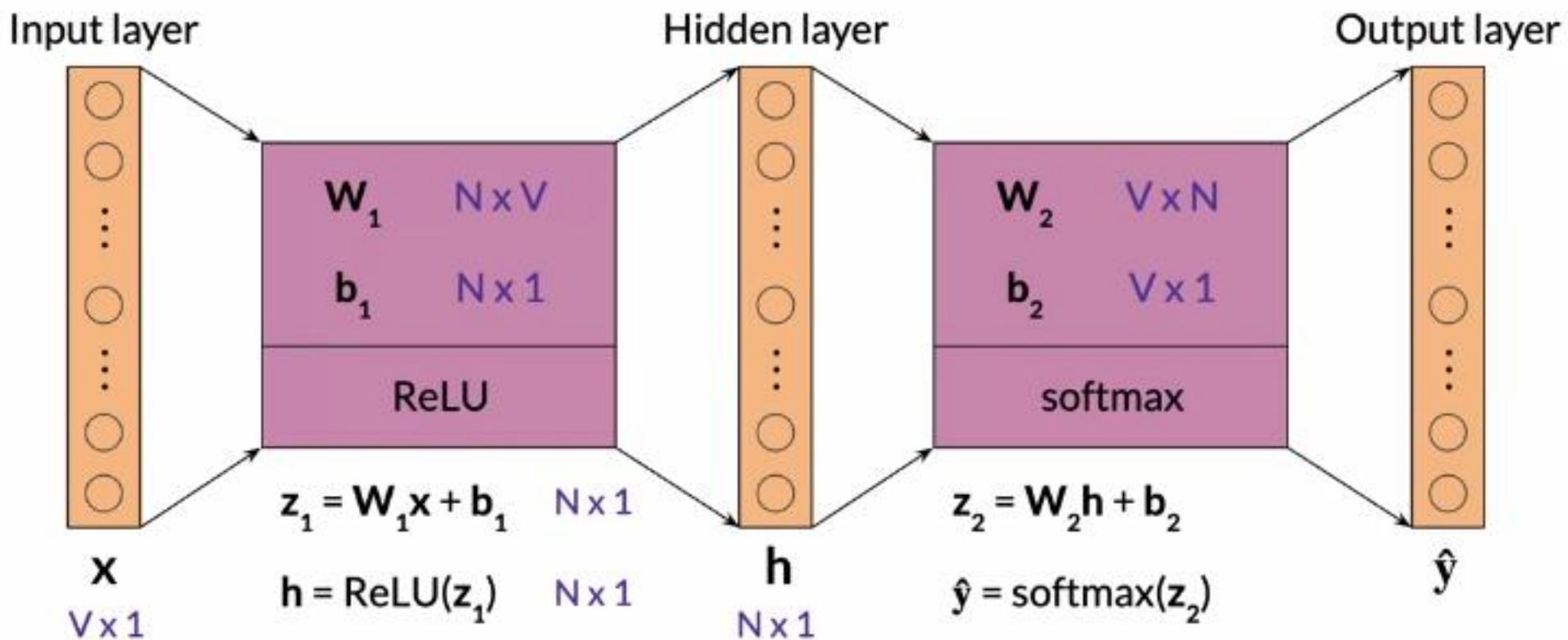
Dimensions (single input)



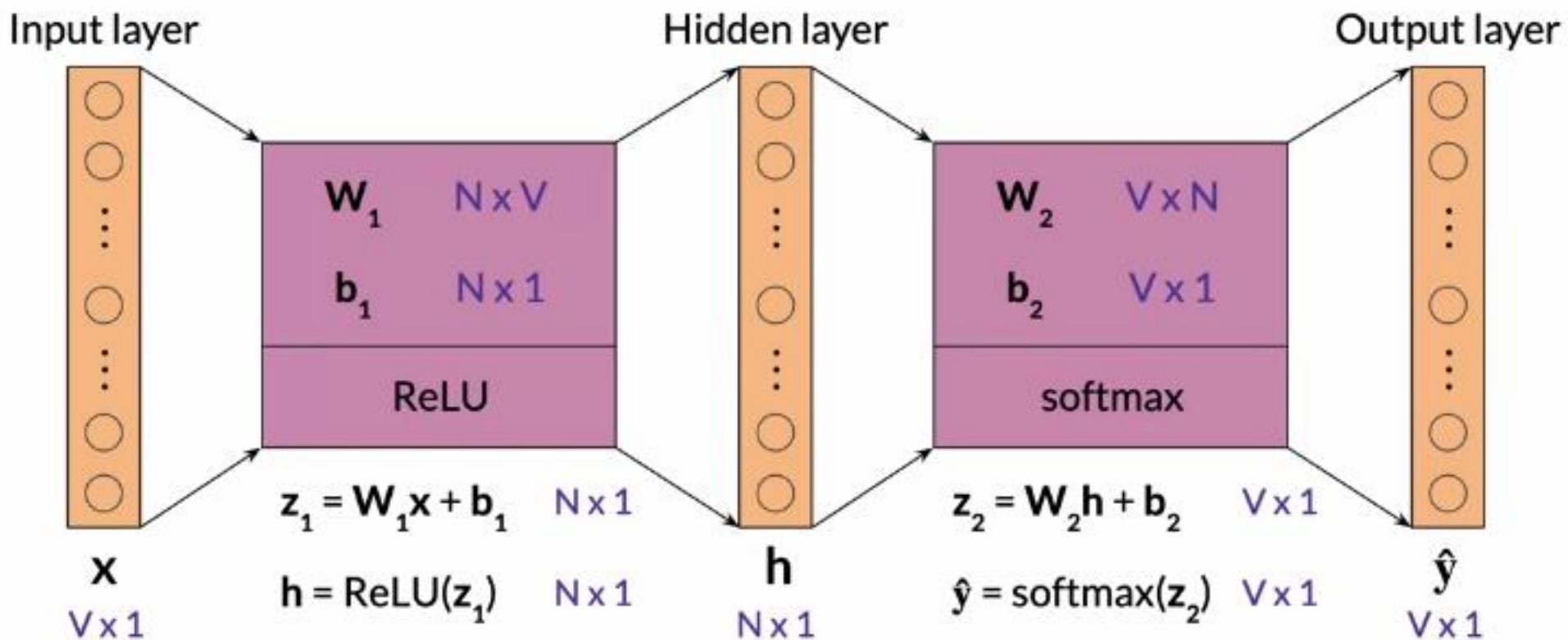
Dimensions (single input)



Dimensions (single input)



Dimensions (single input)



Dimensions (single input)

Column vectors

$$\mathbf{z}_1 = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1$$

$$\mathbf{z}_1 = \begin{bmatrix} \quad \\ \quad \end{bmatrix} \quad \mathbf{W}_1 = \begin{bmatrix} \quad & \mathbf{N} \times \mathbf{V} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \quad \\ \quad \end{bmatrix} \quad \mathbf{b}_1 = \begin{bmatrix} \quad \\ \quad \end{bmatrix}$$

$\mathbf{N} \times 1$ $\mathbf{V} \times 1$ $\mathbf{N} \times 1$

Dimensions (single input)

Column vectors

$$\mathbf{z}_1 = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1$$

$$\mathbf{z}_1 = \begin{bmatrix} \quad \\ \quad \end{bmatrix} \quad \mathbf{W}_1 = \begin{bmatrix} \quad & \mathbf{N} \times \mathbf{V} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \quad \\ \quad \end{bmatrix} \quad \mathbf{b}_1 = \begin{bmatrix} \quad \\ \quad \end{bmatrix}$$

$\mathbf{N} \times 1$ $\mathbf{V} \times 1$ $\mathbf{N} \times 1$

Dimensions (single input)

Column vectors

$$\mathbf{z}_1 = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1$$

$$\mathbf{z}_1 = \begin{bmatrix} \quad \\ \quad \end{bmatrix} \quad \mathbf{W}_1 = \begin{bmatrix} \quad & \mathbf{N} \times \mathbf{V} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \quad \\ \quad \end{bmatrix} \quad \mathbf{b}_1 = \begin{bmatrix} \quad \\ \quad \end{bmatrix}$$

$\mathbf{N} \times 1$ $\mathbf{V} \times 1$ $\mathbf{N} \times 1$

Row vectors

$$\mathbf{z}_1 = \mathbf{x} \mathbf{W}_1^T + \mathbf{b}_1$$

$$\mathbf{b}_1 = \begin{bmatrix} \mathbf{1} \times \mathbf{N} \end{bmatrix} \quad \mathbf{W}_1 = \begin{bmatrix} \quad & \mathbf{N} \times \mathbf{V} \end{bmatrix} \quad \mathbf{b}_1 = \begin{bmatrix} \mathbf{1} \times \mathbf{N} \end{bmatrix}$$
$$\mathbf{x} = \begin{bmatrix} \quad & \mathbf{1} \times \mathbf{V} \end{bmatrix}$$

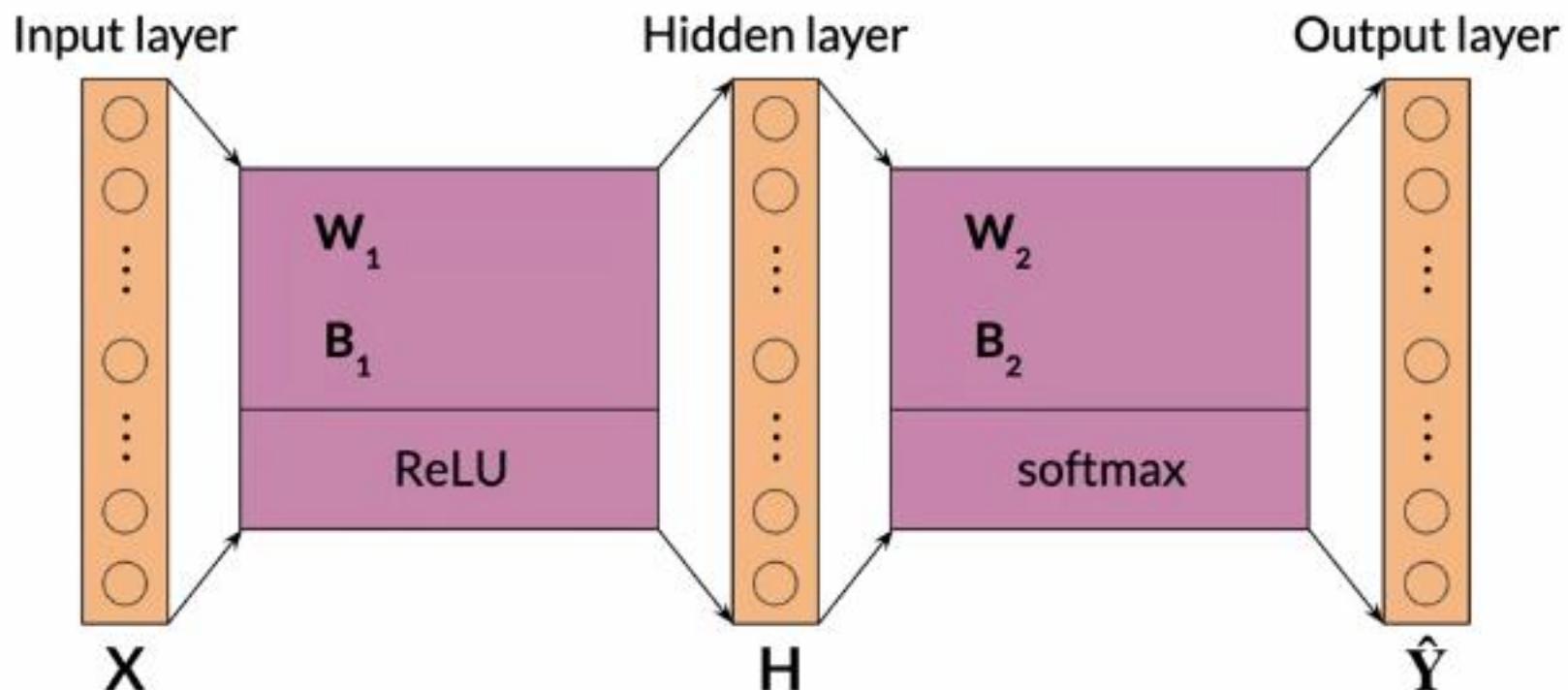


deeplearning.ai

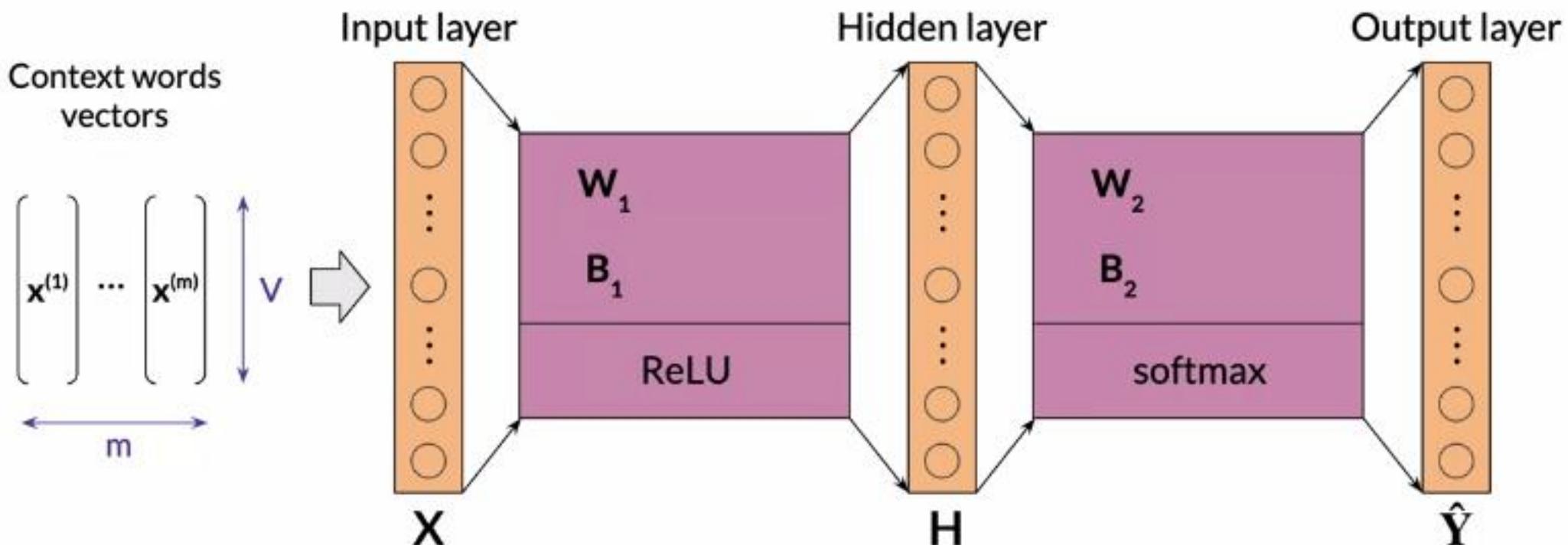
Architecture of the CBOW Model:

Dimensions 2

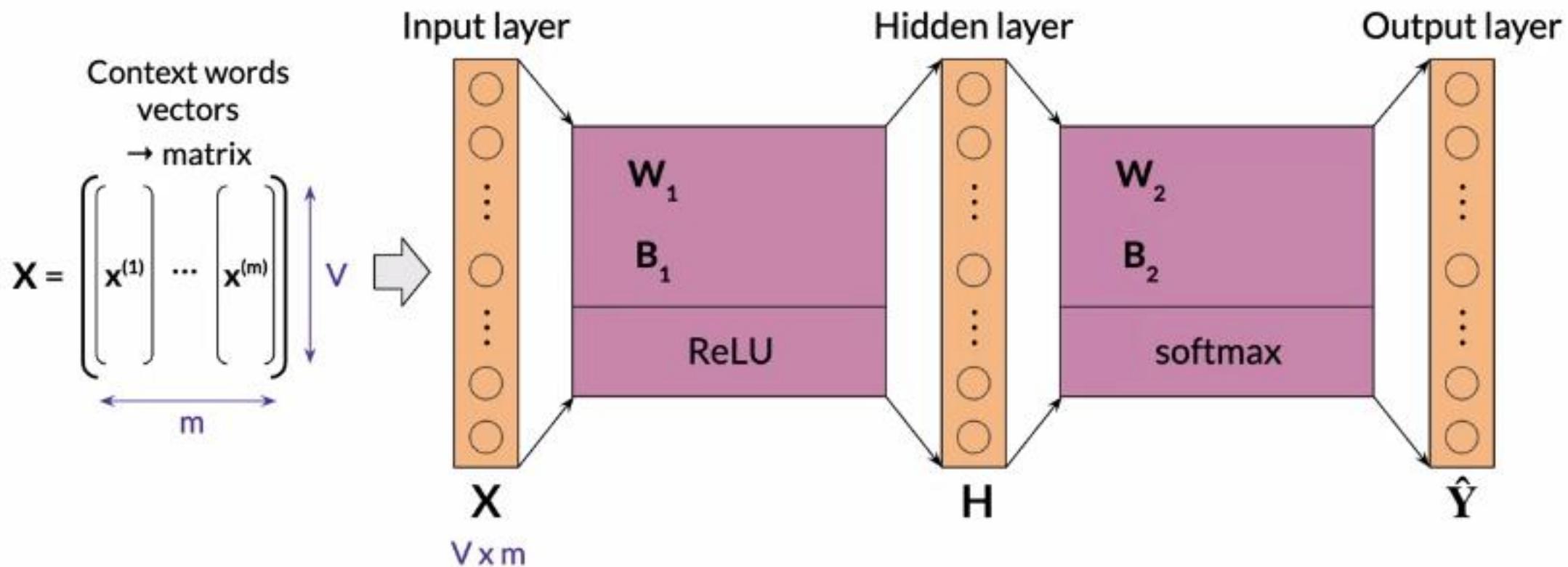
Dimensions (batch input)



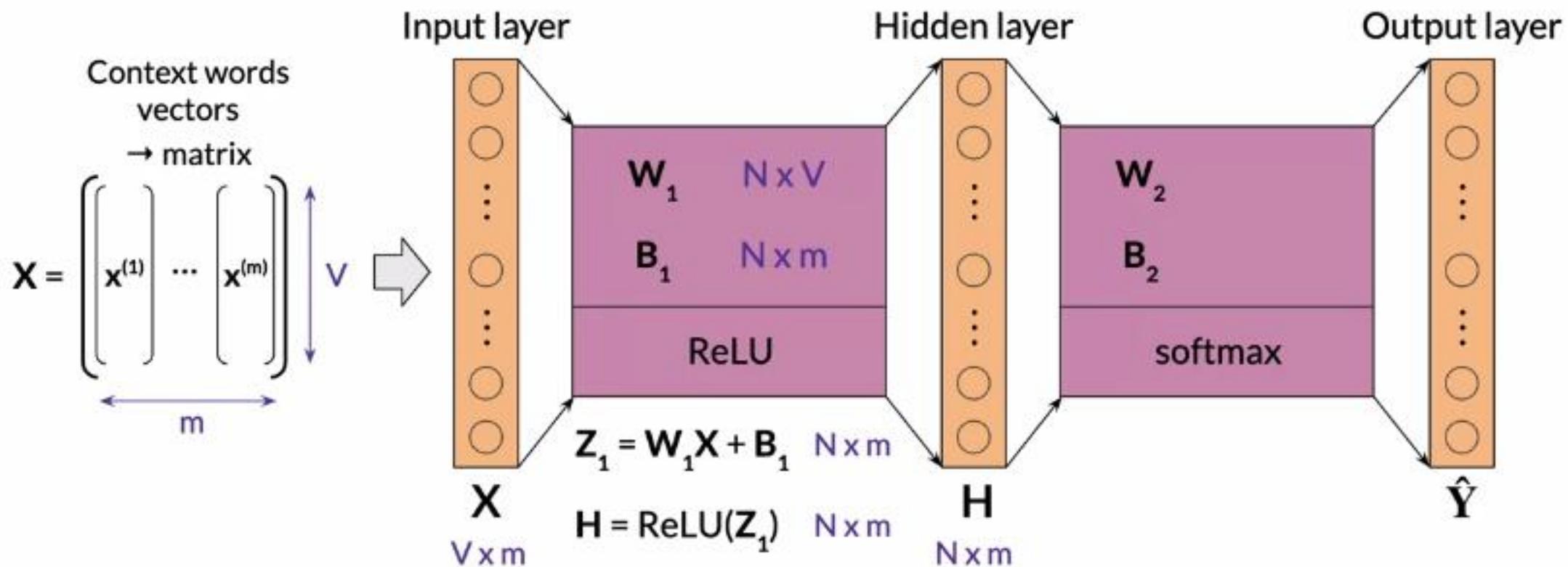
Dimensions (batch input)



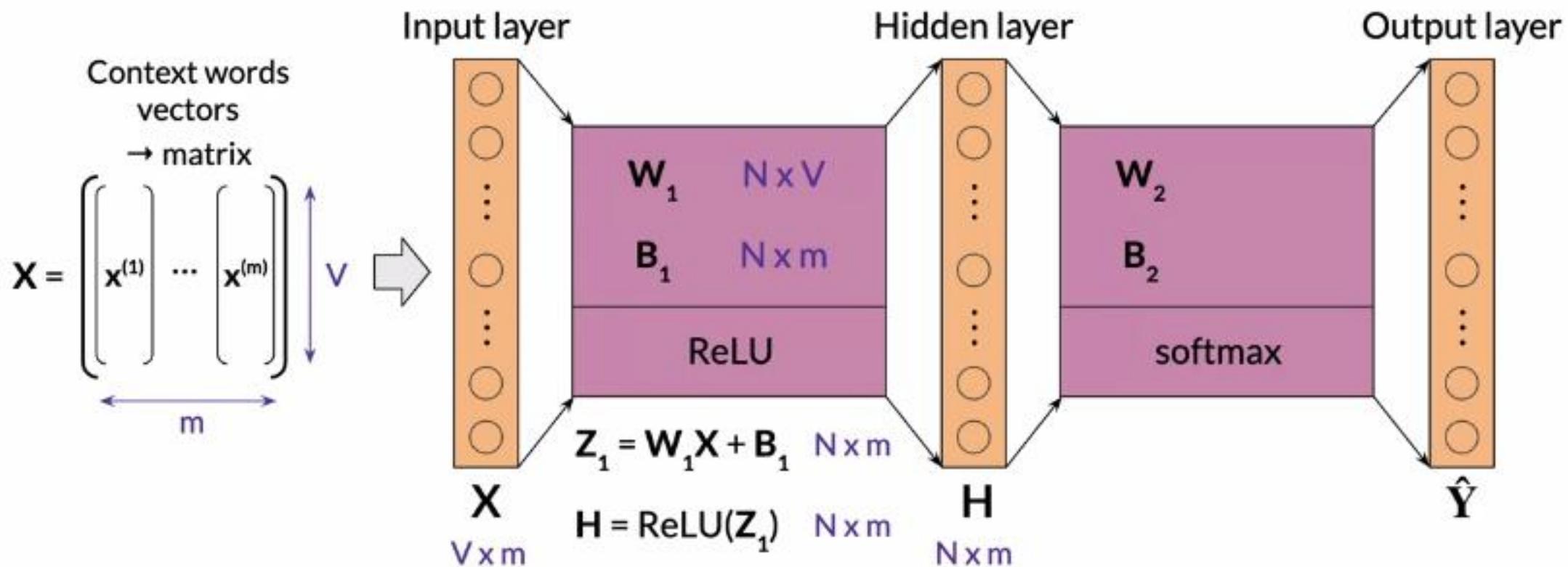
Dimensions (batch input)



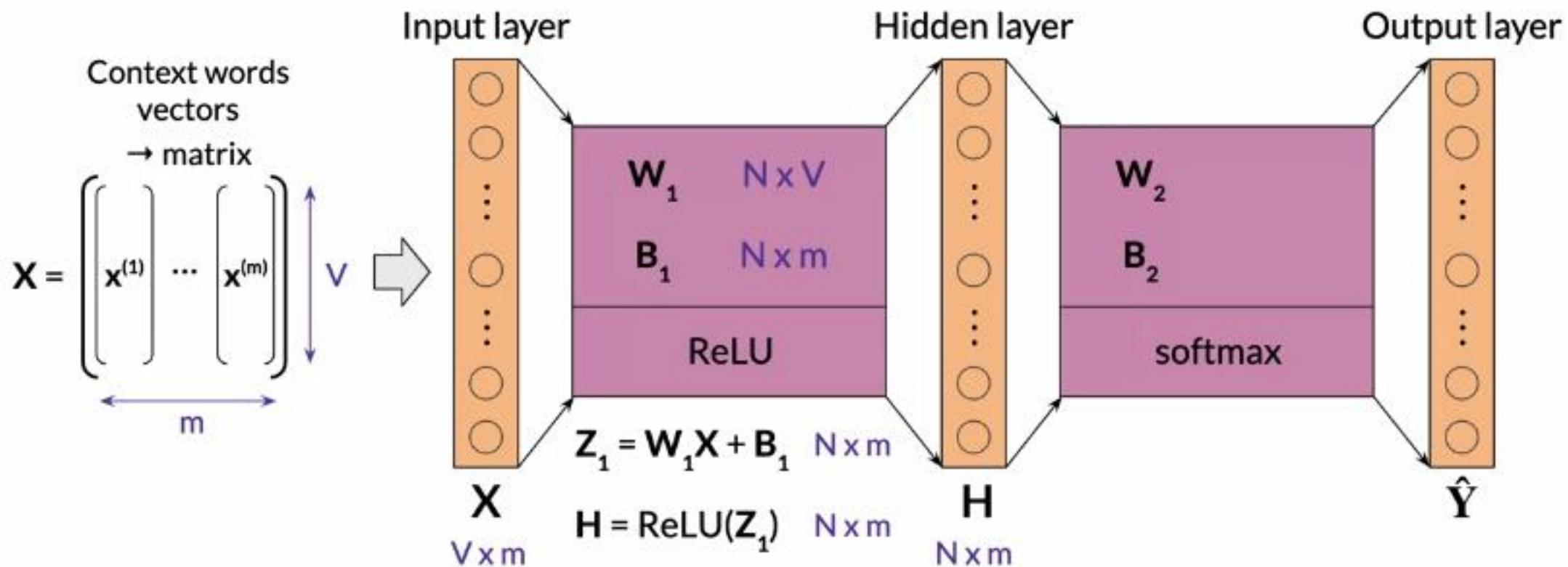
Dimensions (batch input)



Dimensions (batch input)

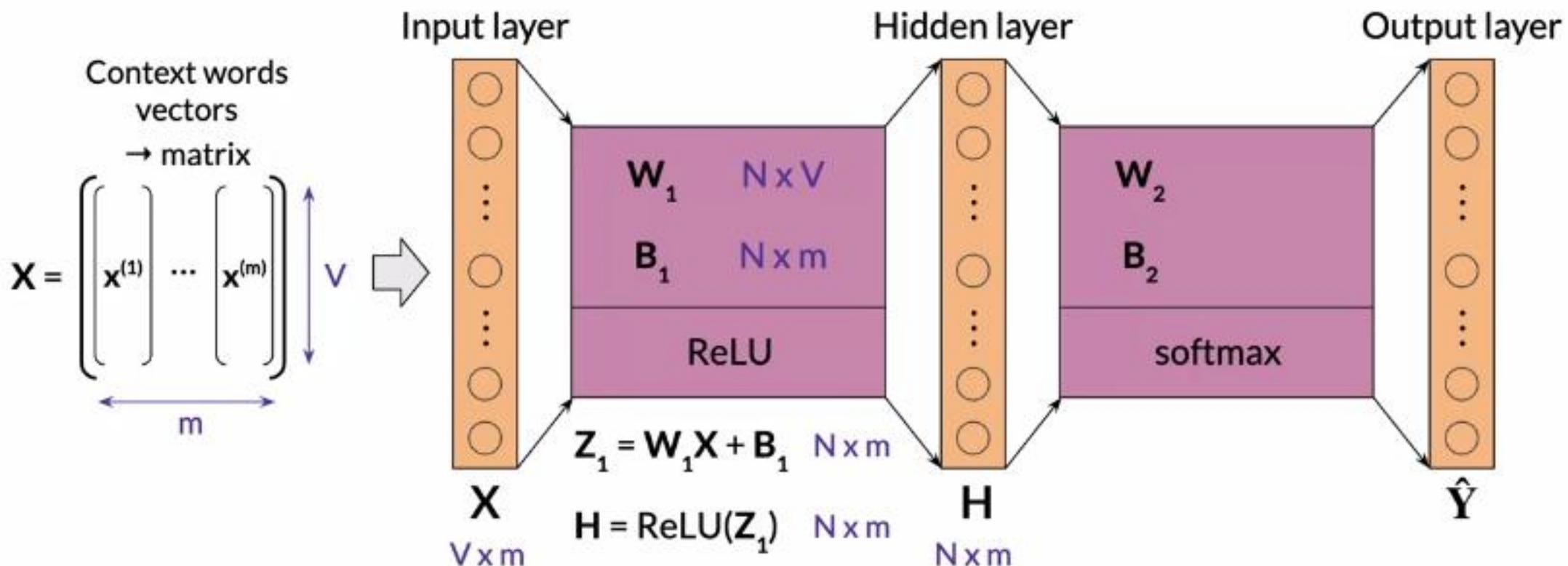


Dimensions (batch input)



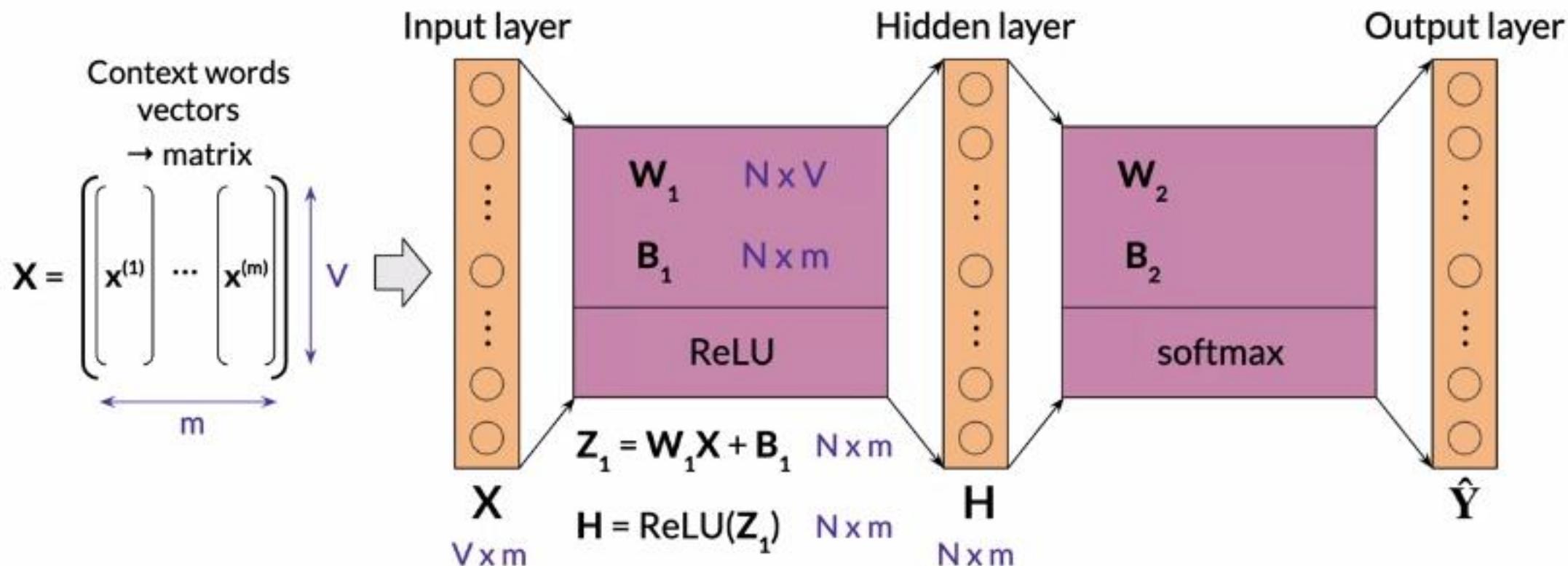
Dimensions (batch input)

$$\begin{bmatrix} \mathbf{b}_1 \end{bmatrix} \rightarrow \mathbf{B}_1 = \begin{bmatrix} \begin{bmatrix} \mathbf{b}_1 \end{bmatrix} & \dots & \begin{bmatrix} \mathbf{b}_1 \end{bmatrix} \end{bmatrix} \underset{m}{\underbrace{\qquad\qquad\qquad}} \underset{N}{\updownarrow}$$



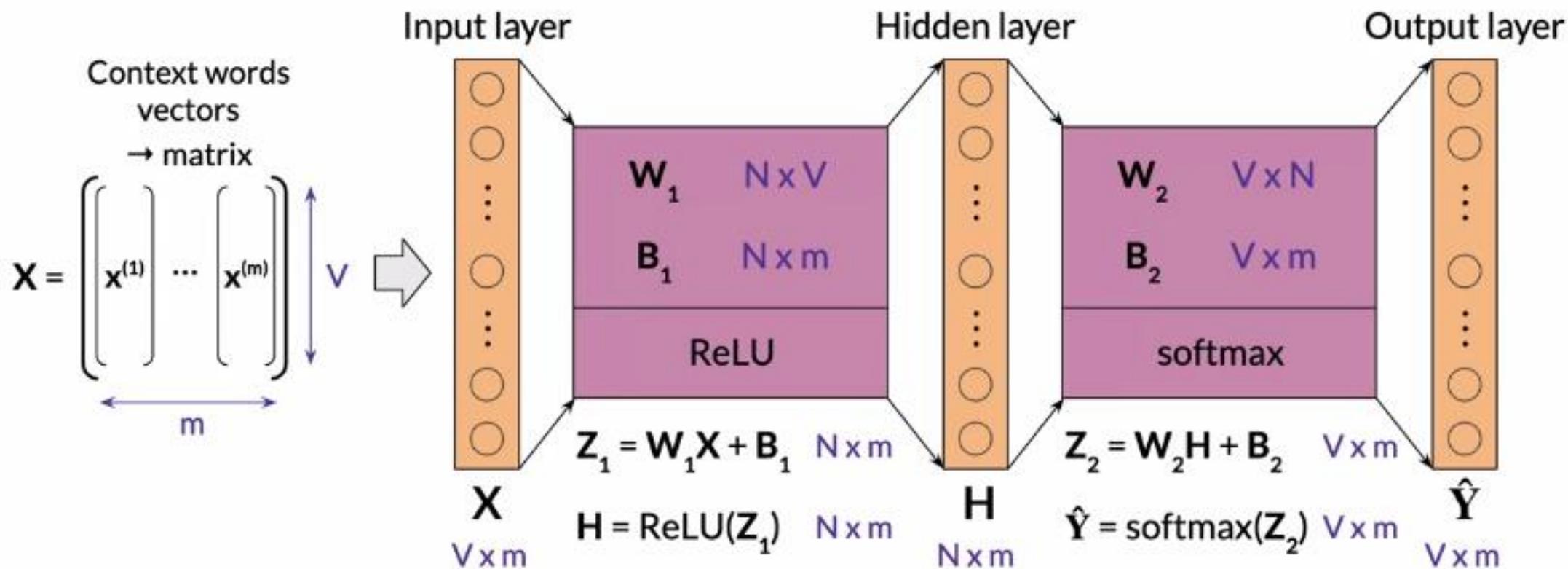
Dimensions (batch input)

$$\begin{bmatrix} \mathbf{b}_1 \end{bmatrix} \rightarrow \mathbf{B}_1 = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_1 \end{bmatrix} \cdots \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_1 \end{bmatrix} \xleftarrow[m]{N} \text{broadcasting}$$

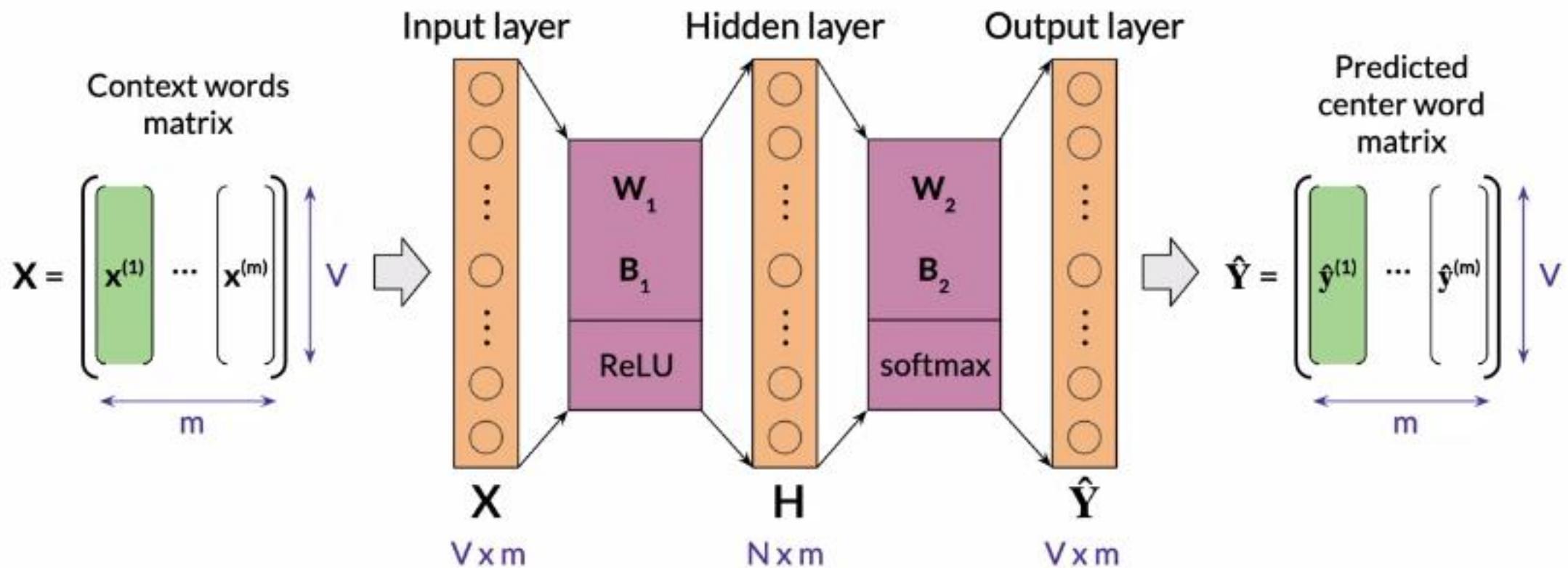


Dimensions (batch input)

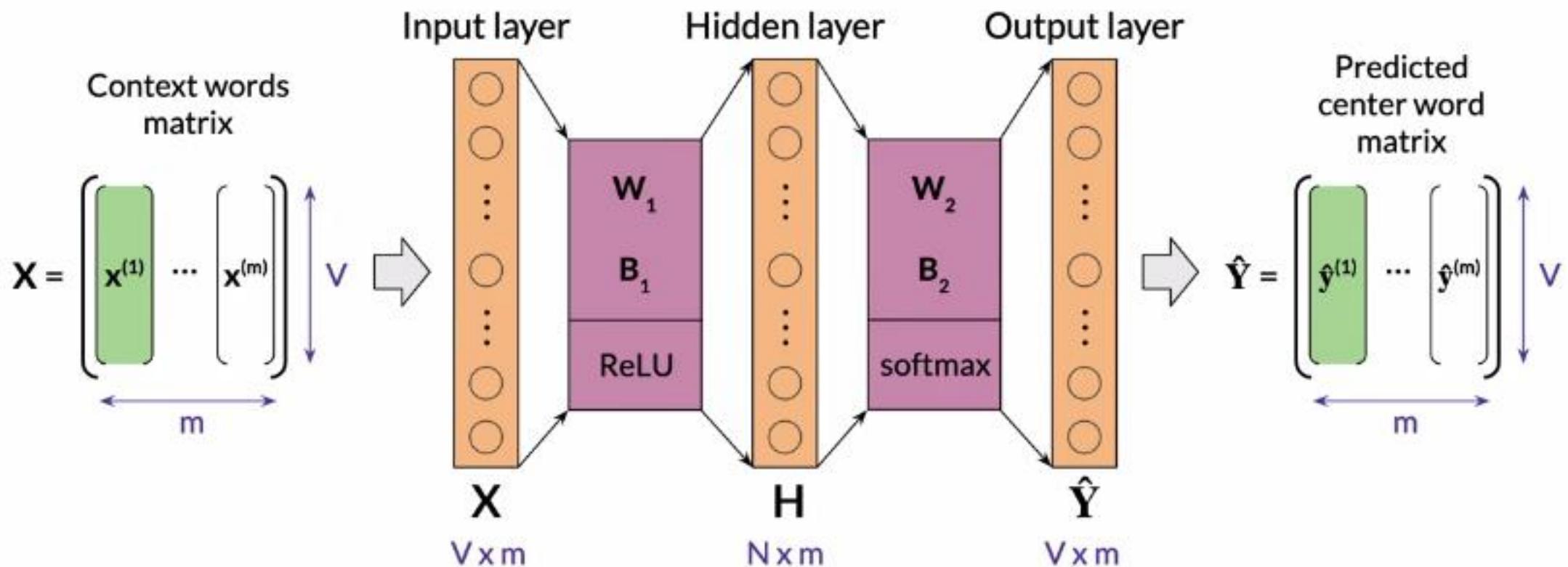
$$\begin{bmatrix} \mathbf{b}_1 \end{bmatrix} \rightarrow \mathbf{B}_1 = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_1 \end{bmatrix} \cdots \begin{bmatrix} \mathbf{b}_1 \end{bmatrix} \xleftarrow[m]{\text{N}} \text{broadcasting}$$



Dimensions (batch input)



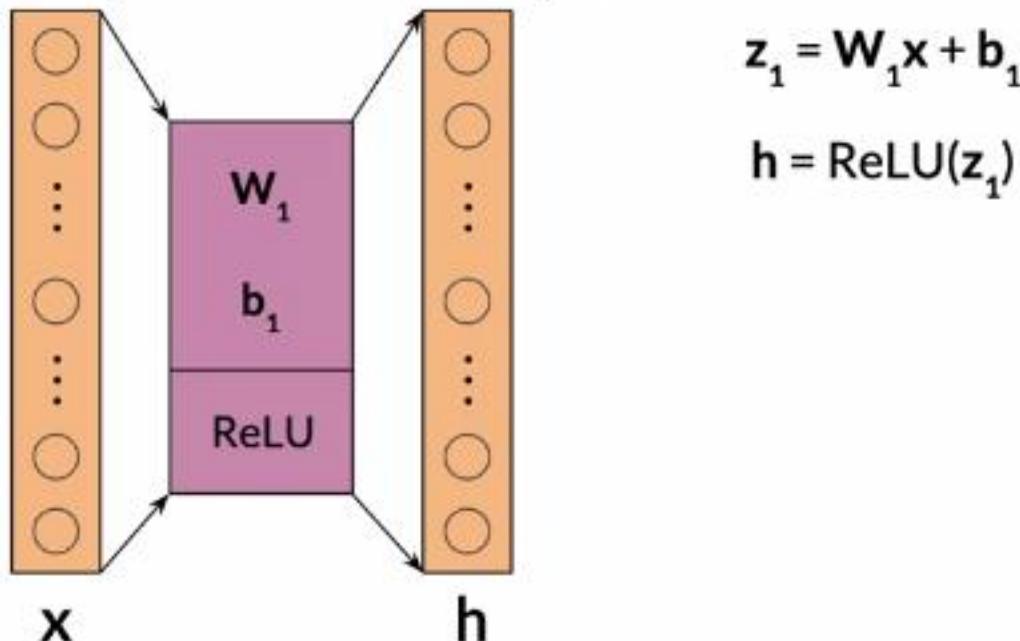
Dimensions (batch input)



Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU)

Input layer Hidden layer



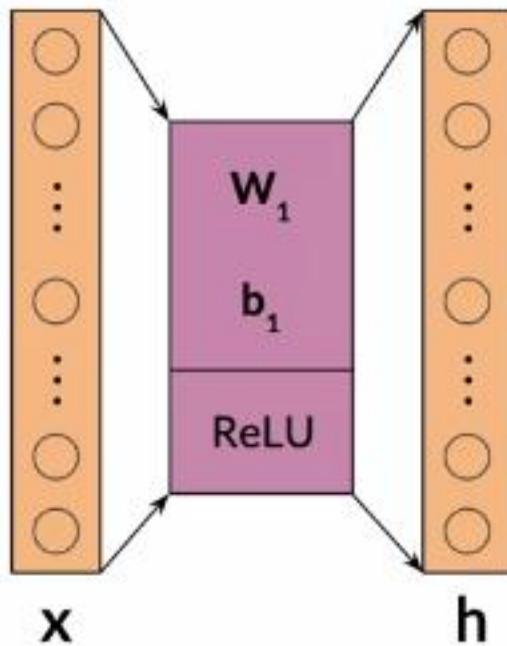
$$z_1 = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1$$

$$\mathbf{h} = \text{ReLU}(z_1)$$

Rectified Linear Unit (ReLU)

Input layer

Hidden layer



$$\text{ReLU}(x) = \max(0, x)$$

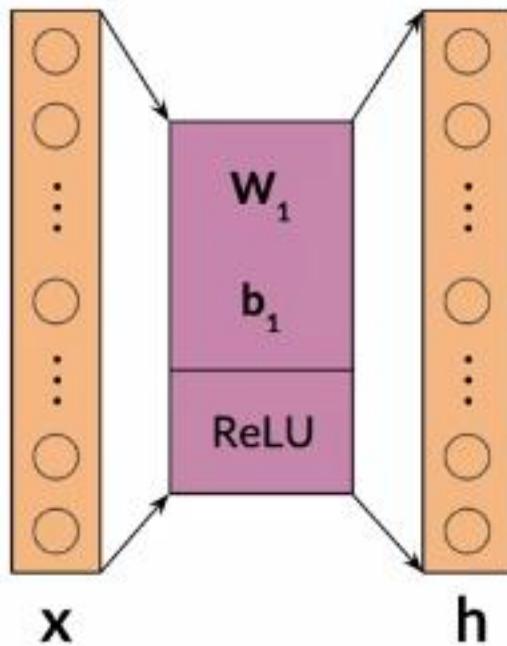
$$z_1 = W_1 x + b_1$$

$$h = \text{ReLU}(z_1)$$

Rectified Linear Unit (ReLU)

Input layer

Hidden layer



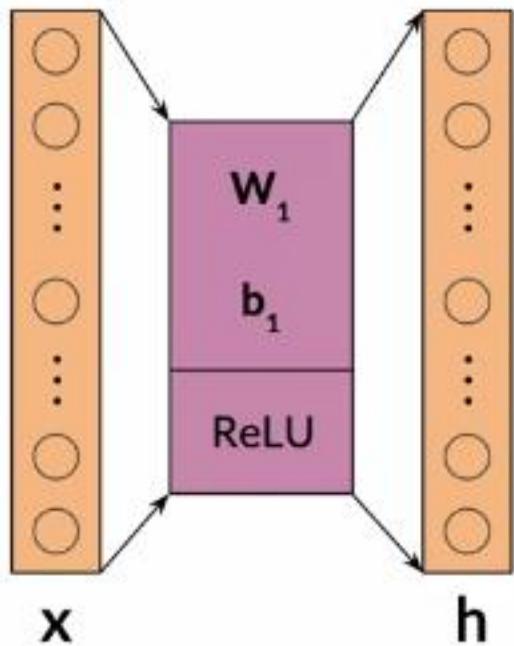
$$\text{ReLU}(x) = \max(0, x)$$

$$z_1 = W_1 x + b_1$$

$$h = \text{ReLU}(z_1)$$

Rectified Linear Unit (ReLU)

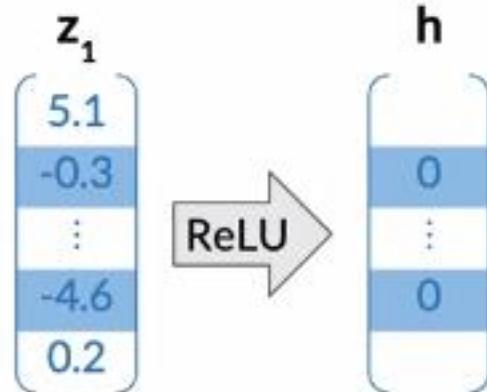
Input layer



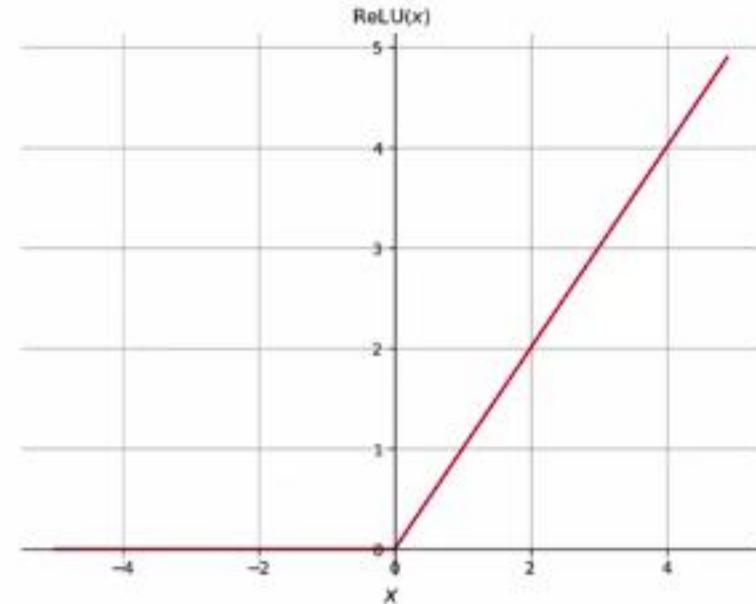
Hidden layer

$$z_1 = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1$$

$$\mathbf{h} = \text{ReLU}(z_1)$$

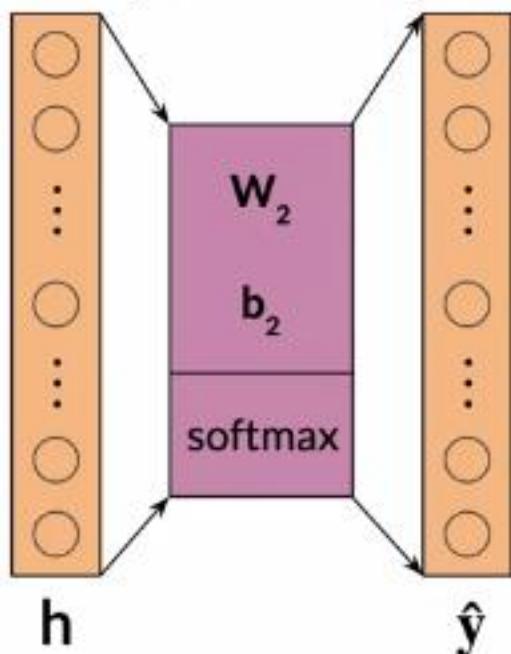


$$\text{ReLU}(x) = \max(0, x)$$



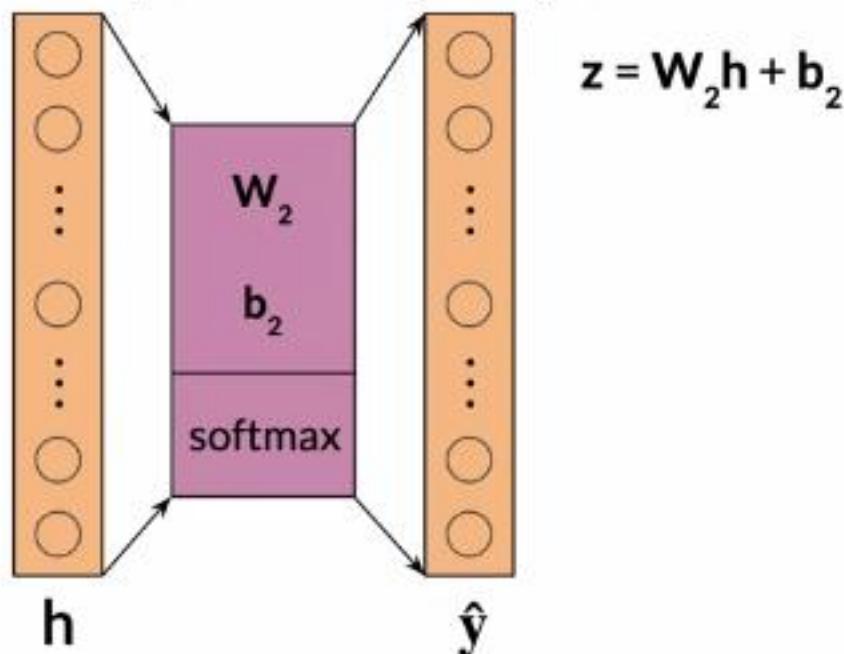
Softmax

Hidden layer Output layer



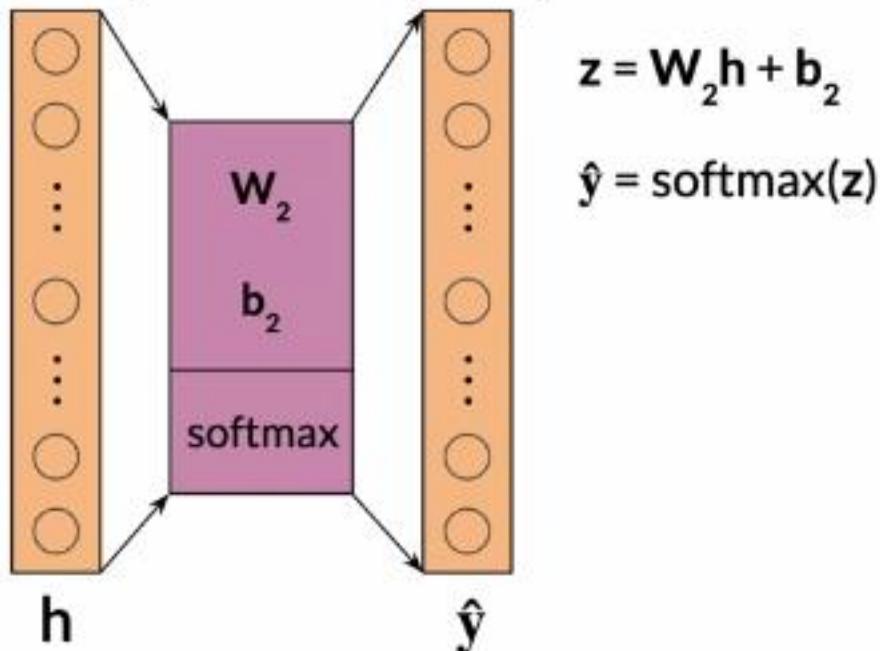
Softmax

Hidden layer Output layer



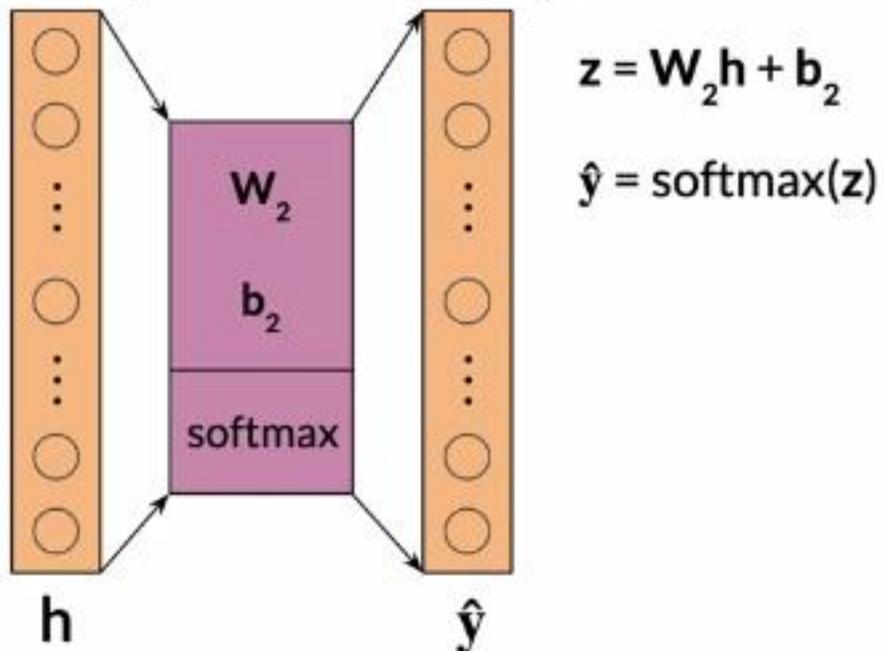
Softmax

Hidden layer Output layer



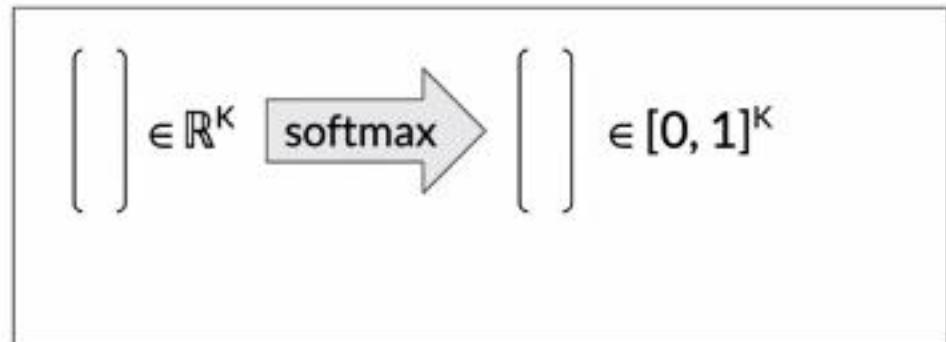
Softmax

Hidden layer Output layer



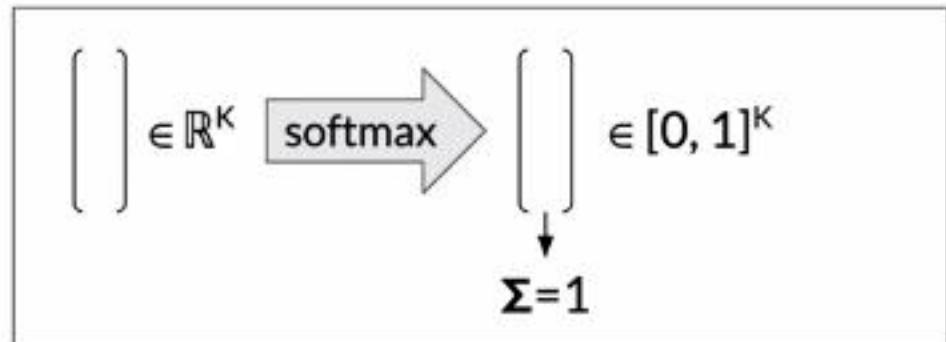
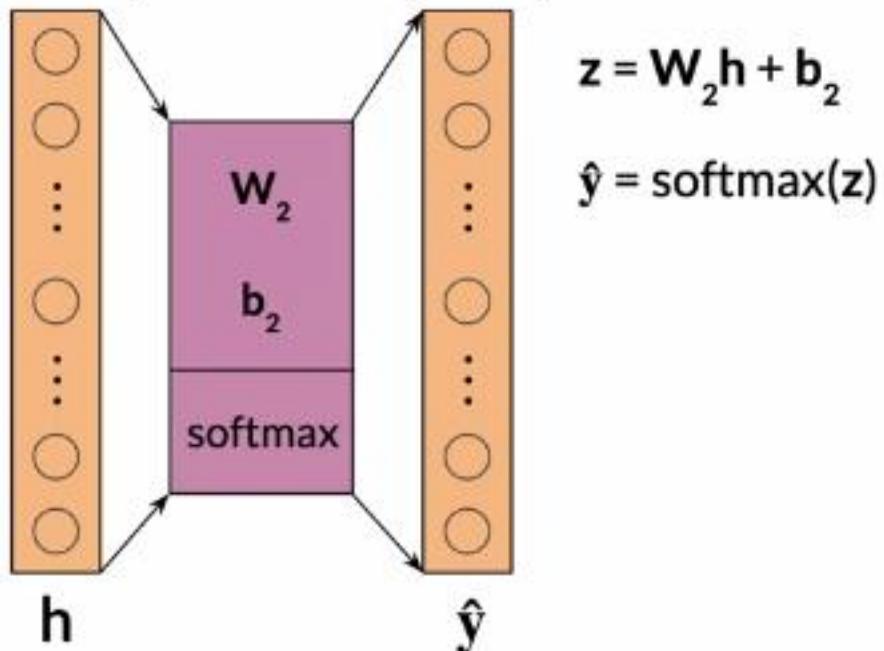
$$z = W_2 h + b_2$$

$$\hat{y} = \text{softmax}(z)$$



Softmax

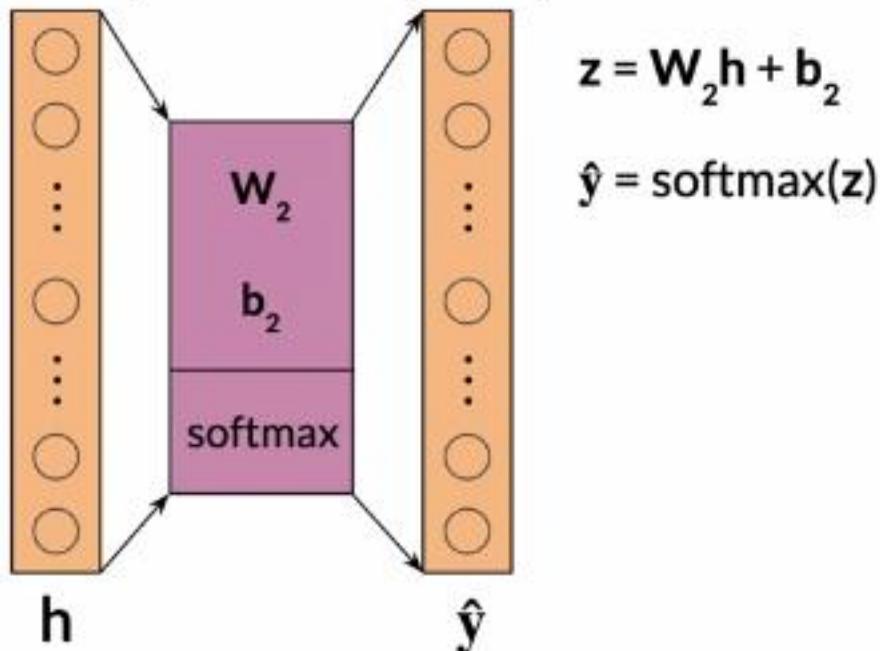
Hidden layer Output layer



Softmax

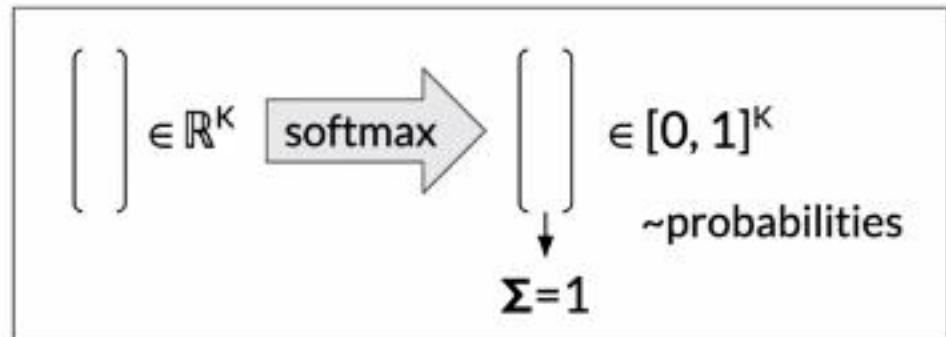
Hidden layer

Output layer



$$z = w_2 h + b_2$$

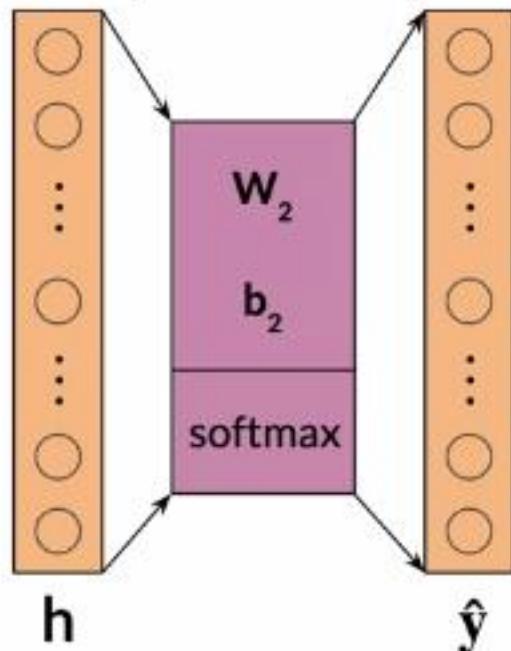
$$\hat{y} = \text{softmax}(z)$$



Softmax

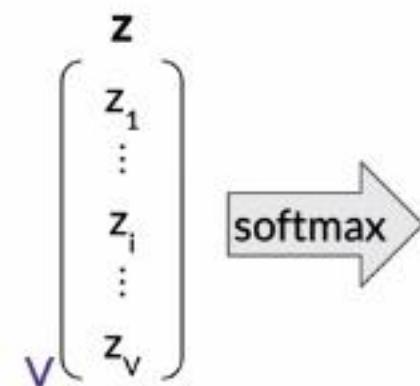
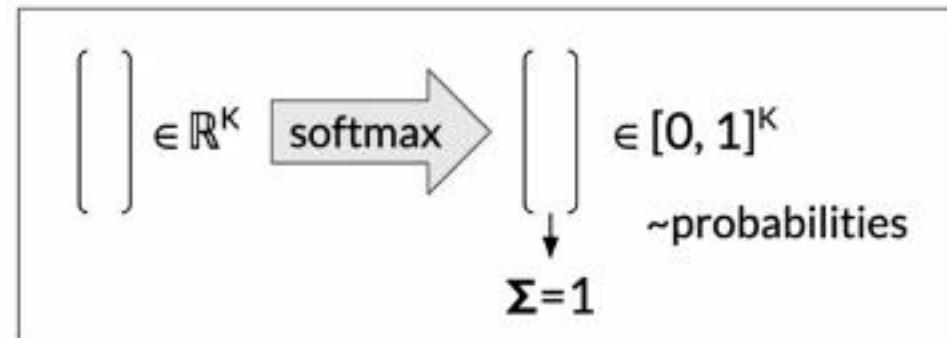
Hidden layer

Output layer



$$z = \mathbf{W}_2 \mathbf{h} + \mathbf{b}_2$$

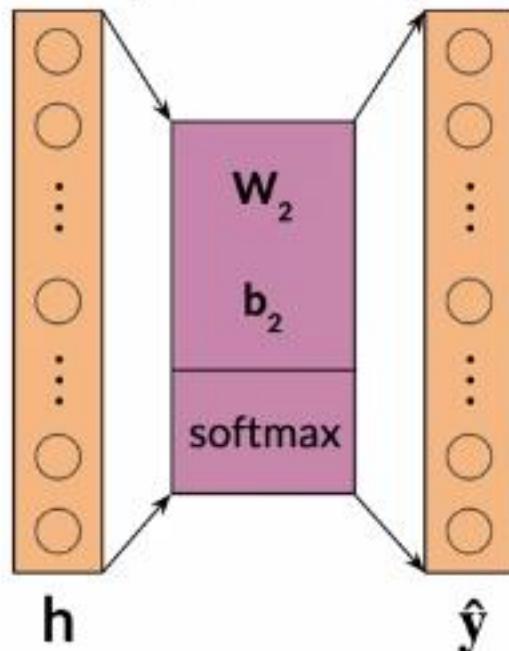
$$\hat{\mathbf{y}} = \text{softmax}(z)$$



Softmax

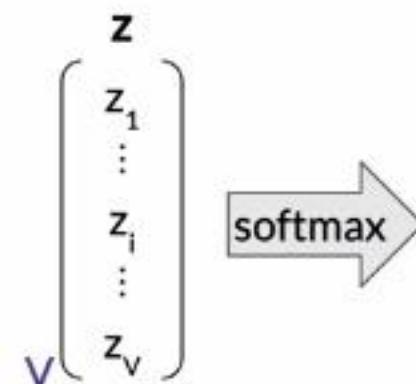
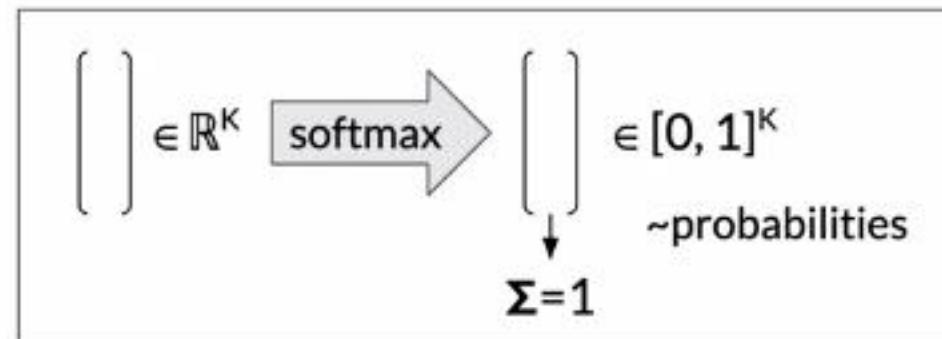
Hidden layer

Output layer



$$z = \mathbf{W}_2 \mathbf{h} + \mathbf{b}_2$$

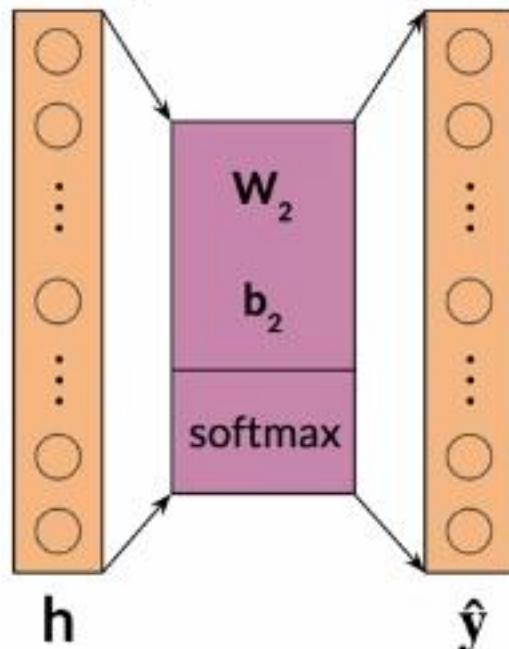
$$\hat{\mathbf{y}} = \text{softmax}(z)$$



Softmax

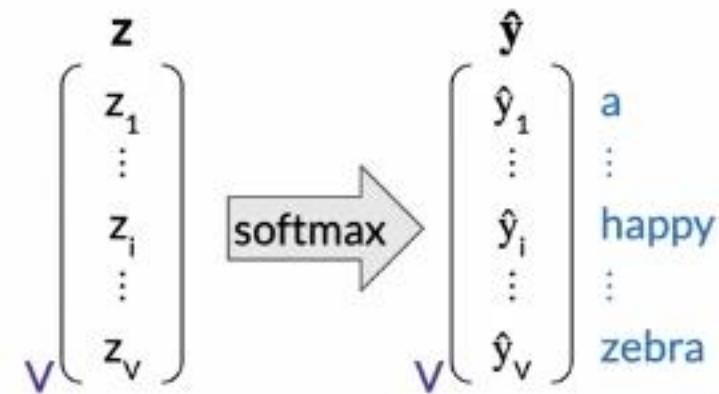
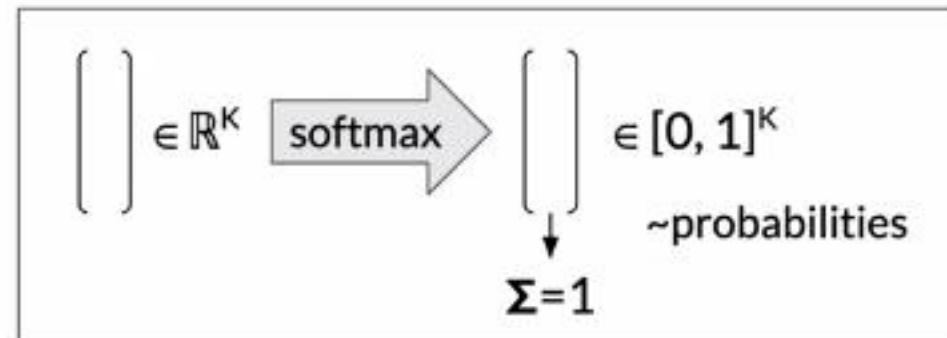
Hidden layer

Output layer



$$z = \mathbf{W}_2 \mathbf{h} + \mathbf{b}_2$$

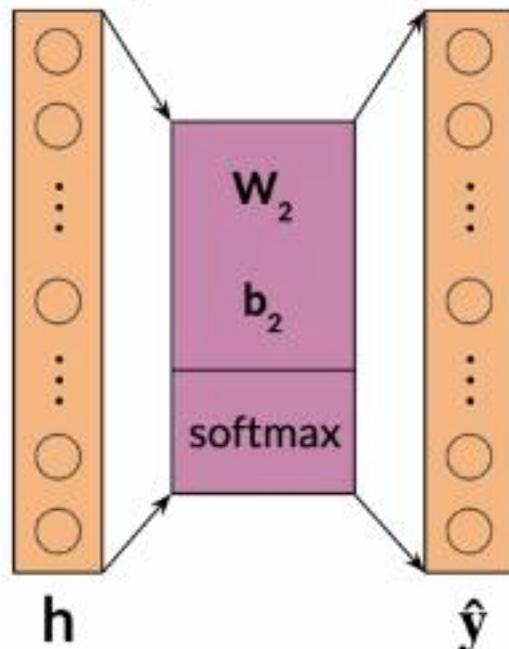
$$\hat{\mathbf{y}} = \text{softmax}(z)$$



Softmax

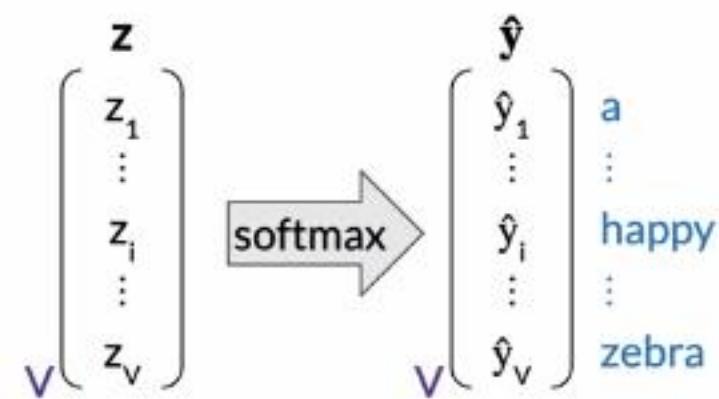
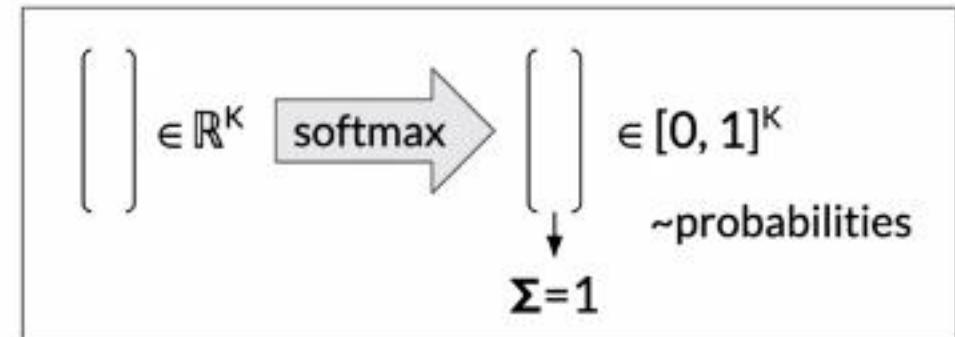
Hidden layer

Output layer



$$z = W_2 h + b_2$$

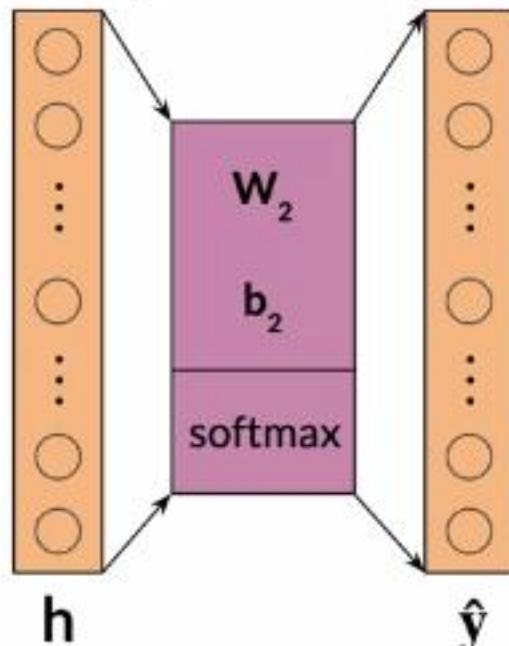
$$\hat{y} = \text{softmax}(z)$$



Probabilities of
being center word

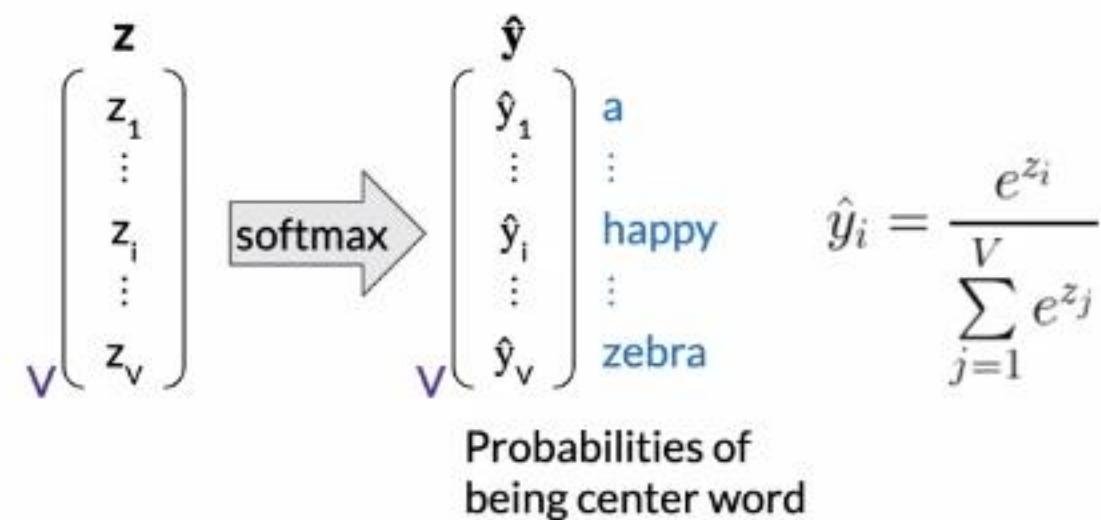
Softmax

Hidden layer Output layer



$$z = W_2 h + b_2$$

$$\hat{y} = \text{softmax}(z)$$

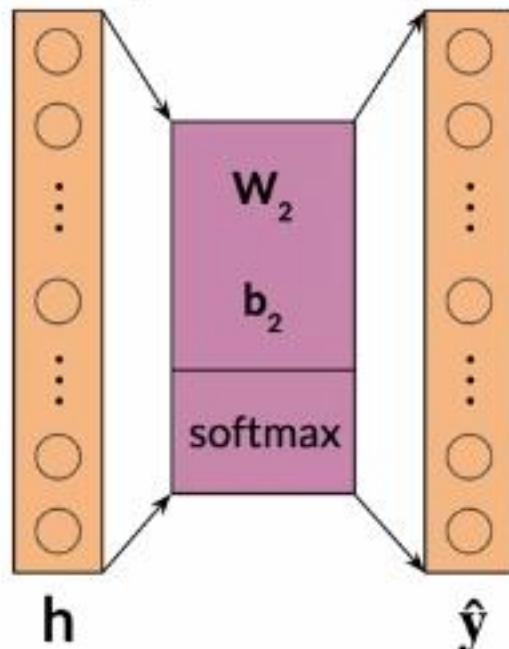


Probabilities of
being center word

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$

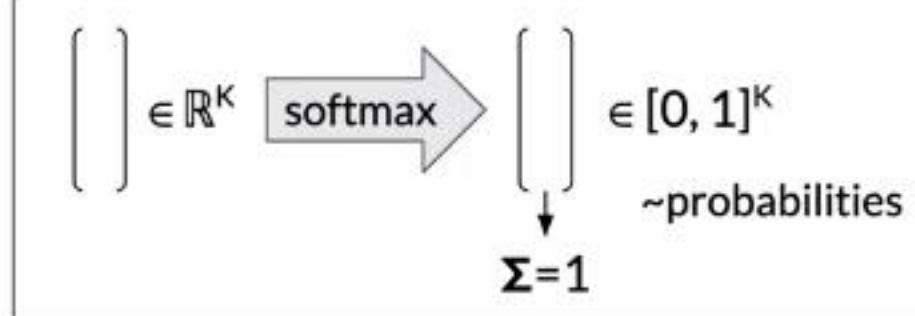
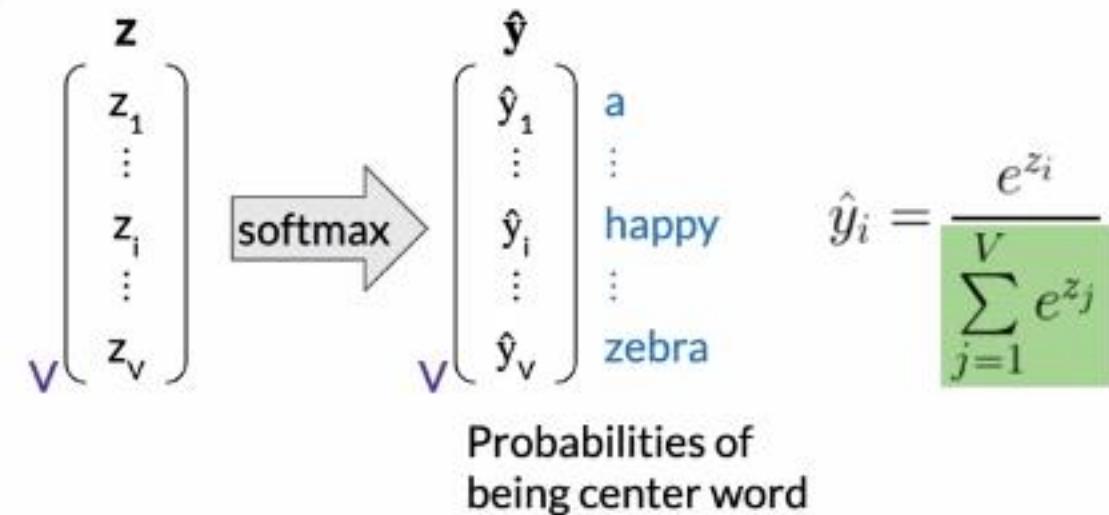
Softmax

Hidden layer Output layer



$$z = W_2 h + b_2$$

$$\hat{y} = \text{softmax}(z)$$



Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$

z

$$\begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix}$$

Softmax: example

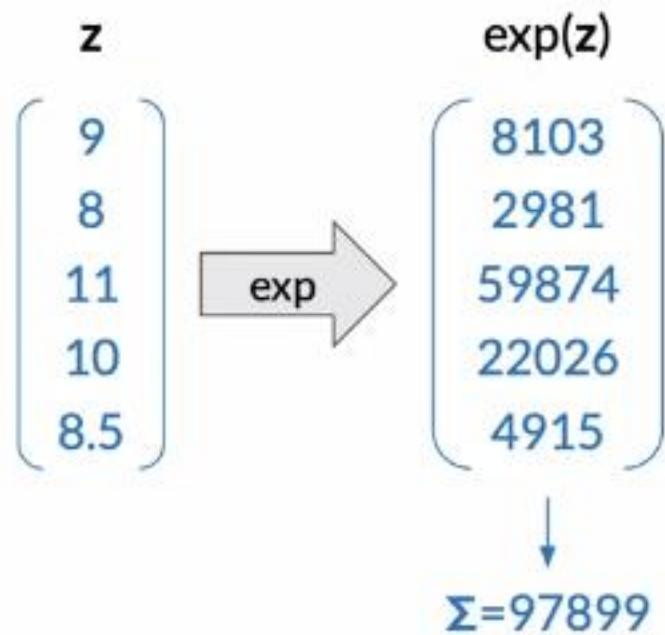
$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$

z	$\exp(z)$
9	8103
8	2981
11	59874
10	22026
8.5	4915

 \exp

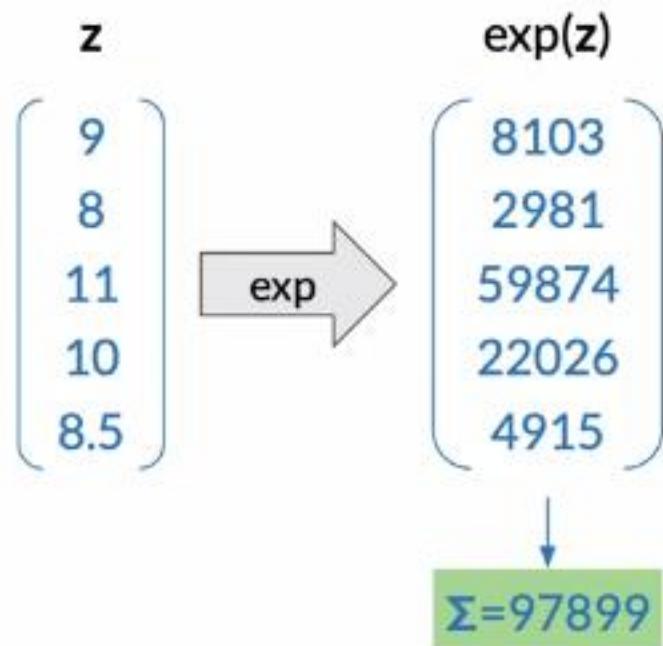
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



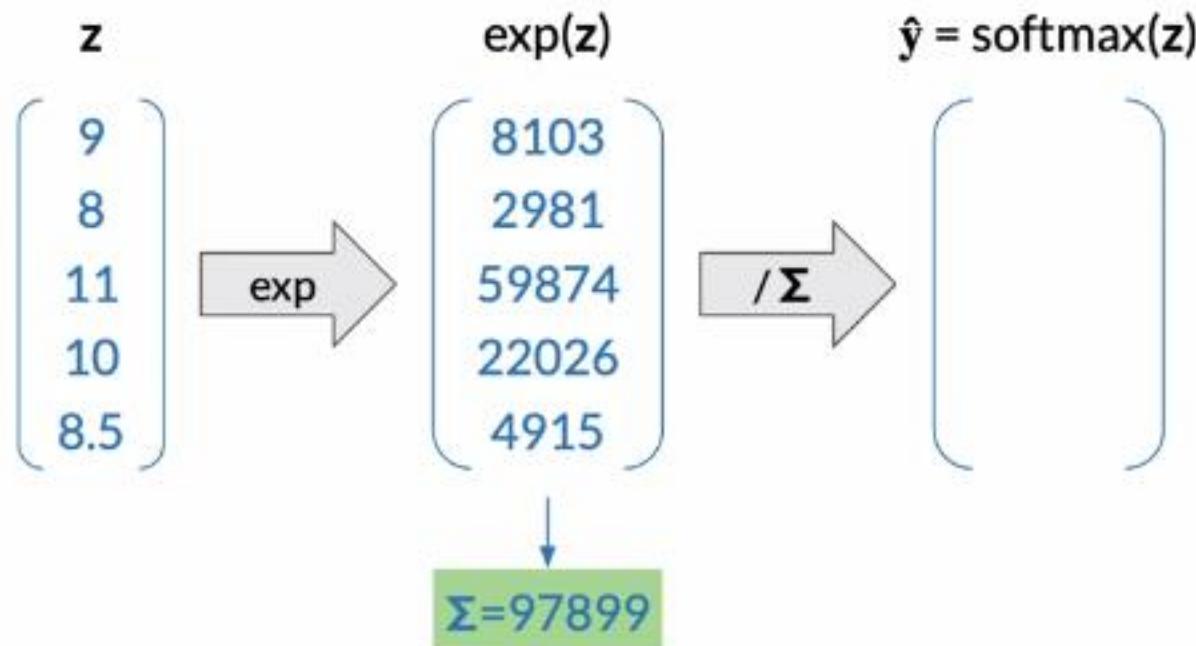
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



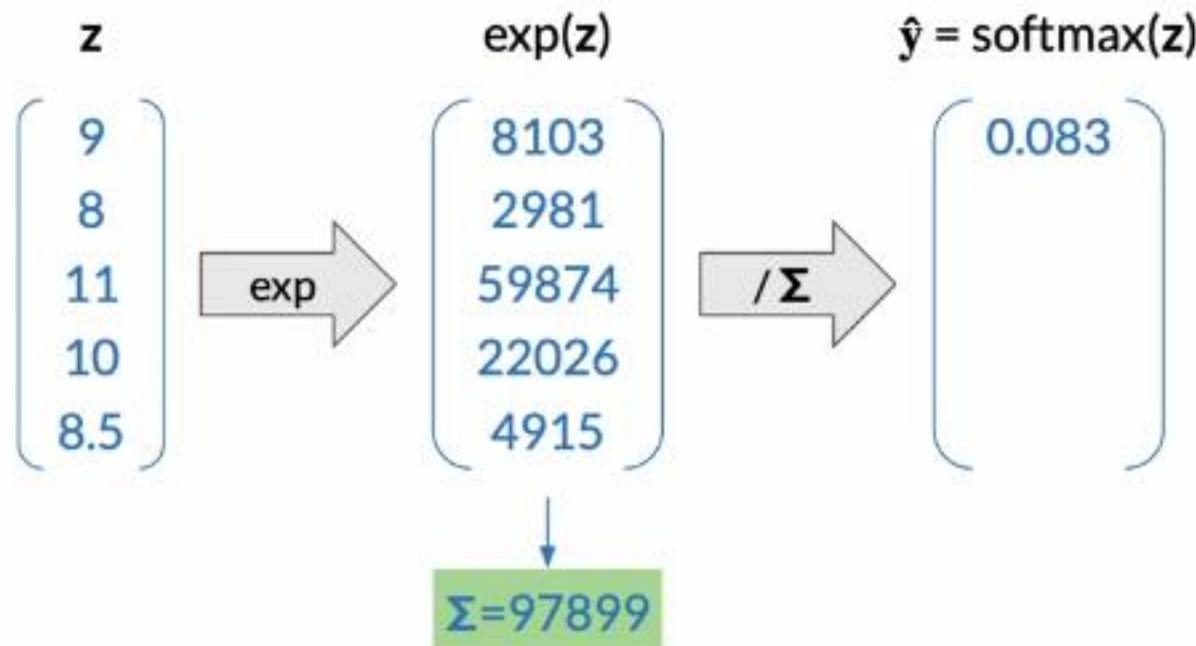
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



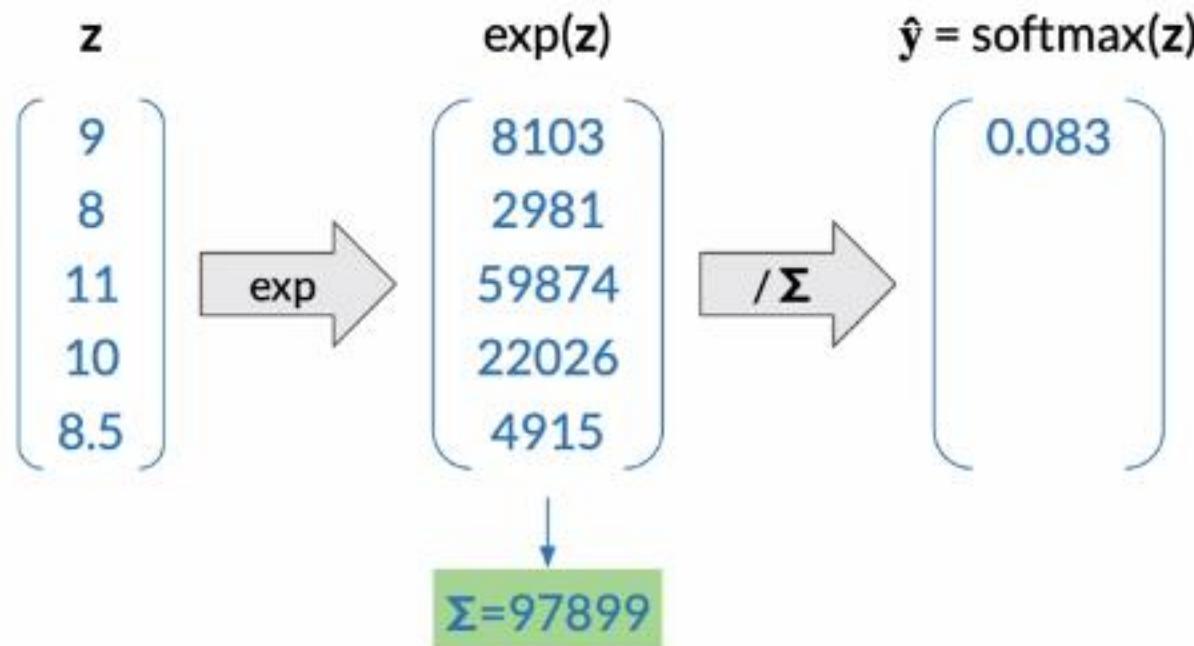
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



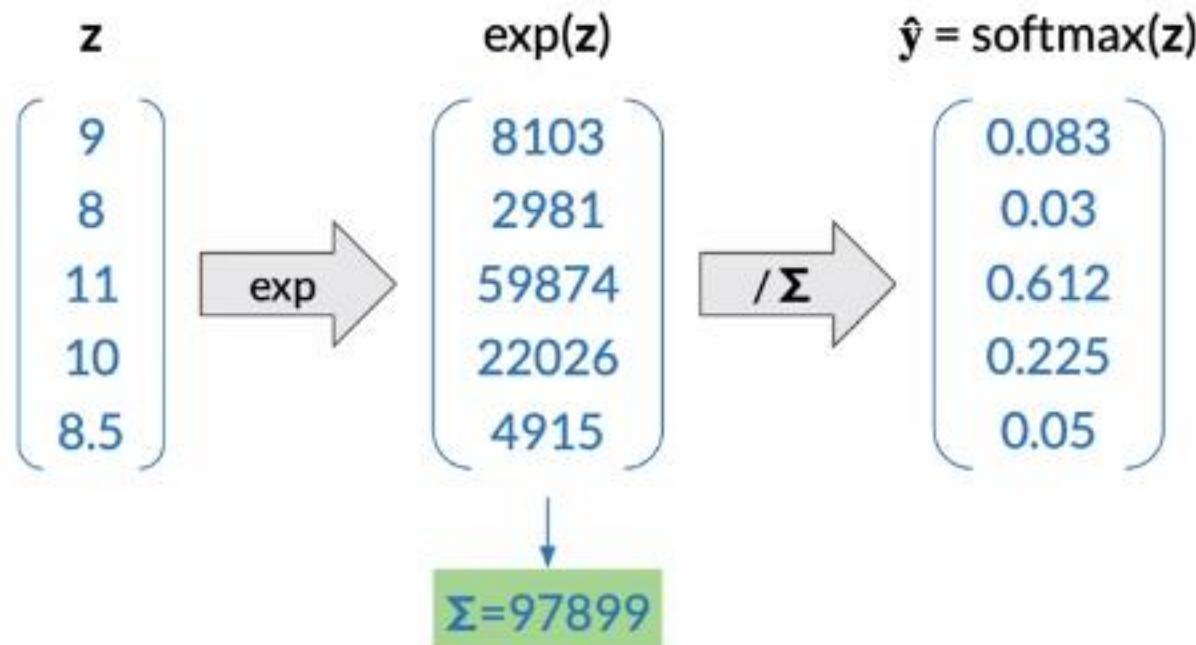
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



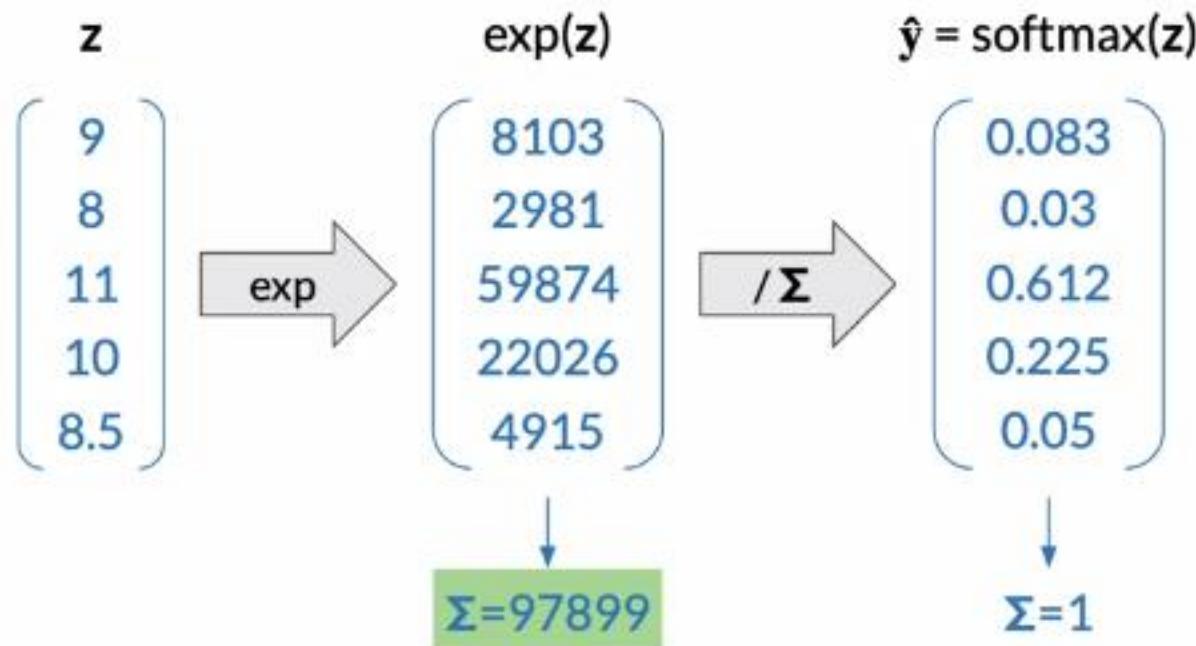
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



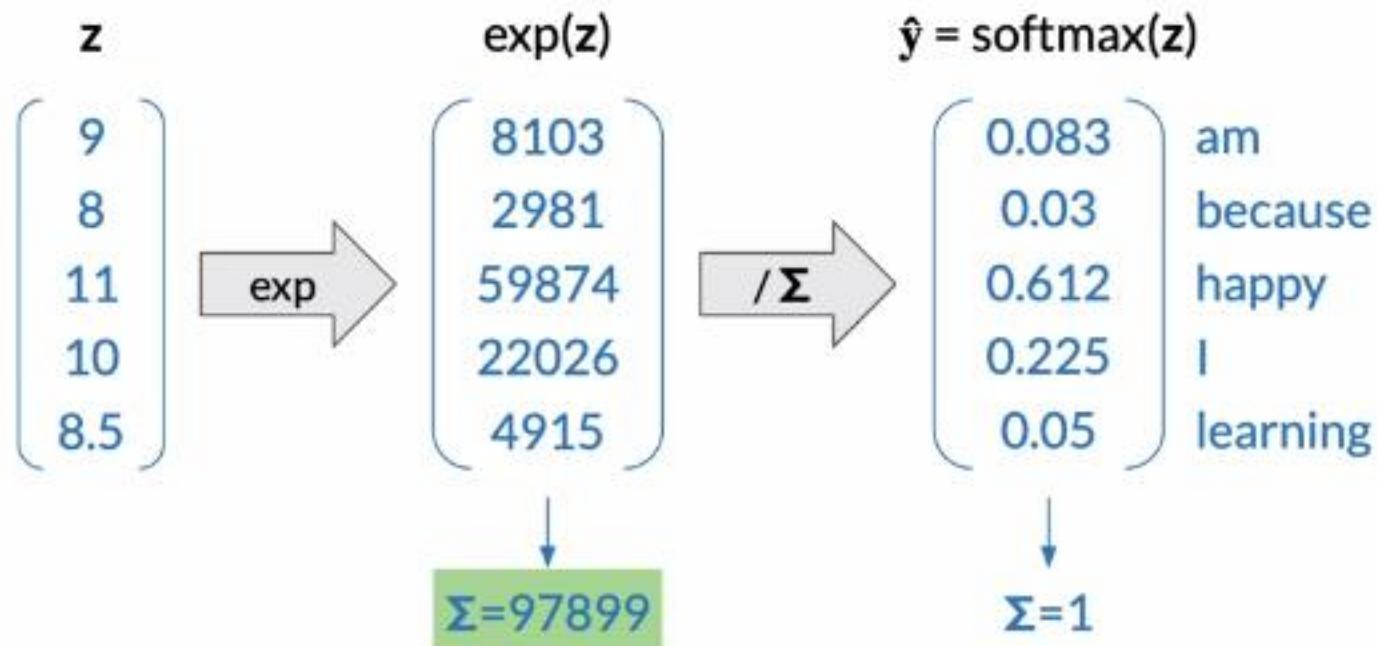
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



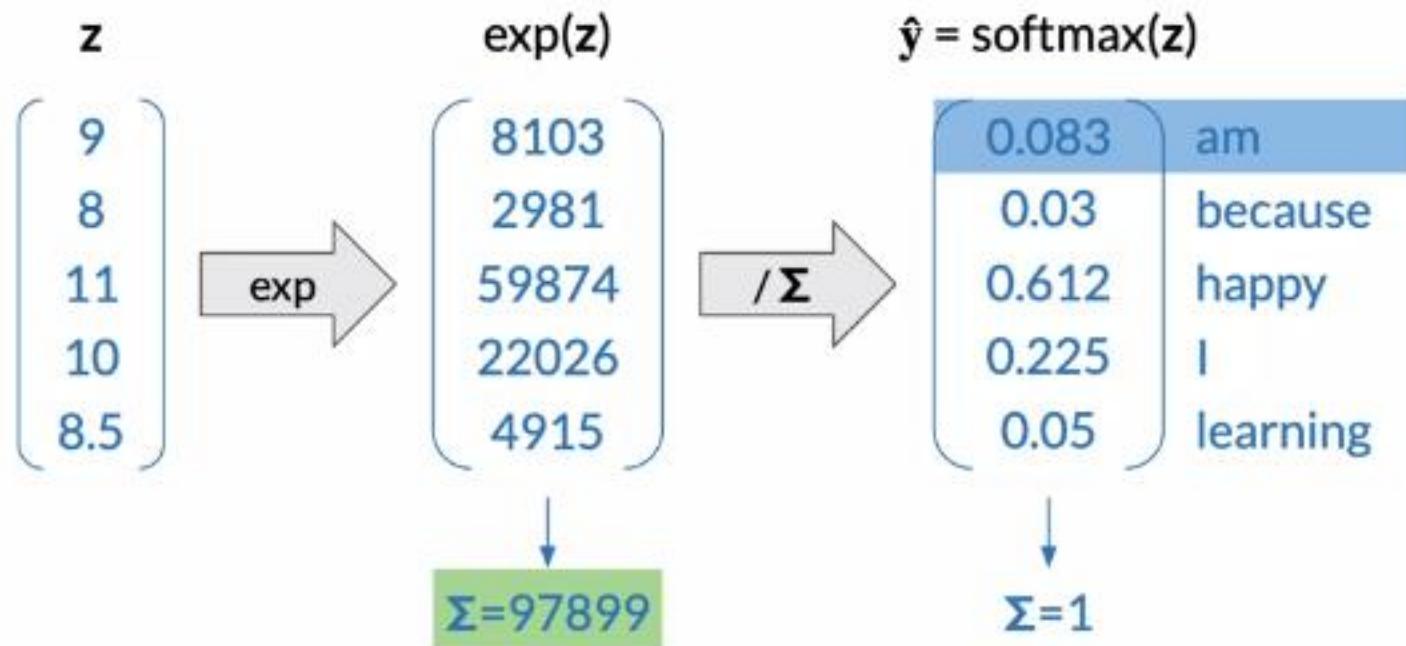
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



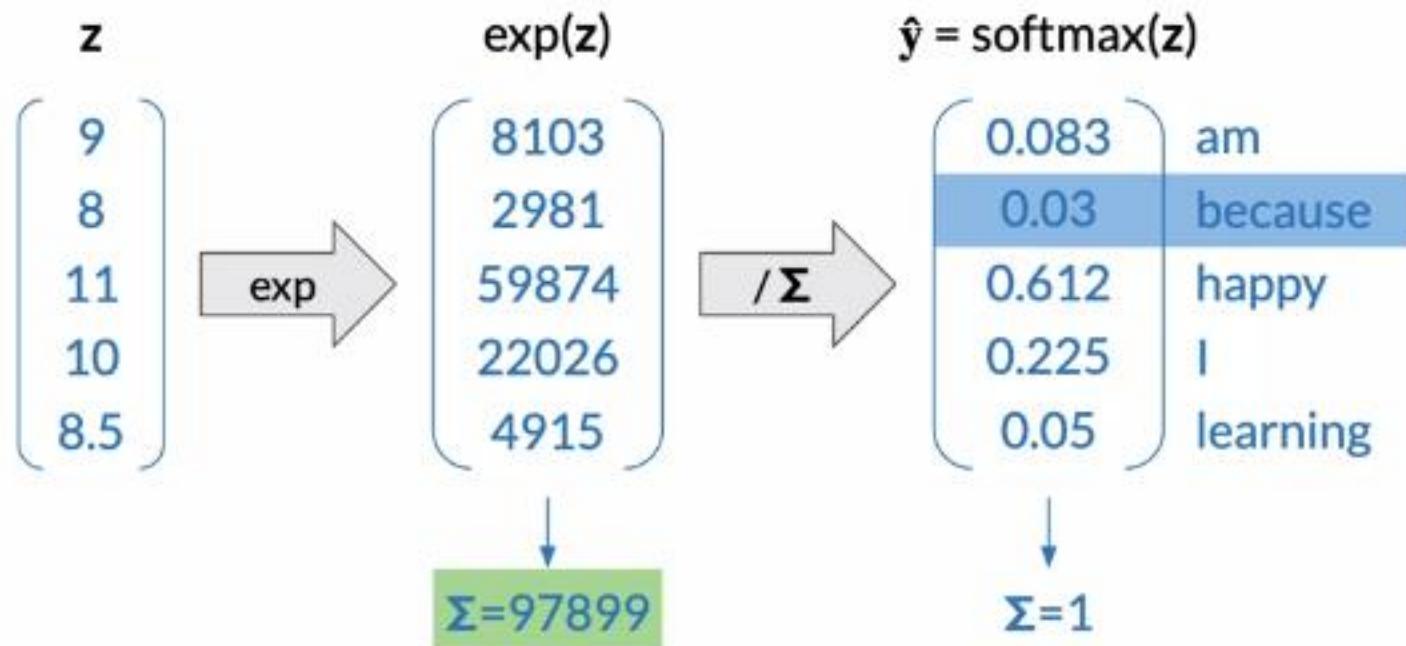
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



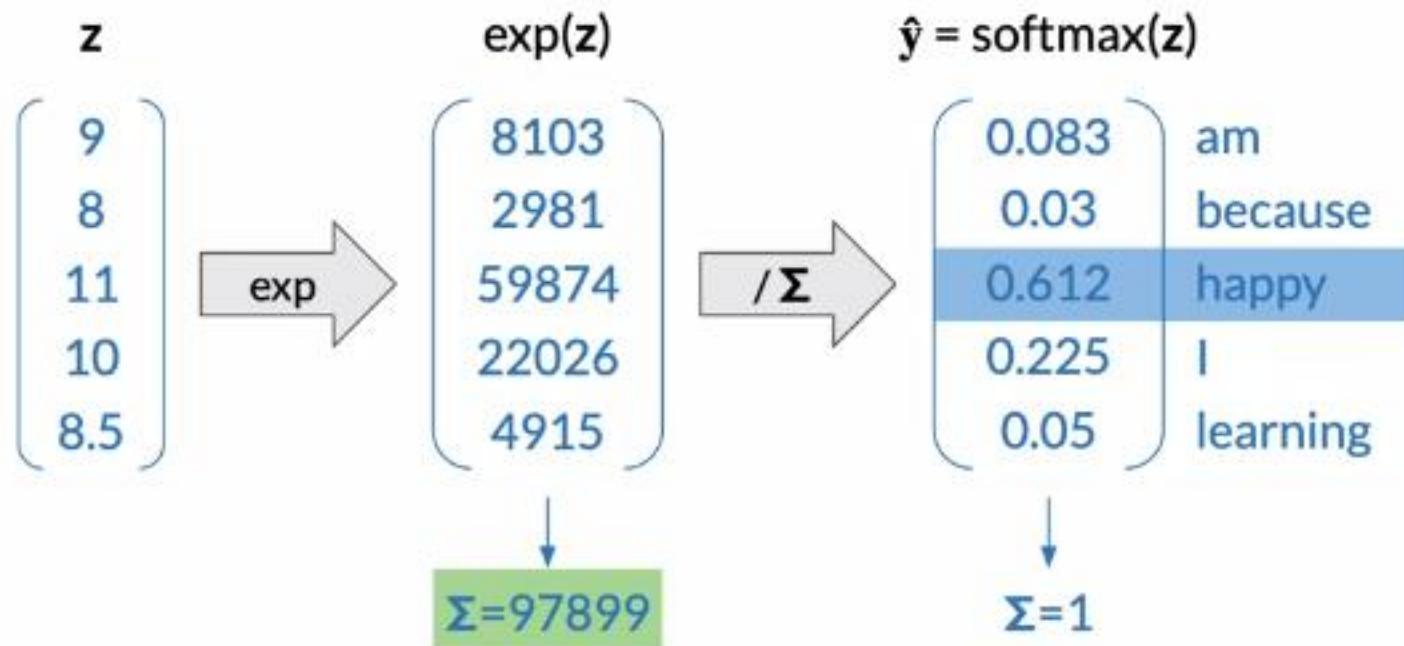
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



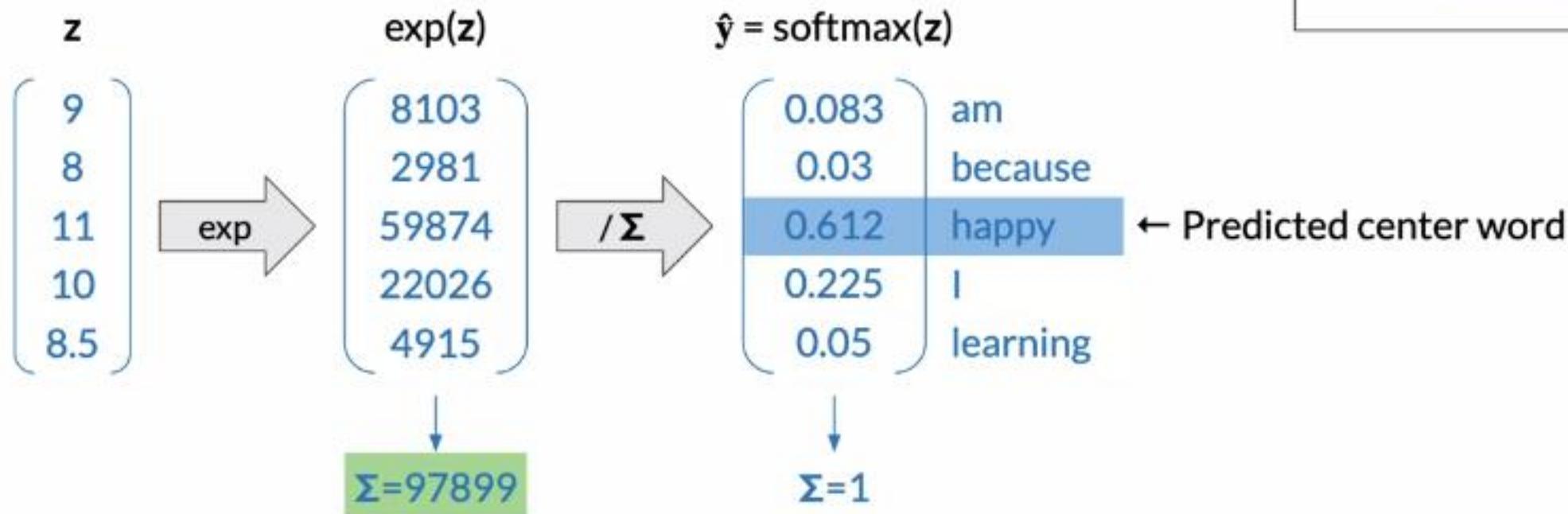
Softmax: example

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$

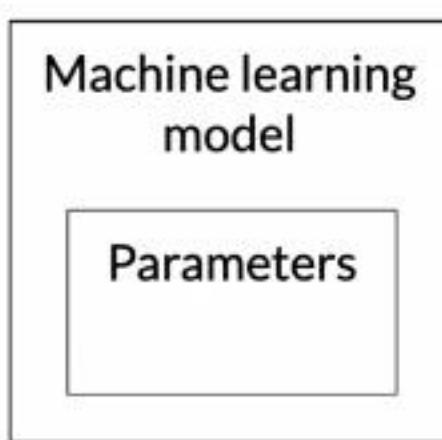


Softmax: example

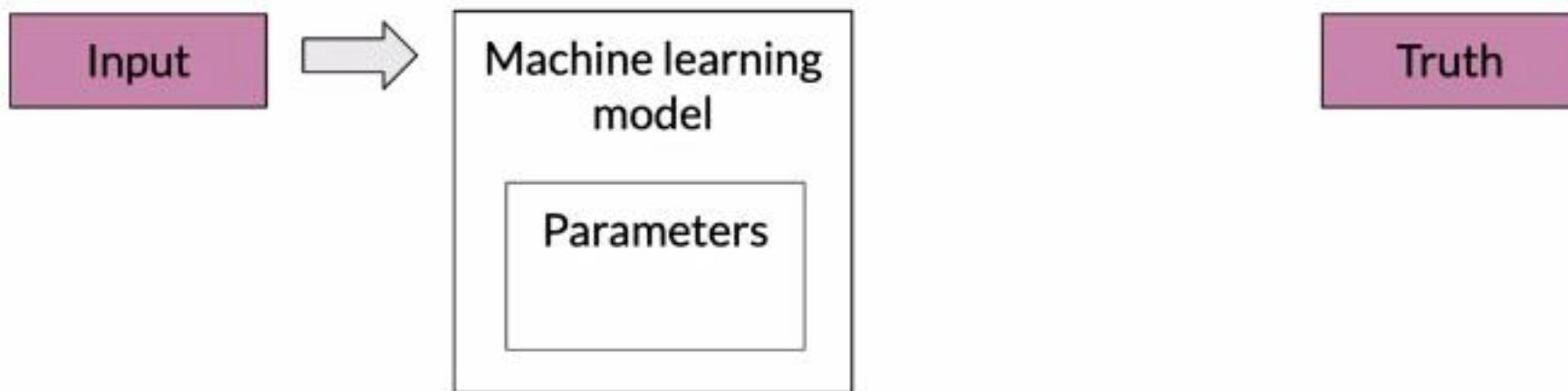
$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$



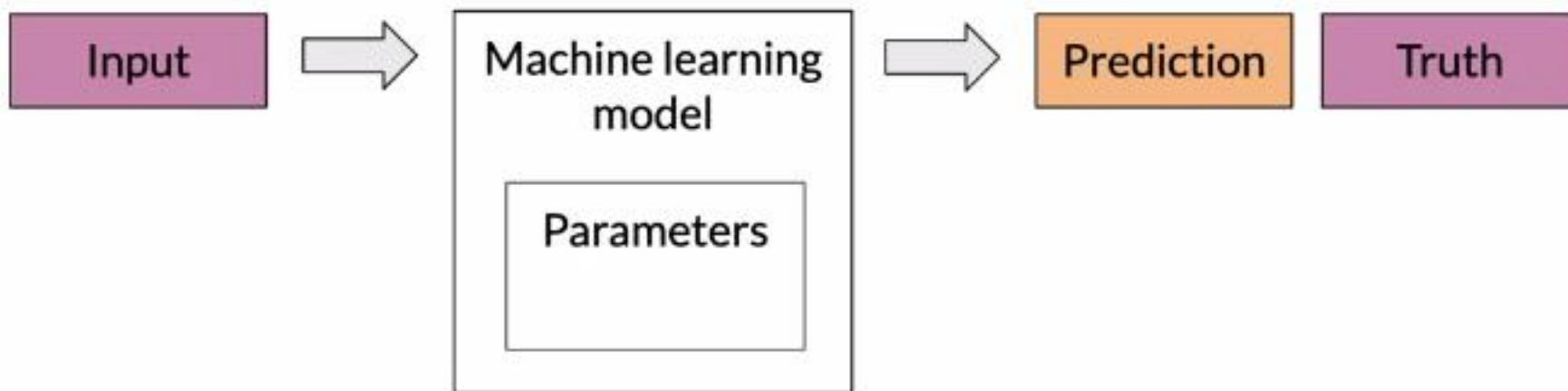
Loss



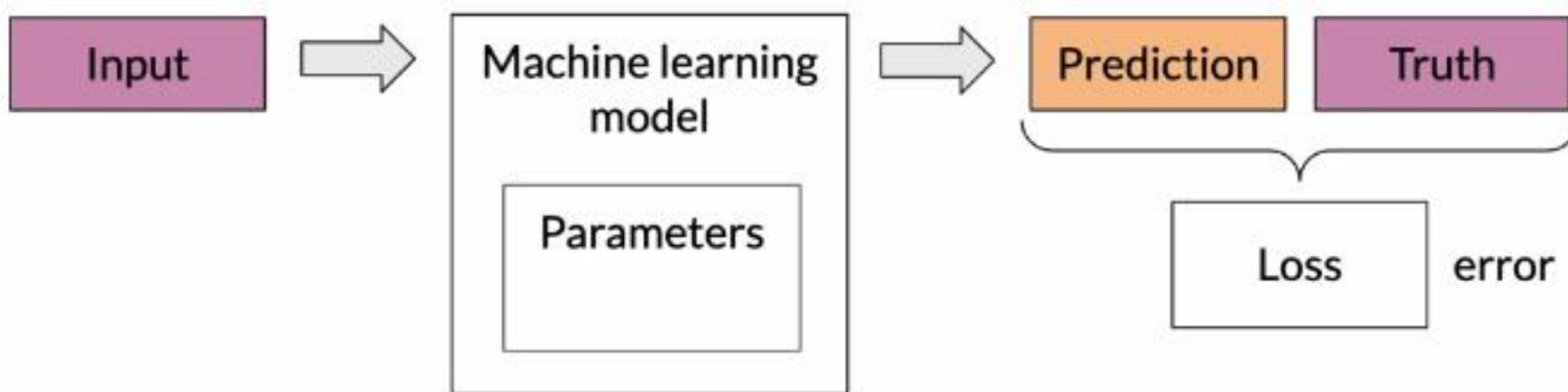
Loss



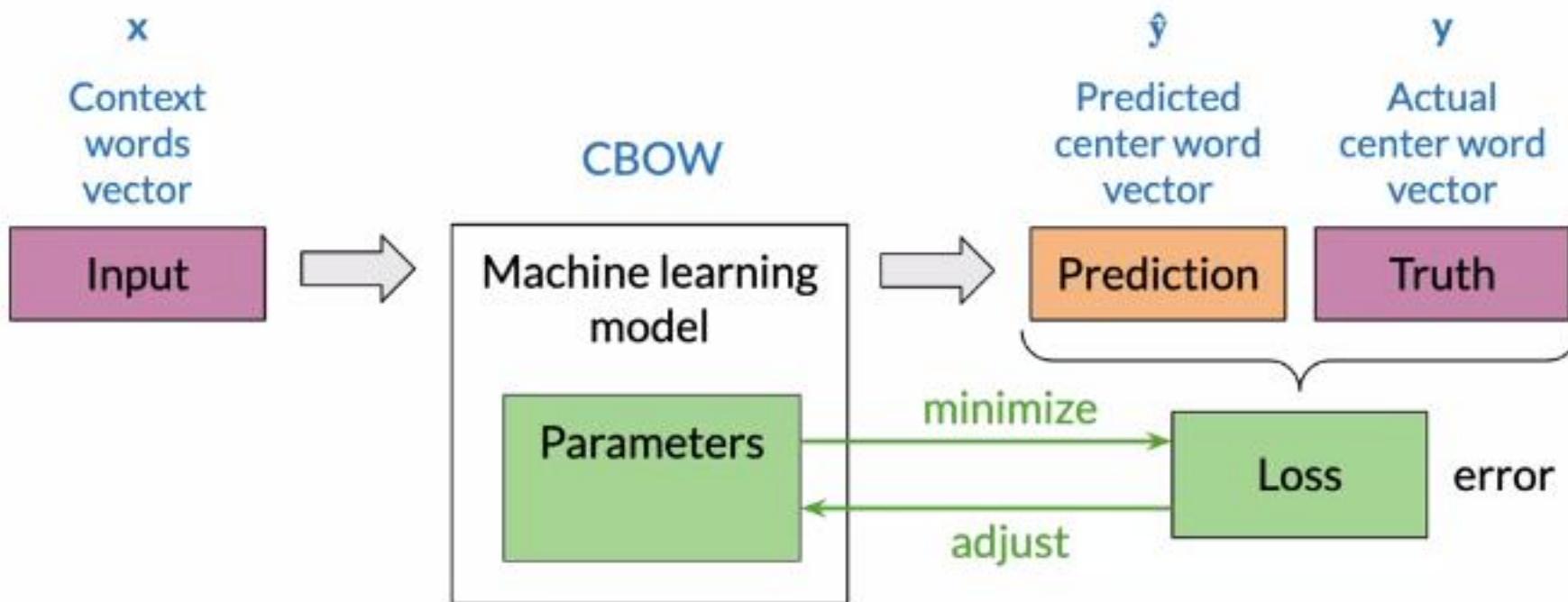
Loss



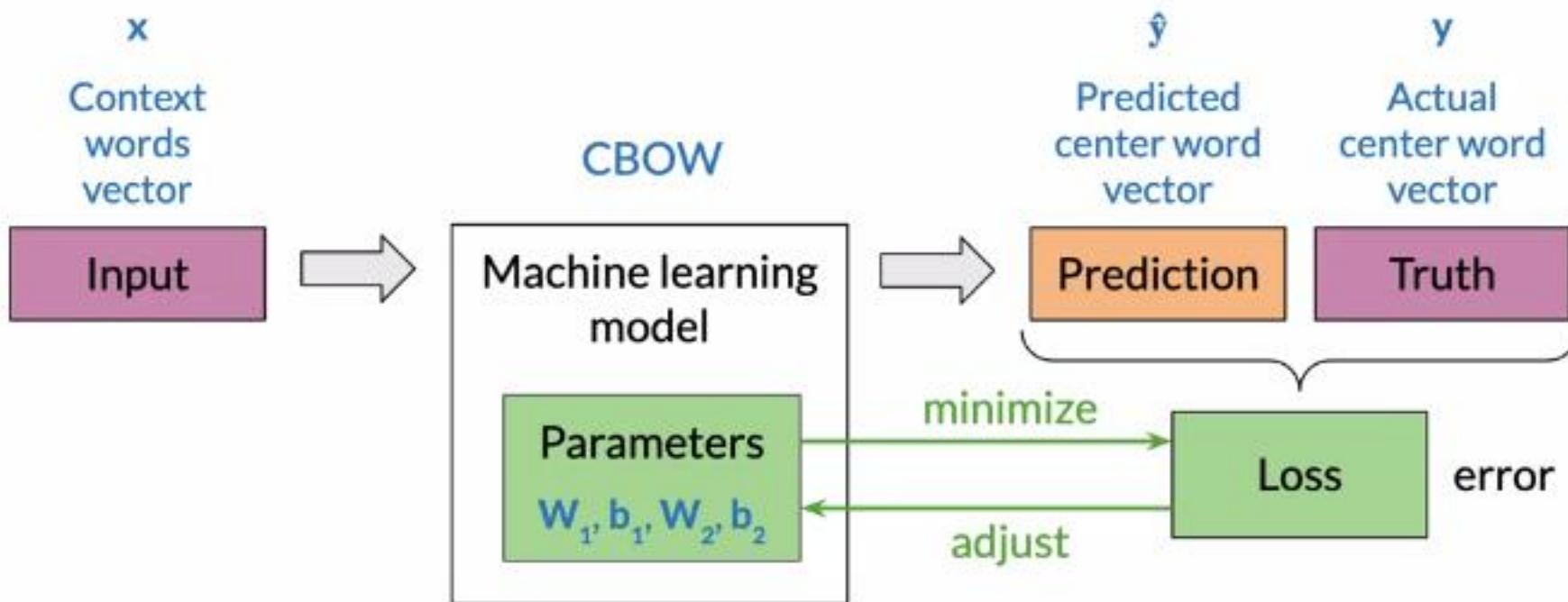
Loss



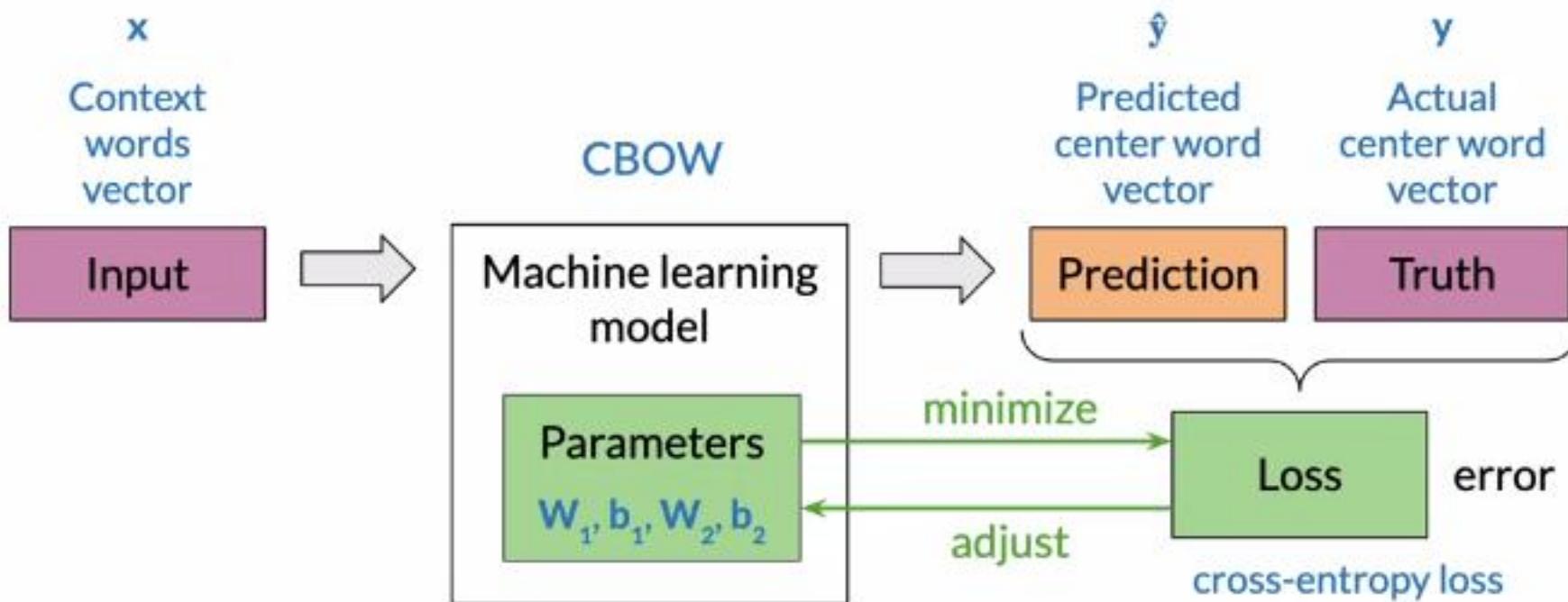
Loss



Loss



Loss



Cross-entropy loss

Actual	Predicted
$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_v \end{pmatrix}$	$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_v \end{pmatrix}$

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Actual	Predicted
$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_V \end{bmatrix}$	$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_V \end{bmatrix}$

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Actual	Predicted
$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_V \end{pmatrix}$	$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_V \end{pmatrix}$

I am happy because I am learning

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

y

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ am
because
happy
I
learning}$$

Actual	Predicted
$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_V \end{bmatrix}$	$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_V \end{bmatrix}$

I am happy because I am learning

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Actual	Predicted
$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_v \end{pmatrix}$	$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_v \end{pmatrix}$

I am happy because I am learning

\mathbf{y}	$\hat{\mathbf{y}}$	
0	am	0.083
0	because	0.03
1	happy	0.611
0	I	0.225
0	learning	0.05

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

y		\hat{y}
0	am	0.083
0	because	0.03
1	happy	0.611
0	I	0.225
0	learning	0.05

Actual $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_v \end{pmatrix}$	Predicted $\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_v \end{pmatrix}$
---	--

I am happy because I am learning

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Actual $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_v \end{pmatrix}$	Predicted $\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_v \end{pmatrix}$
---	--

I am happy because I am learning

	\mathbf{y}	$\hat{\mathbf{y}}$	$\log(\hat{\mathbf{y}})$
0	am	0.083	-2.49
0	because	0.03	-3.49
1	happy	0.611	-0.49
0	I	0.225	-1.49
0	learning	0.05	-2.49

log

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Actual $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_v \end{pmatrix}$	Predicted $\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_v \end{pmatrix}$
---	--

I am happy because I am learning

\mathbf{y}	$\hat{\mathbf{y}}$	$\log(\hat{\mathbf{y}})$	$\mathbf{y} \odot \log(\hat{\mathbf{y}})$
0	am	0.083	0
0	because	0.03	0
1	happy	0.611	-0.49
0	I	0.225	-1.49
0	learning	0.05	-2.49

Diagram showing the calculation of cross-entropy loss:

Input \mathbf{y} and $\hat{\mathbf{y}}$ are processed through a \log operation to produce $\log(\hat{\mathbf{y}})$. This is then element-wise multiplied (\odot) by \mathbf{y} to produce the final result.

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Actual $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_v \end{pmatrix}$	Predicted $\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_v \end{pmatrix}$
---	--

I am happy because I am learning

\mathbf{y}	$\hat{\mathbf{y}}$	$\log(\hat{\mathbf{y}})$	$\mathbf{y} \odot \log(\hat{\mathbf{y}})$
0	am	0.083	0
0	because	0.03	0
1	happy	0.611	-0.49
0	I	0.225	0
0	learning	0.05	0

Diagram showing the calculation of cross-entropy loss:

Initial vectors \mathbf{y} and $\hat{\mathbf{y}}$ are transformed by the \log function to produce $\log(\hat{\mathbf{y}})$. This is then element-wise multiplied (\odot) by \mathbf{y} to produce the final result.

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Actual $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_v \end{pmatrix}$	Predicted $\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_v \end{pmatrix}$
---	--

I am happy because I am learning

\mathbf{y}	$\hat{\mathbf{y}}$	$\log(\hat{\mathbf{y}})$	$\mathbf{y} \odot \log(\hat{\mathbf{y}})$	$J = -\Sigma$	
$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{bmatrix}$	$\begin{bmatrix} 0.083 \\ 0.03 \\ 0.611 \\ 0.225 \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} -2.49 \\ -3.49 \\ -0.49 \\ -1.49 \\ -2.49 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ -0.49 \\ 0 \\ 0 \end{bmatrix}$	$J = 0.49$

Diagram illustrating the calculation of cross-entropy loss:

- Inputs:
 - Actual labels (\mathbf{y}): $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$
 - Predicted probabilities ($\hat{\mathbf{y}}$): $\begin{bmatrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{bmatrix}$ (values: 0.083, 0.03, 0.611, 0.225, 0.05)
- Operations:
 - Logarithm (\log): $\begin{bmatrix} -2.49 \\ -3.49 \\ -0.49 \\ -1.49 \\ -2.49 \end{bmatrix}$
 - Element-wise multiplication ($\odot \mathbf{y}$): $\begin{bmatrix} 0 \\ 0 \\ -0.49 \\ 0 \\ 0 \end{bmatrix}$
 - Sum ($-\Sigma$): $J = 0.49$

Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

y
0
0
1
0
0

am
because
happy
I
learning

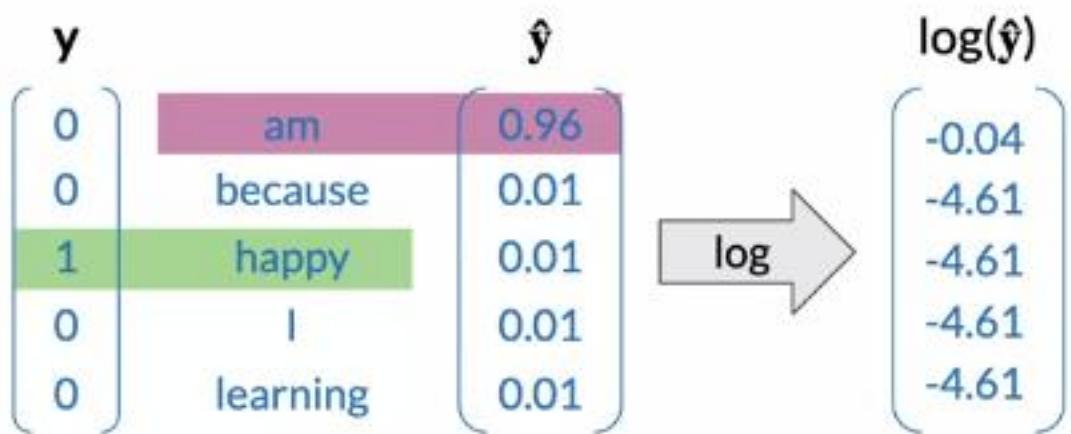
Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

y		\hat{y}
0	am	0.96
0	because	0.01
1	happy	0.01
0	I	0.01
0	learning	0.01

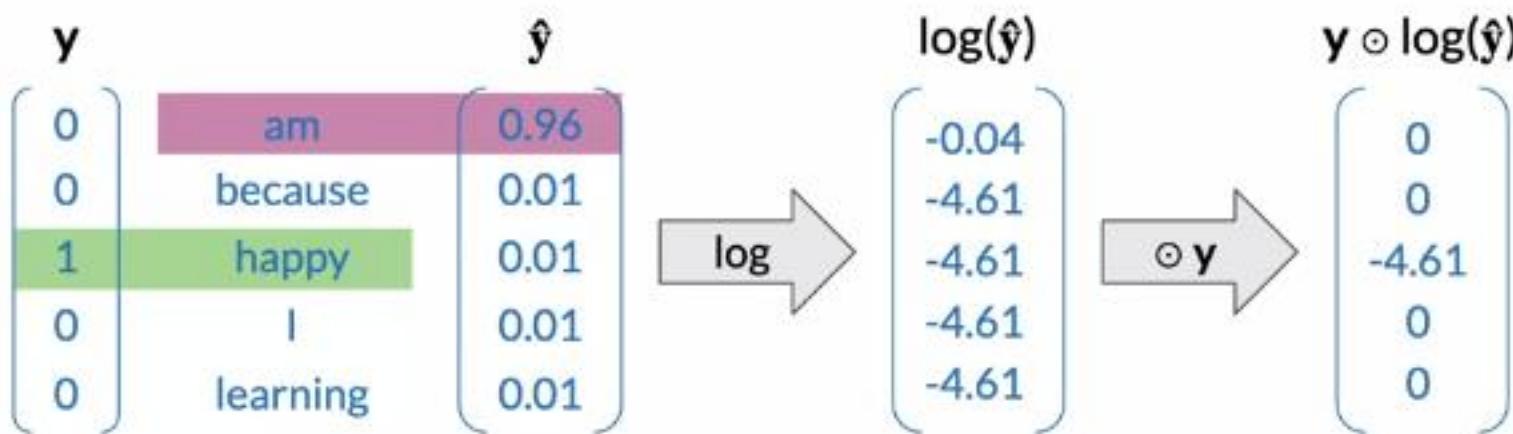
Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$



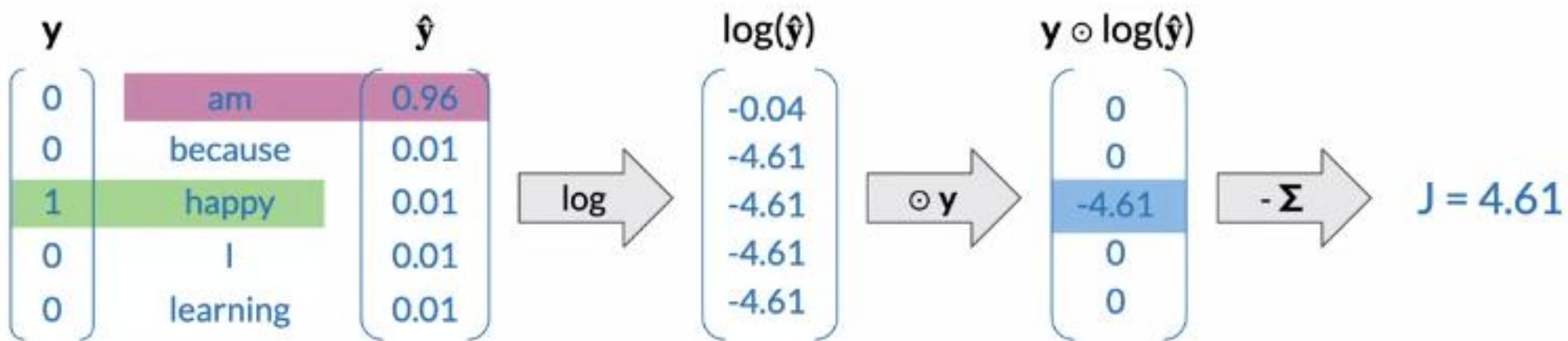
Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$



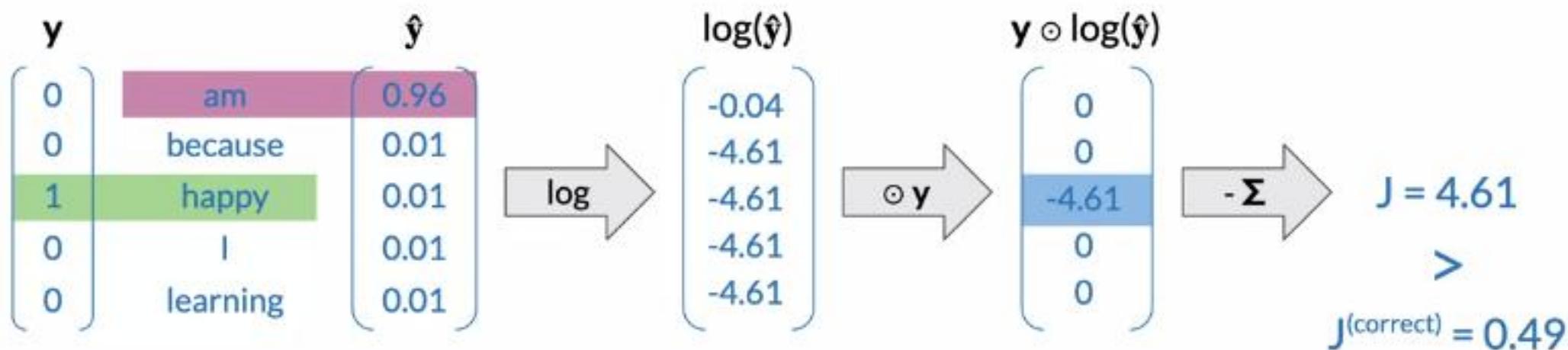
Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$



Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$



Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

$$J = -\log \hat{y}_{\text{actual word}}$$

y		\hat{y}
0	am	0.96
0	because	0.01
1	happy	0.01
0	I	0.01
0	learning	0.01

$$\rightarrow J = 4.61$$

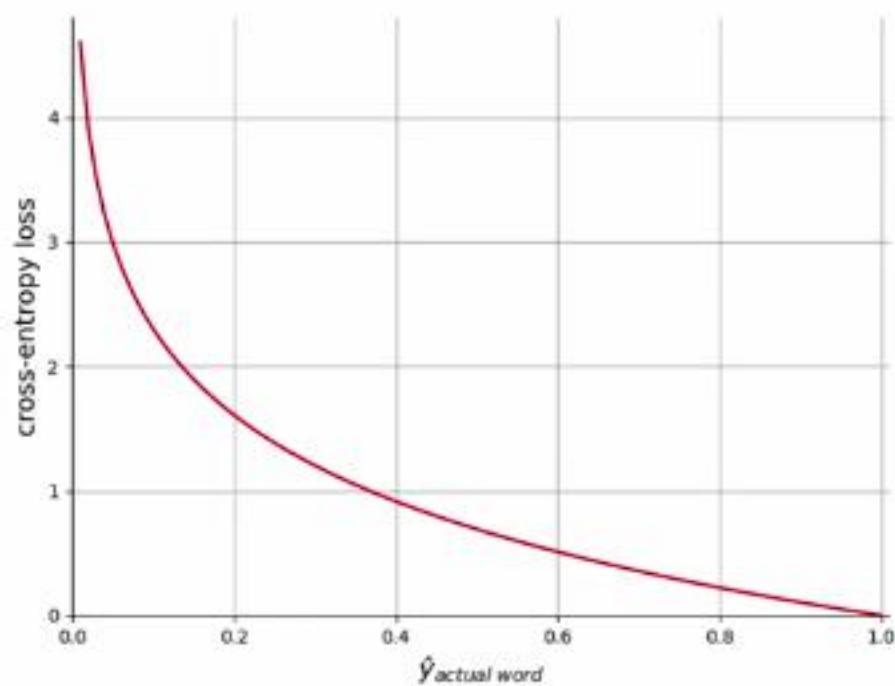
Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

$$J = -\log \hat{y}_{\text{actual word}}$$

y		\hat{y}
0	am	0.96
0	because	0.01
1	happy	0.01
0	I	0.01
0	learning	0.01

$$\rightarrow J = 4.61$$



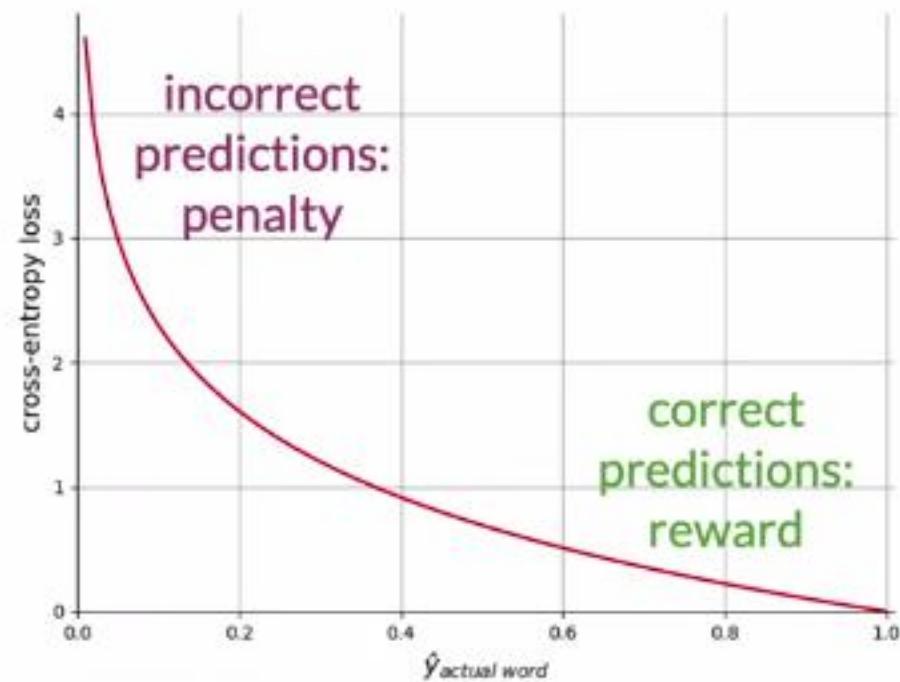
Cross-entropy loss

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

$$J = -\log \hat{y}_{\text{actual word}}$$

y		\hat{y}
0	am	0.96
0	because	0.01
1	happy	0.01
0	I	0.01
0	learning	0.01

$\rightarrow J = 4.61$





deeplearning.ai

Training a CBOW Model

Forward Propagation

Training process

- Forward propagation

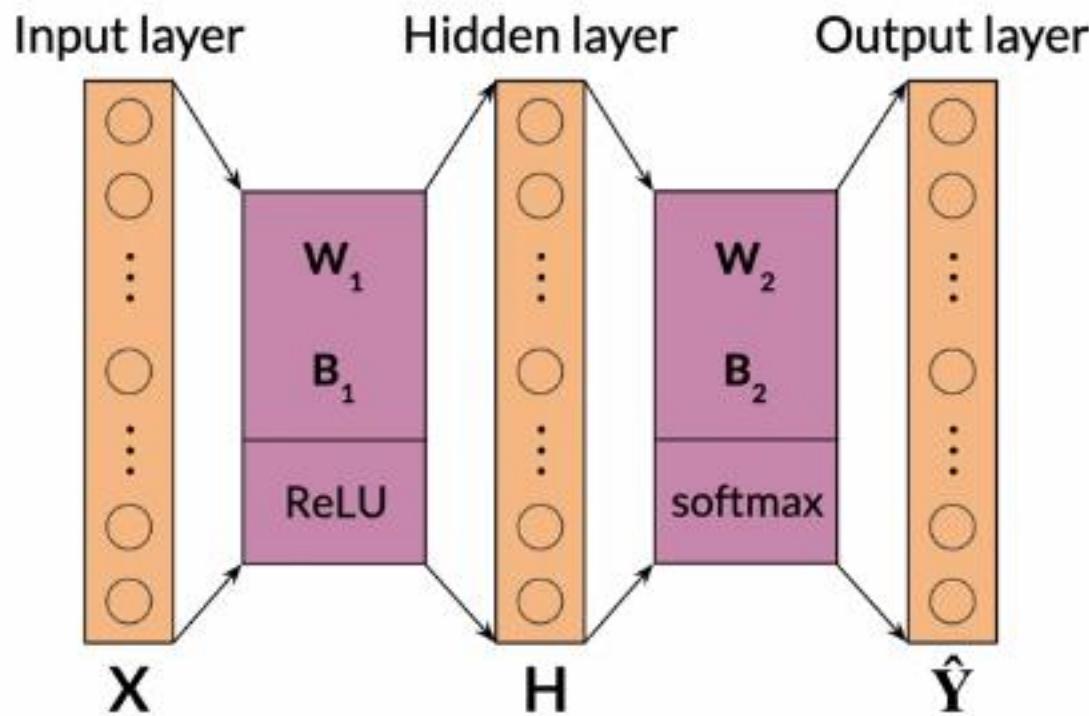
Training process

- Forward propagation
- Cost

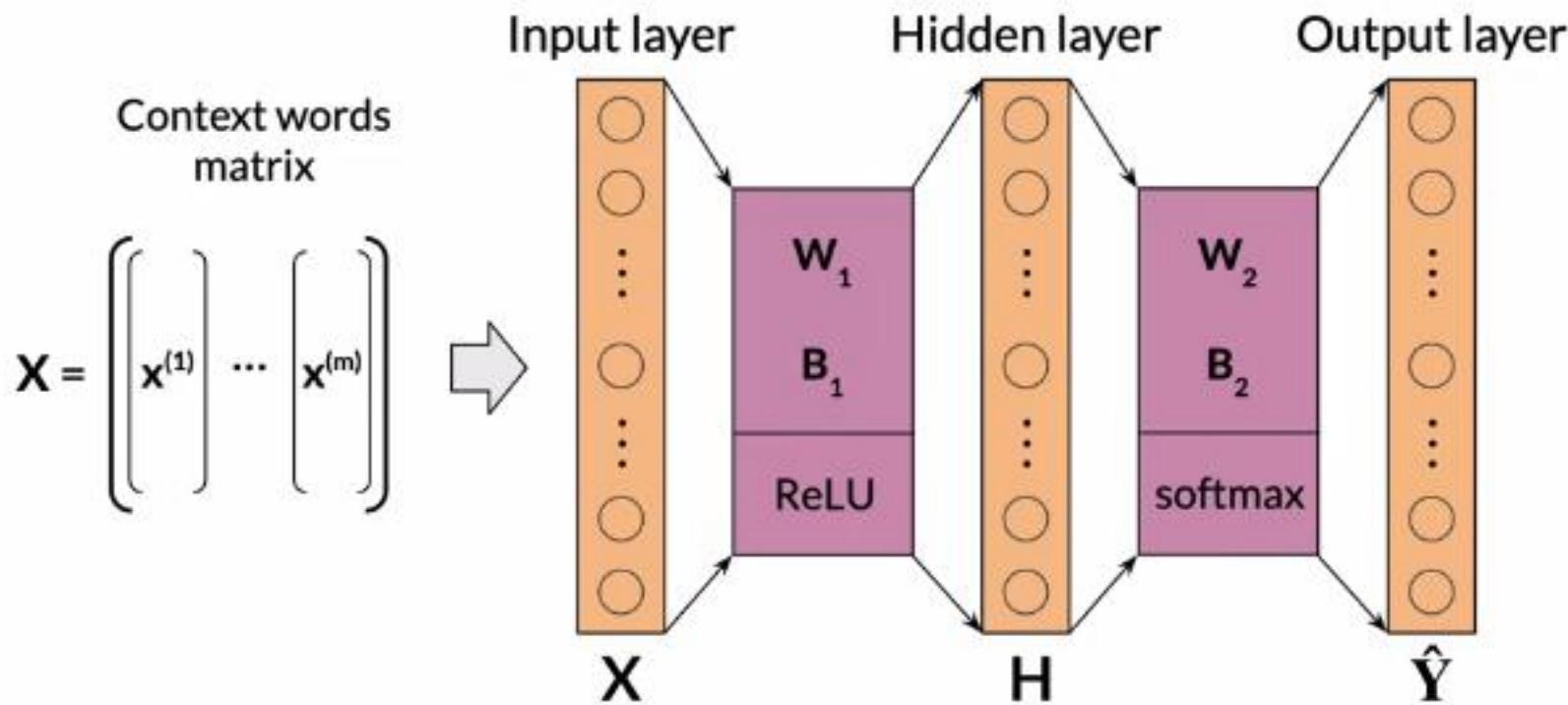
Training process

- Forward propagation
- Cost
- Backpropagation and gradient descent

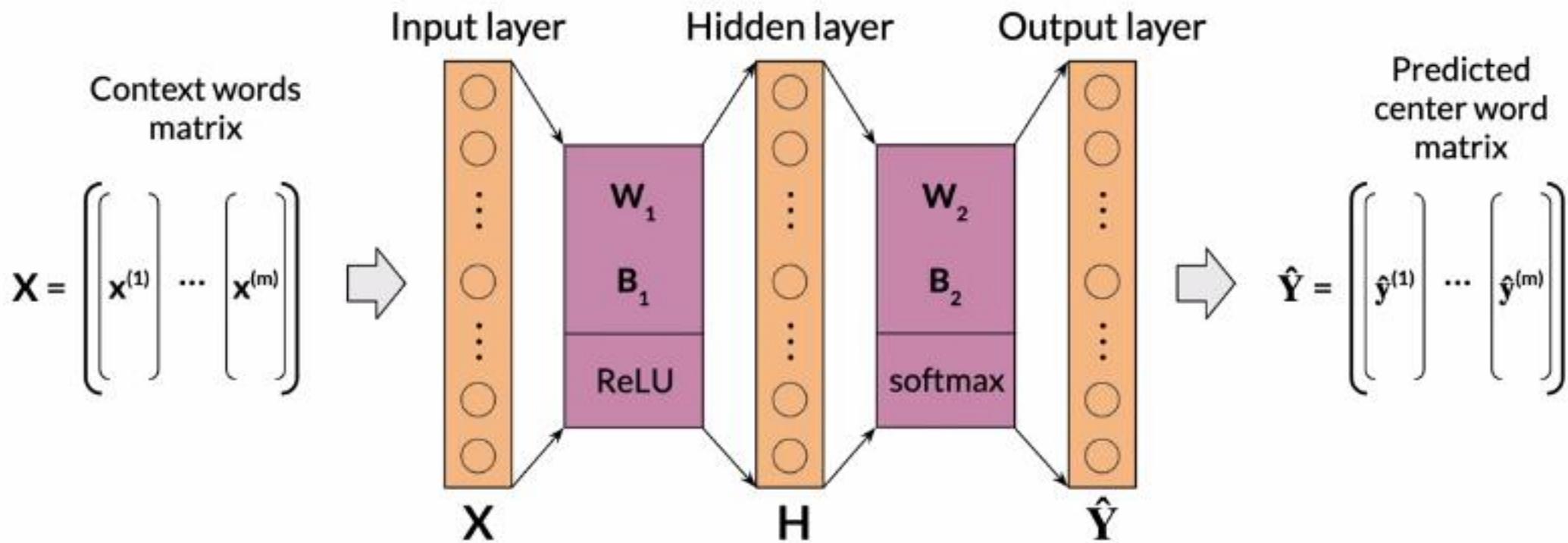
Forward propagation



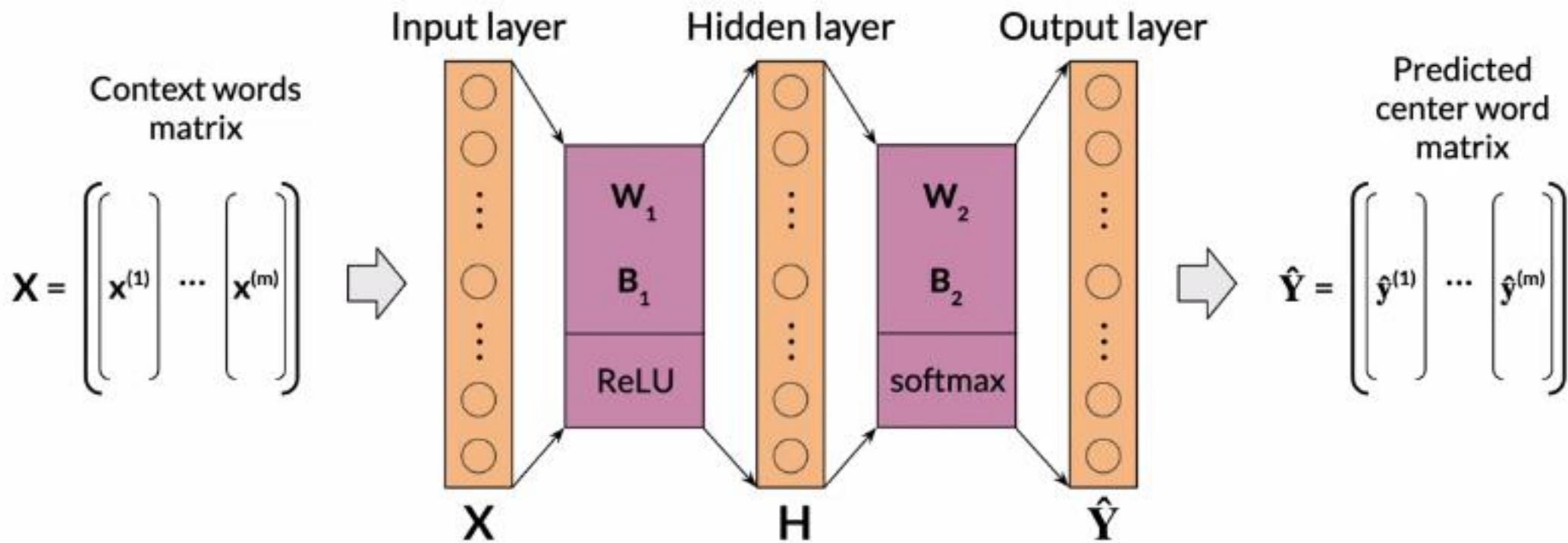
Forward propagation



Forward propagation



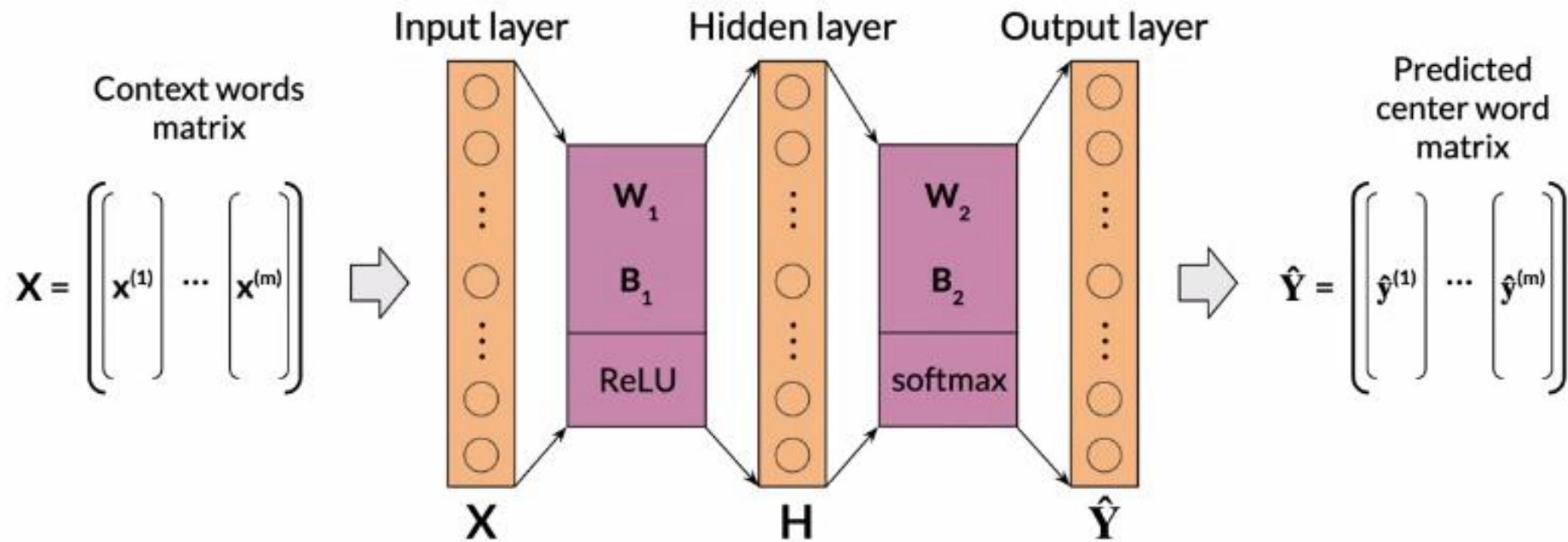
Forward propagation



Forward propagation

$$Z_1 = \mathbf{W}_1 \mathbf{X} + \mathbf{B}_1$$
$$\mathbf{H} = \text{ReLU}(Z_1)$$

$$Z_2 = \mathbf{W}_2 \mathbf{H} + \mathbf{B}_2$$
$$\hat{\mathbf{Y}} = \text{softmax}(Z_2)$$



Cost

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Cost

Cost: mean of losses

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Cost

$$J = - \sum_{k=1}^V y_k \log \hat{y}_k$$

Cost: mean of losses

$$J_{batch} = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^V y_j^{(i)} \log \hat{y}_j^{(i)}$$

Predicted
center word
matrix

$$\hat{\mathbf{Y}} = \begin{pmatrix} \hat{\mathbf{y}}^{(1)} & \dots & \hat{\mathbf{y}}^{(m)} \end{pmatrix}$$

Actual center
word matrix

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}^{(1)} & \dots & \mathbf{y}^{(m)} \end{pmatrix}$$

Cost

Cost: mean of losses

$$J_{batch} = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^V y_j^{(i)} \log \hat{y}_j^{(i)}$$

$$J_{batch} = -\frac{1}{m} \sum_{i=1}^m J^{(i)}$$

Predicted
center word
matrix

$$\hat{\mathbf{Y}} = \begin{pmatrix} \hat{\mathbf{y}}^{(1)} & \cdots & \hat{\mathbf{y}}^{(m)} \end{pmatrix}$$

Actual center
word matrix

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}^{(1)} & \cdots & \mathbf{y}^{(m)} \end{pmatrix}$$

$$J = -\sum_{k=1}^V y_k \log \hat{y}_k$$

Minimizing the cost

Minimizing the cost

- Backpropagation: calculate partial derivatives of cost with respect to weights and biases

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1}, \frac{\partial J_{batch}}{\partial \mathbf{W}_2}, \frac{\partial J_{batch}}{\partial \mathbf{b}_1}, \frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

Minimizing the cost

$$J_{batch} = f(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2)$$

- Backpropagation: calculate partial derivatives of cost with respect to weights and biases

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1}, \frac{\partial J_{batch}}{\partial \mathbf{W}_2}, \frac{\partial J_{batch}}{\partial \mathbf{b}_1}, \frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

Minimizing the cost

- Backpropagation: calculate partial derivatives of cost with respect to weights and biases

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1}, \frac{\partial J_{batch}}{\partial \mathbf{W}_2}, \frac{\partial J_{batch}}{\partial \mathbf{b}_1}, \frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

- Gradient descent: update weights and biases

Backpropagation

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_2}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_1}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_2}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_1}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_2} = \frac{1}{m} (\hat{\mathbf{Y}} - \mathbf{Y}) \mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_1}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_2} = \frac{1}{m} (\hat{\mathbf{Y}} - \mathbf{Y}) \mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{1}_m^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_2} = \frac{1}{m} (\hat{\mathbf{Y}} - \mathbf{Y}) \mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{1}_m^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

$$\mathbf{1}_m = [1, \dots, 1]$$

$\xleftarrow[m]{}$

$$\mathbf{A} \cdot \mathbf{1}_m^\top = \left[\begin{array}{c|c} \text{[]} & \mathbf{1} \\ \hline \end{array} \right] = \left[\begin{array}{c} \Sigma \\ \vdots \\ \Sigma \end{array} \right]$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_2} = \frac{1}{m} (\hat{\mathbf{Y}} - \mathbf{Y}) \mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{1}_m^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

$$\mathbf{1}_m = [1, \dots, 1]$$

\longleftrightarrow
 m

$$\mathbf{A} \cdot \mathbf{1}_m^\top = \left[\begin{array}{c} \boxed{} \\ \vdots \end{array} \right] \left[\begin{array}{c} 1 \\ \vdots \\ 1 \end{array} \right] = \boxed{\Sigma}$$

```
import numpy as np  
# code to initialize matrix a omitted  
np.sum(a, axis=1, keepdims=True)
```

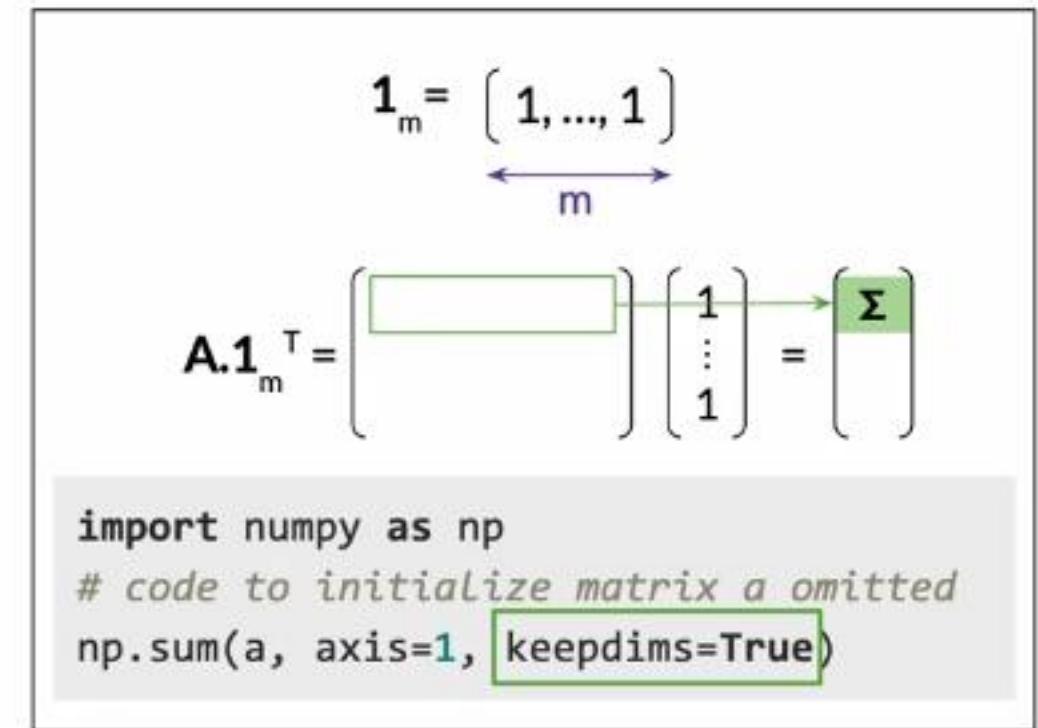
Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W}_2} = \frac{1}{m} (\hat{\mathbf{Y}} - \mathbf{Y}) \mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_1} = \frac{1}{m} \text{ReLU} \left(\mathbf{W}_2^\top (\hat{\mathbf{Y}} - \mathbf{Y}) \right) \mathbf{1}_m^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b}_2} = \frac{1}{m} (\hat{\mathbf{Y}} - \mathbf{Y}) \mathbf{1}_m^\top$$



Gradient descent

Hyperparameter: learning rate α

$$\mathbf{W}_1 := \mathbf{W}_1 - \alpha \frac{\partial J_{batch}}{\partial \mathbf{W}_1}$$

$$\mathbf{W}_2 := \mathbf{W}_2 - \alpha \frac{\partial J_{batch}}{\partial \mathbf{W}_2}$$

$$\mathbf{b}_1 := \mathbf{b}_1 - \alpha \frac{\partial J_{batch}}{\partial \mathbf{b}_1}$$

$$\mathbf{b}_2 := \mathbf{b}_2 - \alpha \frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$

Gradient descent

Hyperparameter: learning rate α

$$\mathbf{W}_1 := \mathbf{W}_1 - \alpha \frac{\partial J_{batch}}{\partial \mathbf{W}_1}$$

$$\mathbf{W}_2 := \mathbf{W}_2 - \alpha \frac{\partial J_{batch}}{\partial \mathbf{W}_2}$$

$$\mathbf{b}_1 := \mathbf{b}_1 - \alpha \frac{\partial J_{batch}}{\partial \mathbf{b}_1}$$

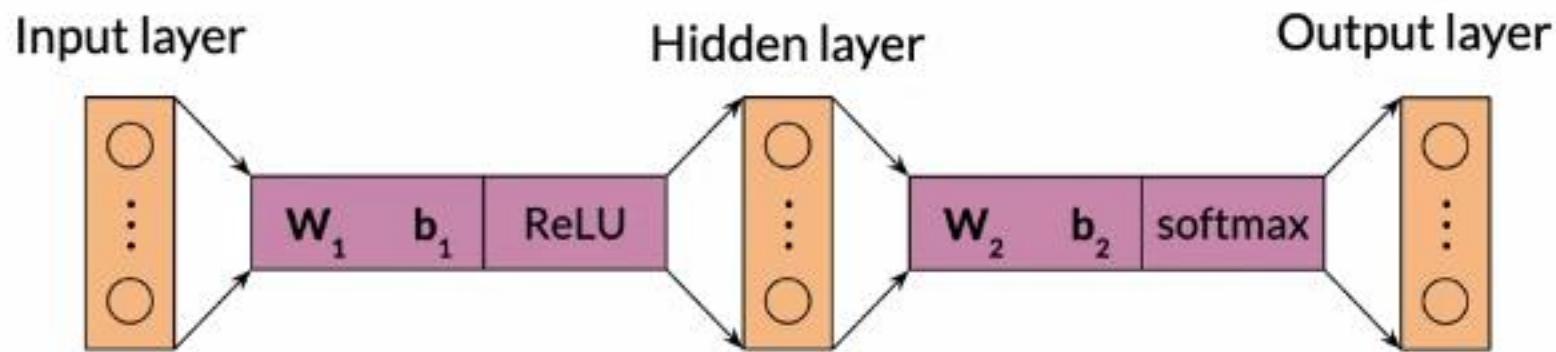
$$\mathbf{b}_2 := \mathbf{b}_2 - \alpha \frac{\partial J_{batch}}{\partial \mathbf{b}_2}$$



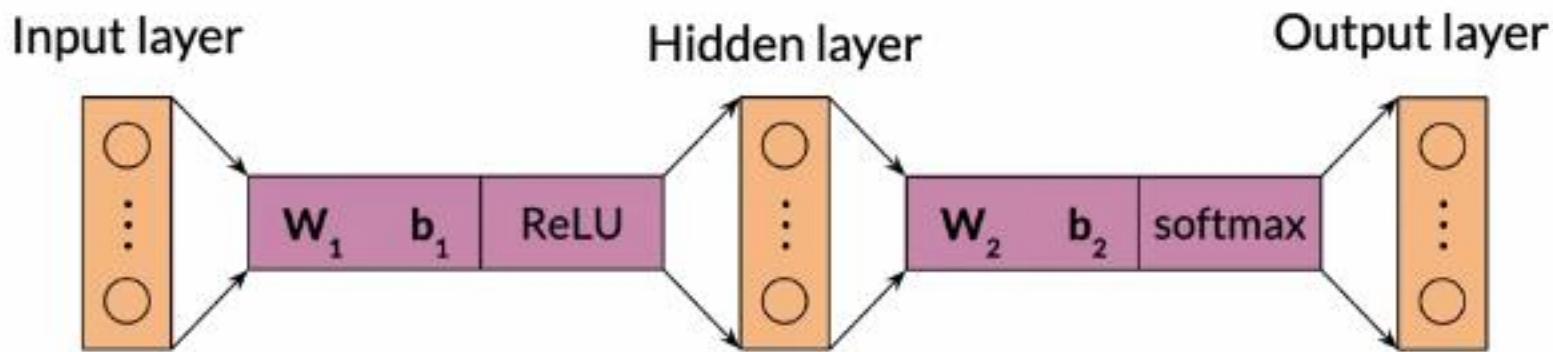
deeplearning.ai

Extracting Word Embedding Vectors

Extracting word embedding vectors: option 1

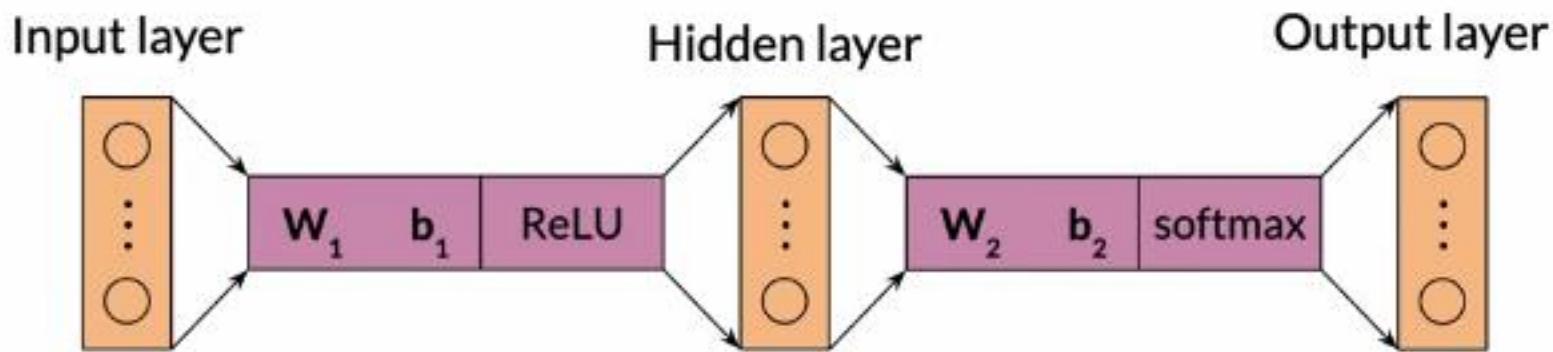


Extracting word embedding vectors: option 1



$$\mathbf{W}_1 = \left[\begin{matrix} \mathbf{w}^{(1)} & \dots & \mathbf{w}^{(N)} \end{matrix} \right]$$

Extracting word embedding vectors: option 1



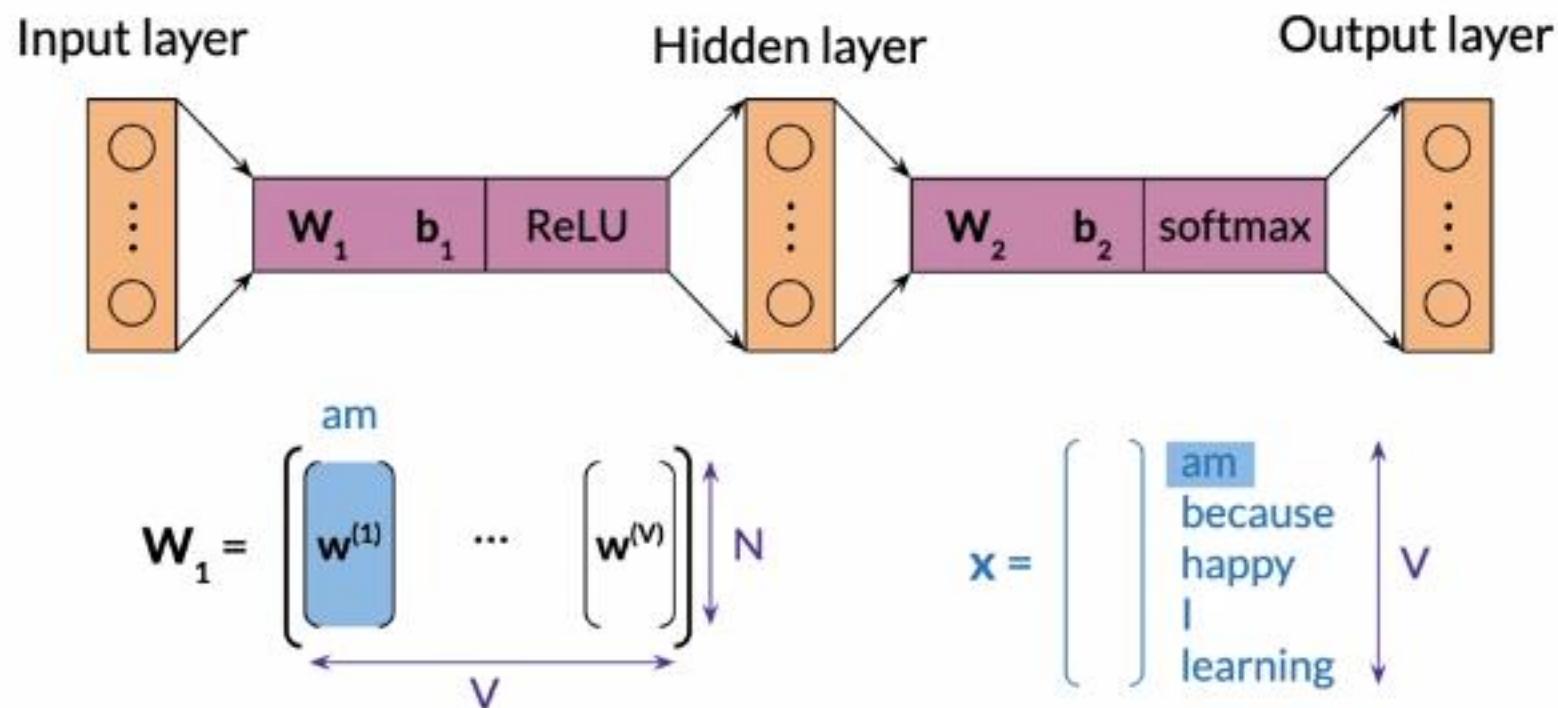
$$\mathbf{W}_1 = \begin{pmatrix} \mathbf{w}^{(1)} & \cdots & \mathbf{w}^{(N)} \end{pmatrix}$$

$\xleftarrow[V]{} \quad \xrightarrow[N]{} \quad$

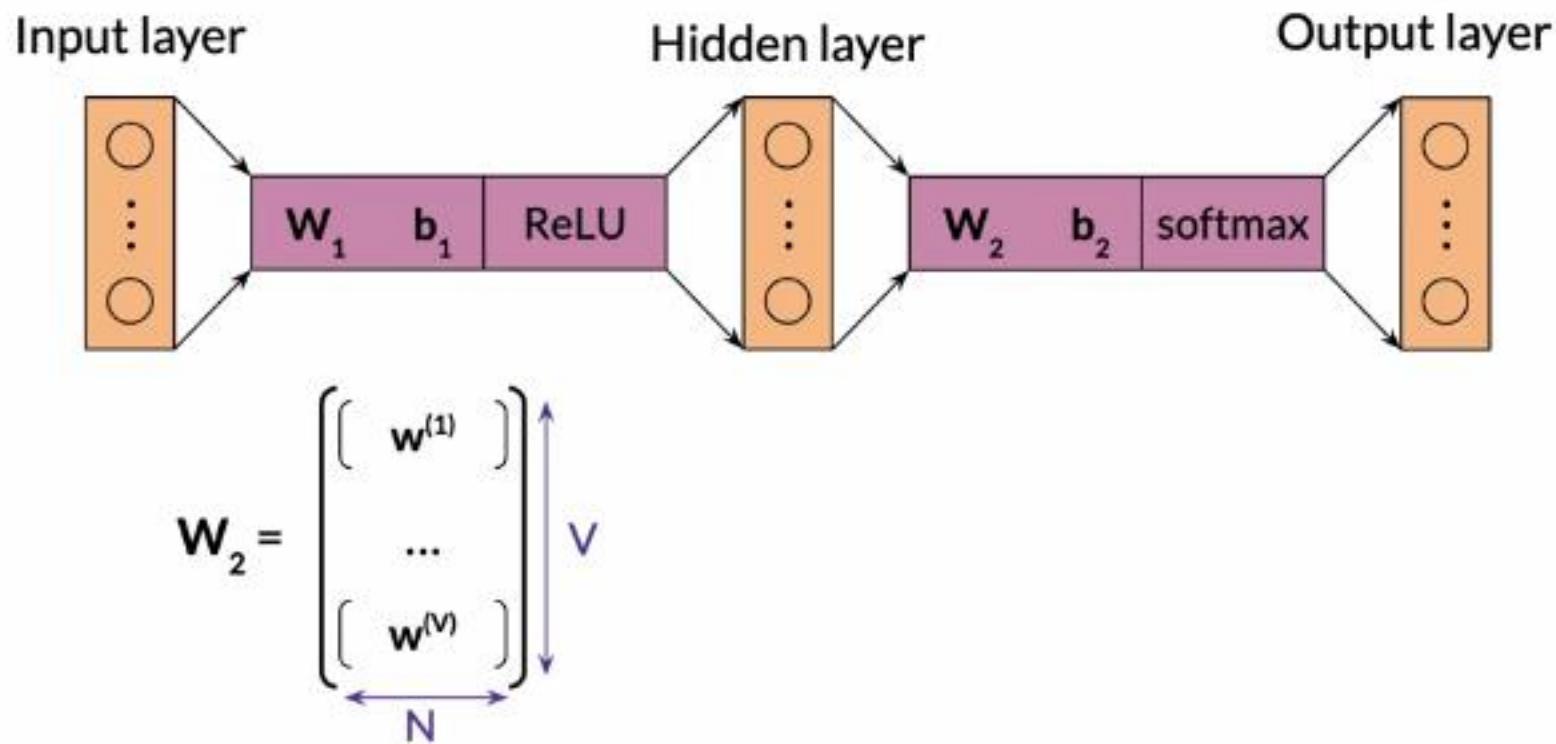
$$\mathbf{x} = \begin{pmatrix} \text{am} \\ \text{because} \\ \text{happy} \\ \vdots \\ \text{learning} \end{pmatrix}$$

$\xleftarrow[V]{} \quad \xrightarrow[V]{} \quad$

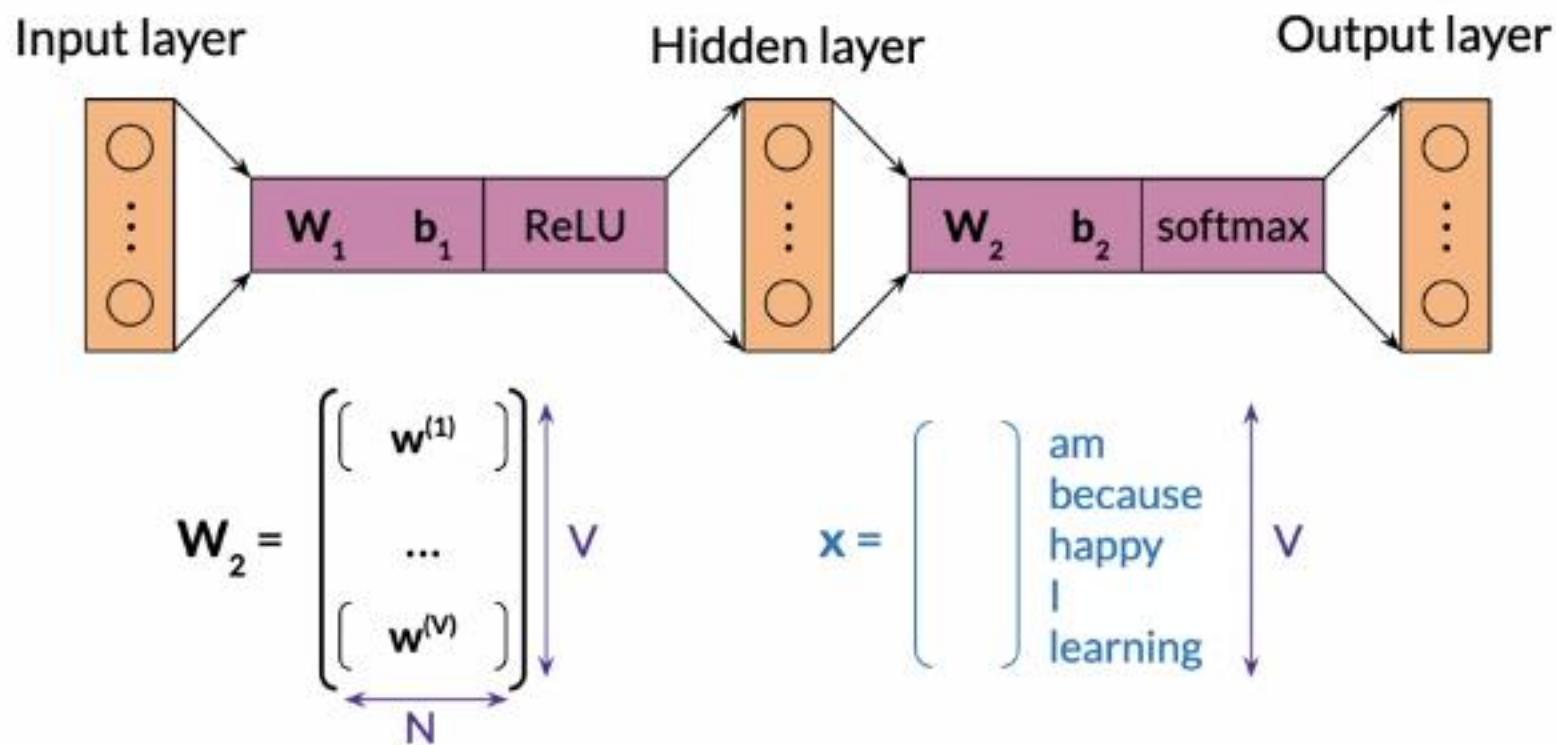
Extracting word embedding vectors: option 1



Extracting word embedding vectors: option 2



Extracting word embedding vectors: option 2



Extracting word embedding vectors: option 3

$$\mathbf{W}_1 = \begin{pmatrix} \mathbf{w}_1^{(1)} & \dots & \mathbf{w}_1^{(V)} \end{pmatrix}$$

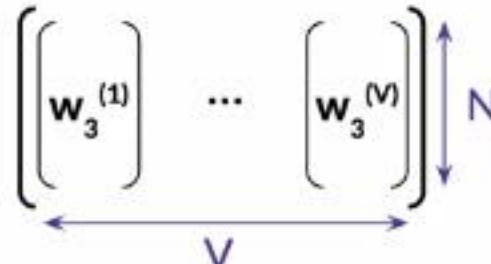
$$\mathbf{W}_2 = \begin{pmatrix} \mathbf{w}_2^{(1)} \\ \dots \\ \mathbf{w}_2^{(V)} \end{pmatrix}$$

Extracting word embedding vectors: option 3

$$\mathbf{W}_1 = \begin{pmatrix} \mathbf{w}_1^{(1)} & \dots & \mathbf{w}_1^{(V)} \end{pmatrix}$$

$$\mathbf{W}_2 = \begin{pmatrix} \mathbf{w}_2^{(1)} \\ \dots \\ \mathbf{w}_2^{(V)} \end{pmatrix}$$

$$\mathbf{W}_3 = 0.5 (\mathbf{W}_1 + \mathbf{W}_2^T) = \begin{pmatrix} \mathbf{w}_3^{(1)} & \dots & \mathbf{w}_3^{(V)} \end{pmatrix}$$



Extracting word embedding vectors: option 3

$$\mathbf{W}_1 = \begin{pmatrix} \mathbf{w}_1^{(1)} & \cdots & \mathbf{w}_1^{(V)} \end{pmatrix}$$

$$\mathbf{W}_2 = \begin{pmatrix} \mathbf{w}_2^{(1)} \\ \cdots \\ \mathbf{w}_2^{(V)} \end{pmatrix}$$

$$\mathbf{W}_3 = 0.5 (\mathbf{W}_1 + \mathbf{W}_2^T) = \begin{pmatrix} \mathbf{w}_3^{(1)} & \cdots & \mathbf{w}_3^{(V)} \end{pmatrix}$$

$\xleftarrow[V]{} \quad \xrightarrow[N]{} \quad \uparrow V$

$$\mathbf{x} = \begin{pmatrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{pmatrix}$$

Intrinsic evaluation

Intrinsic evaluation

Test relationships between words

Intrinsic evaluation

Test relationships between words

- Analogies

Intrinsic evaluation

Test relationships between words

- Analogies

Semantic analogies

Intrinsic evaluation

Test relationships between words

- Analogies

Semantic analogies

“France” is to “Paris” as “Italy” is to <?>

Syntactic analogies

Intrinsic evaluation

Test relationships between words

- Analogies

Semantic analogies

“France” is to “Paris” as “Italy” is to <?>

Syntactic analogies

“seen” is to “saw” as “been” is to <?>

Intrinsic evaluation

Test relationships between words

- Analogies

Semantic analogies

“France” is to “Paris” as “Italy” is to <?>

Syntactic analogies

“seen” is to “saw” as “been” is to <?>

 Ambiguity

Intrinsic evaluation

Test relationships between words

- Analogies

Semantic analogies

“France” is to “Paris” as “Italy” is to <?>

Syntactic analogies

“seen” is to “saw” as “been” is to <?>

⚡ Ambiguity

“wolf” is to “pack” as “bee” is to <?> → swarm? colony?

Intrinsic evaluation

Test relationships between words

- Analogies

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Intrinsic evaluation

Test relationships between words

- Analogies

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Intrinsic evaluation

Test relationships between words

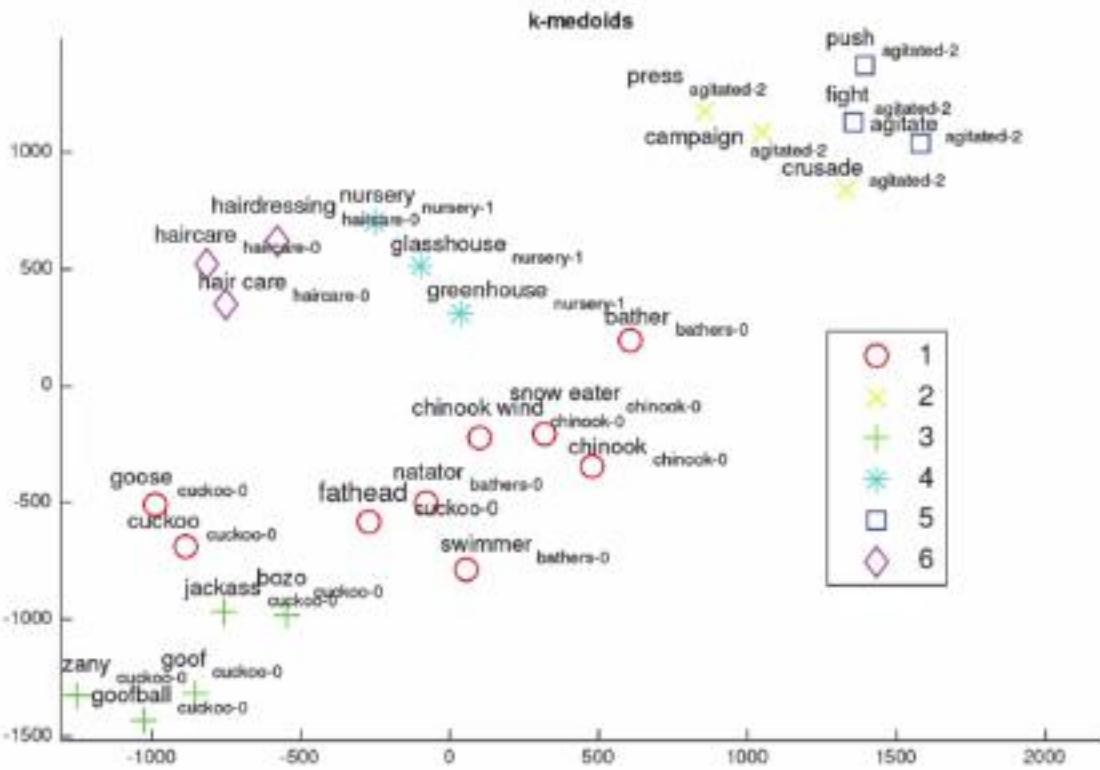
- Analogies
- Clustering

Intrinsic evaluation

Test relationships between words

- Analogies
- Clustering

Source: Michael Zhai, Johnny Tan, and Jinho D. Choi. 2016. [Intrinsic and extrinsic evaluations of word embeddings](#)



Intrinsic evaluation

Test relationships between words

- Analogies
- Clustering
- Visualization





deeplearning.ai

Evaluating Word Embeddings

Extrinsic Evaluation

Extrinsic evaluation

Test word embeddings on external task

Extrinsic evaluation

Test word embeddings on external task

e.g. named entity recognition, parts-of-speech tagging

Extrinsic evaluation

Test word embeddings on external task

e.g. named entity recognition, parts-of-speech tagging

Named entity

Andrew works at deeplearning.ai

Extrinsic evaluation

Test word embeddings on external task

e.g. named entity recognition, parts-of-speech tagging

Named entity

Andrew works at deeplearning.ai

person

Extrinsic evaluation

Test word embeddings on external task

e.g. named entity recognition, parts-of-speech tagging

Named entity

Andrew works at deeplearning.ai

person

Extrinsic evaluation

Test word embeddings on external task

e.g. named entity recognition, parts-of-speech tagging

Named entity

Andrew works at deeplearning.ai

person

organization

Extrinsic evaluation

Test word embeddings on external task

e.g. named entity recognition, parts-of-speech tagging

- + Evaluates actual usefulness of embeddings

Extrinsic evaluation

Test word embeddings on external task

e.g. named entity recognition, parts-of-speech tagging

- + Evaluates actual usefulness of embeddings
- Time-consuming

Extrinsic evaluation

Test word embeddings on external task

e.g. named entity recognition, parts-of-speech tagging

- + Evaluates actual usefulness of embeddings
- Time-consuming
- More difficult to troubleshoot



deeplearning.ai

Conclusion

Recap and assignment

Recap and assignment

- Data preparation
- Word representations

Recap and assignment

- Data preparation
- Word representations
- Continuous bag-of-words model

Recap and assignment

- Data preparation
- Word representations
- Continuous bag-of-words model
- Evaluation

Going further

- Advanced language modelling and word embeddings

Going further

- Advanced language modelling and word embeddings
- NLP and machine learning libraries

Going further

- Advanced language modelling and word embeddings
- NLP and machine learning libraries

Keras *# from keras.layers.embeddings import Embedding
embed_layer = Embedding(10000, 400)*

PyTorch *# import torch.nn as nn
embed_layer = nn.Embedding(10000, 400)*