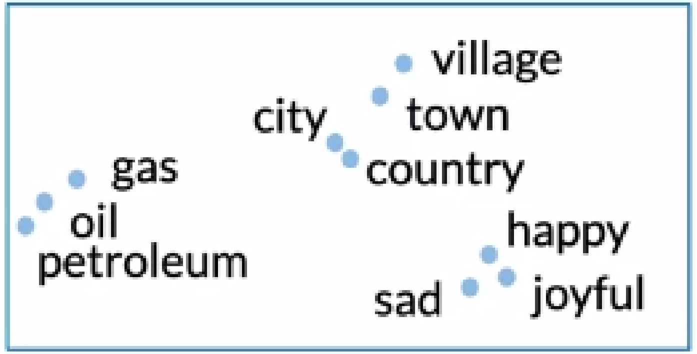
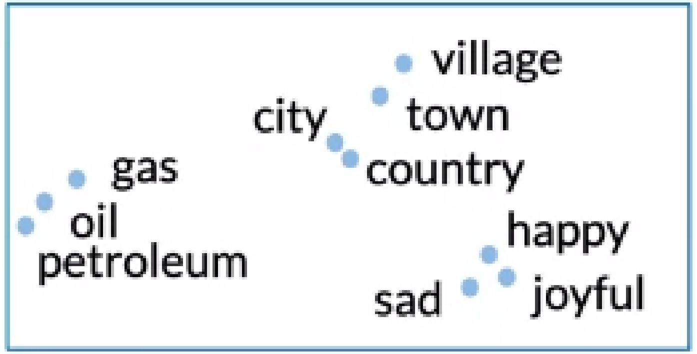
Some basic applications of word embeddings

Some basic applications of word embeddings



Semantic analogies and similarity

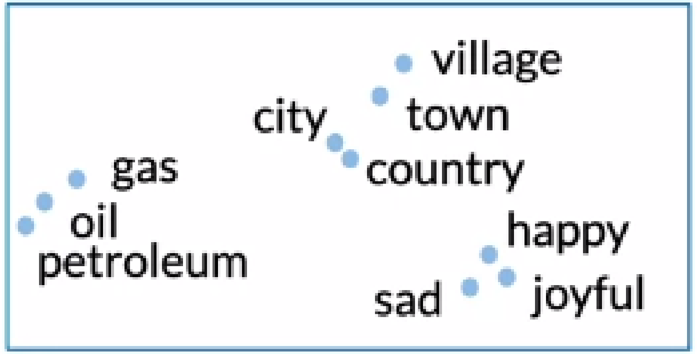
Some basic applications of word embeddings



Semantic analogies and similarity

Sentiment analysis

Some basic applications of word embeddings

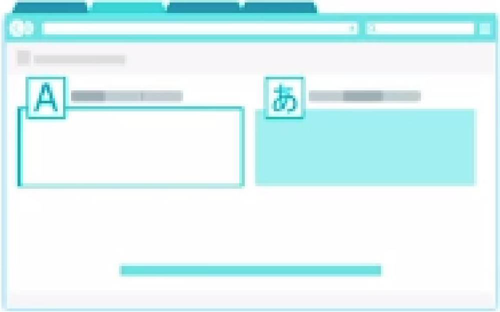


Semantic analogies and similarity

Sentiment analysis

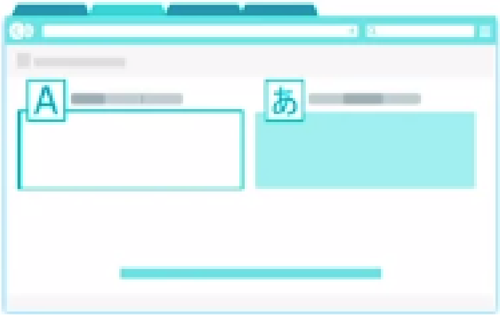
Classification of customer feedback

Advanced applications of word embeddings

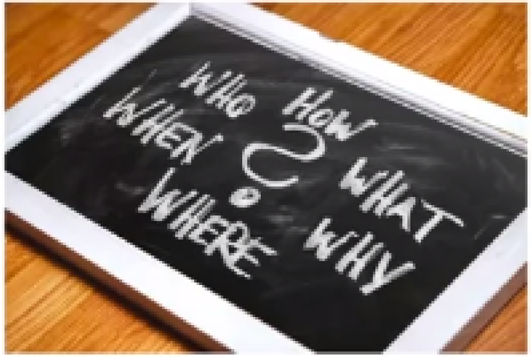


Machine translation

Advanced applications of word embeddings

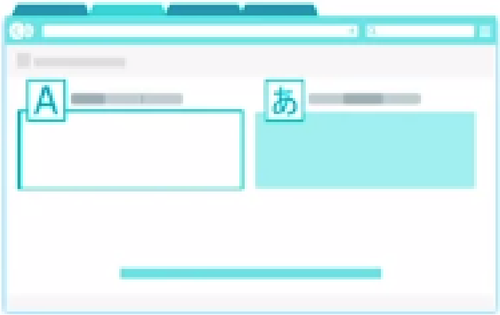


Machine translation

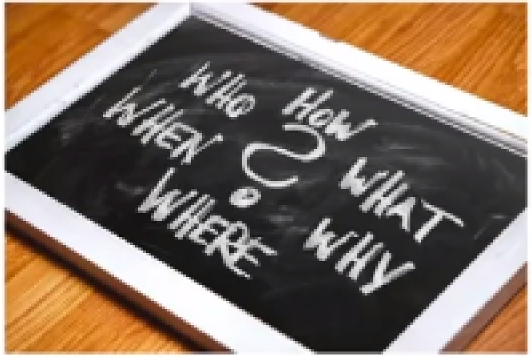


Information extraction

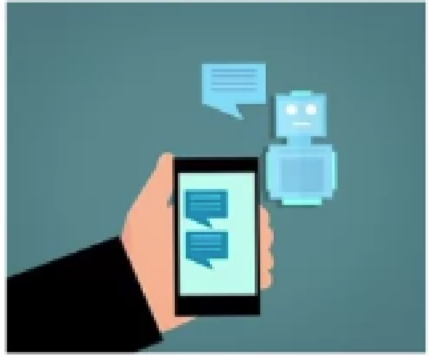
Advanced applications of word embeddings



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

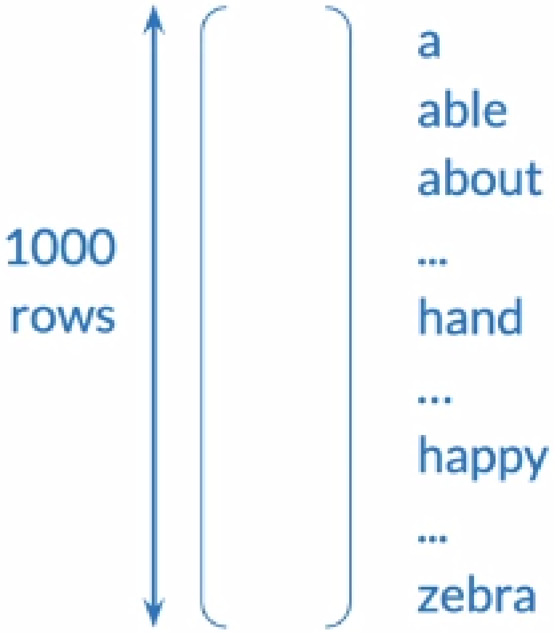
Integers

\+ Simple

\- Ordering: little semantic sense

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

One-hot vectors

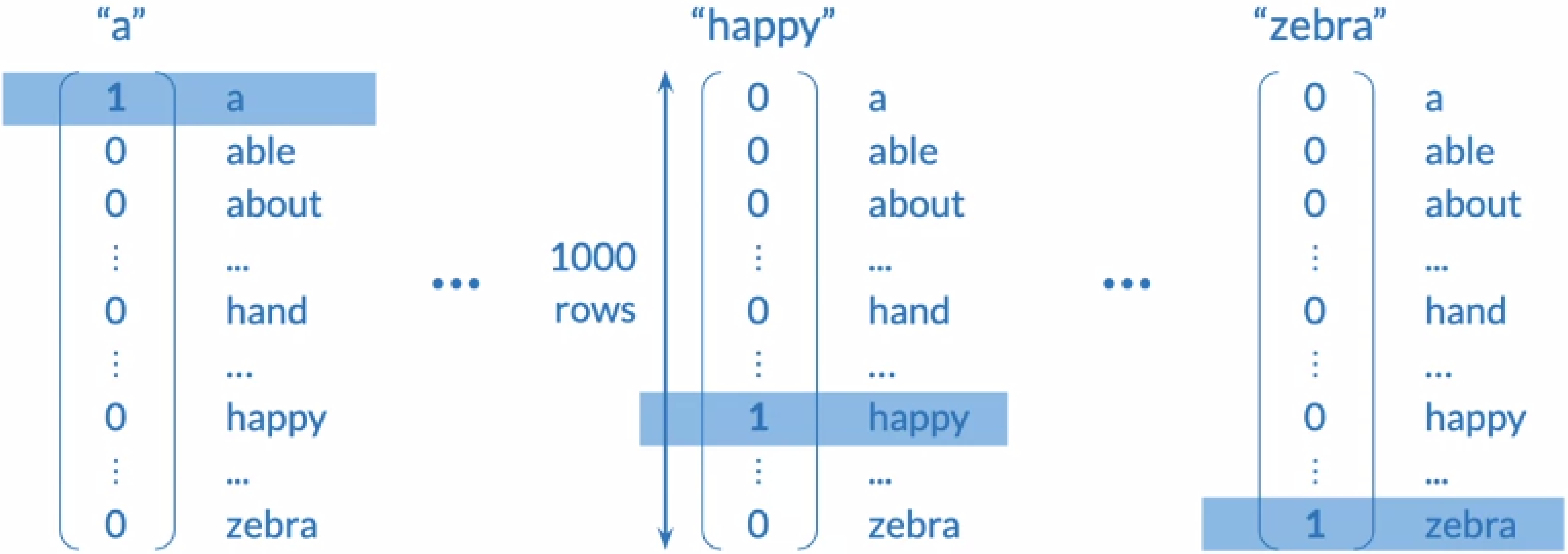


One-hot vectors

"happy"

$$\begin{array}{c}1000\\\text{rows}\end{array}\ \begin{pmatrix}0\\0\\0\\\vdots\\0\\\vdots\\1\\\vdots\\0\end{pmatrix}\ \begin{array}{l}\text{a}\\\text{able}\\\text{about}\\\dots\\\text{hand}\\\dots\\\text{happy}\\\dots\\\text{zebra}\end{array}$$

One-hot vectors



One-hot vectors

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

"happy"

$$\begin{array}{r} 1 \\ 2 \\ 3 \\ \cdots \\ 615 \\ \cdots \\ 621 \\ \cdots \\ 1000 \end{array} \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \begin{array}{l} \text{a} \\ \text{able} \\ \text{about} \\ \cdots \\ \text{hand} \\ \cdots \\ \text{happy} \\ \cdots \\ \text{zebra} \end{array}$$

One-hot vectors

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

"happy"

$$
\begin{array}{r c l}
1 & \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array}\right) & \begin{array}{l} \text{a} \\ \text{able} \\ \text{about} \\ \ldots \\ \text{hand} \\ \ldots \\ \text{happy} \\ \ldots \\ \text{zebra} \end{array}
\end{array}
$$

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

$$\sim\text{10k—1M+ rows} \quad \left(\begin{array}{c}0\\ \vdots\\ 1\\ \vdots\\ 0\end{array}\right) \quad \begin{array}{l}\text{a}\\ \text{...}\\ \text{happy}\\ \text{...}\\ \text{Zyzzyva}\end{array}$$

One-hot vectors

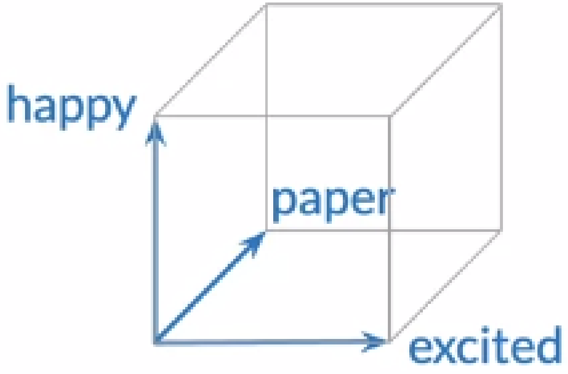
+ Simple
+ No implied ordering
- Huge vectors
- No embedded meaning

$$\begin{array}{r} \text{~10k—1M+} \\ \text{rows} \end{array} \updownarrow \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \begin{array}{l} \text{a} \\ \ldots \\ \text{happy} \\ \ldots \\ \text{Zyzzyva} \end{array}$$

One-hot vectors

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

$$\text{~10k—1M+ rows} \quad \begin{array}{c} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \end{array} \quad \begin{array}{l} \text{a} \\ \text{...} \\ \text{happy} \\ \text{...} \\ \text{Zyzzyva} \end{array}$$

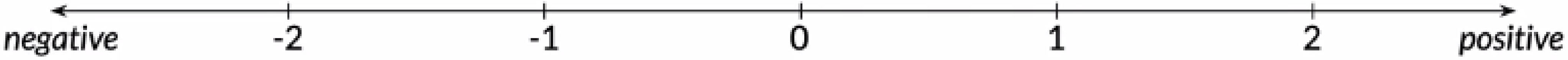


d(paper, excited)
= d(paper, happy)
= d(excited, happy)

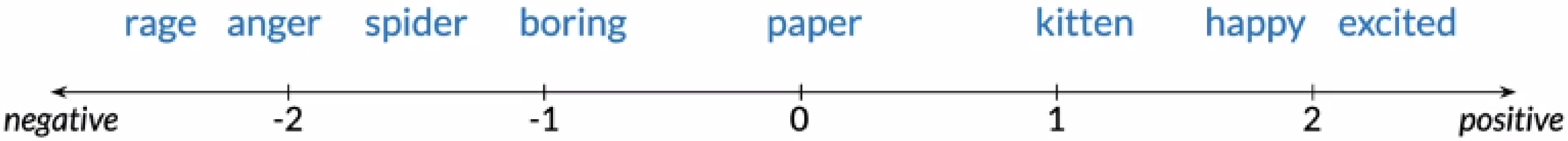
Word Embeddings

deeplearning.ai

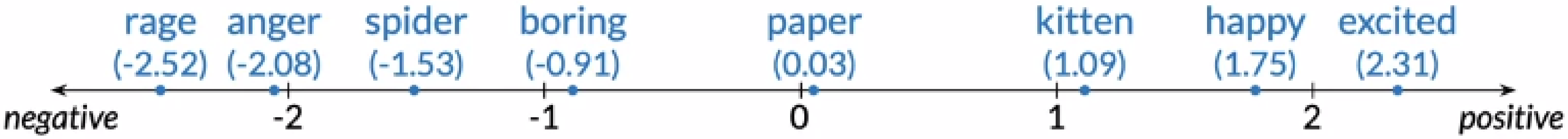
Meaning as vectors



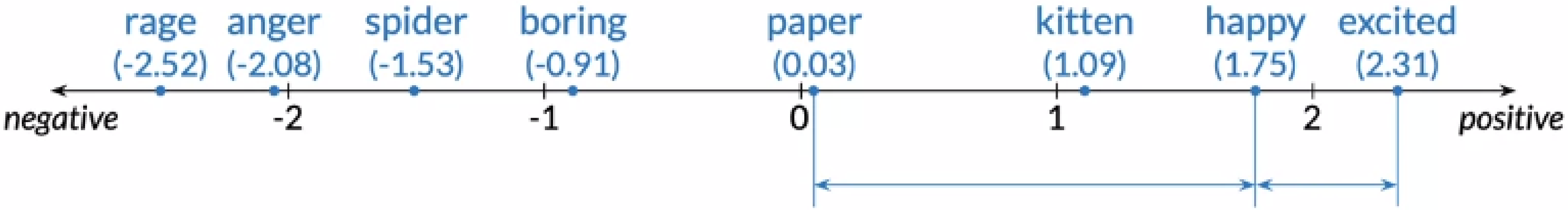
Meaning as vectors



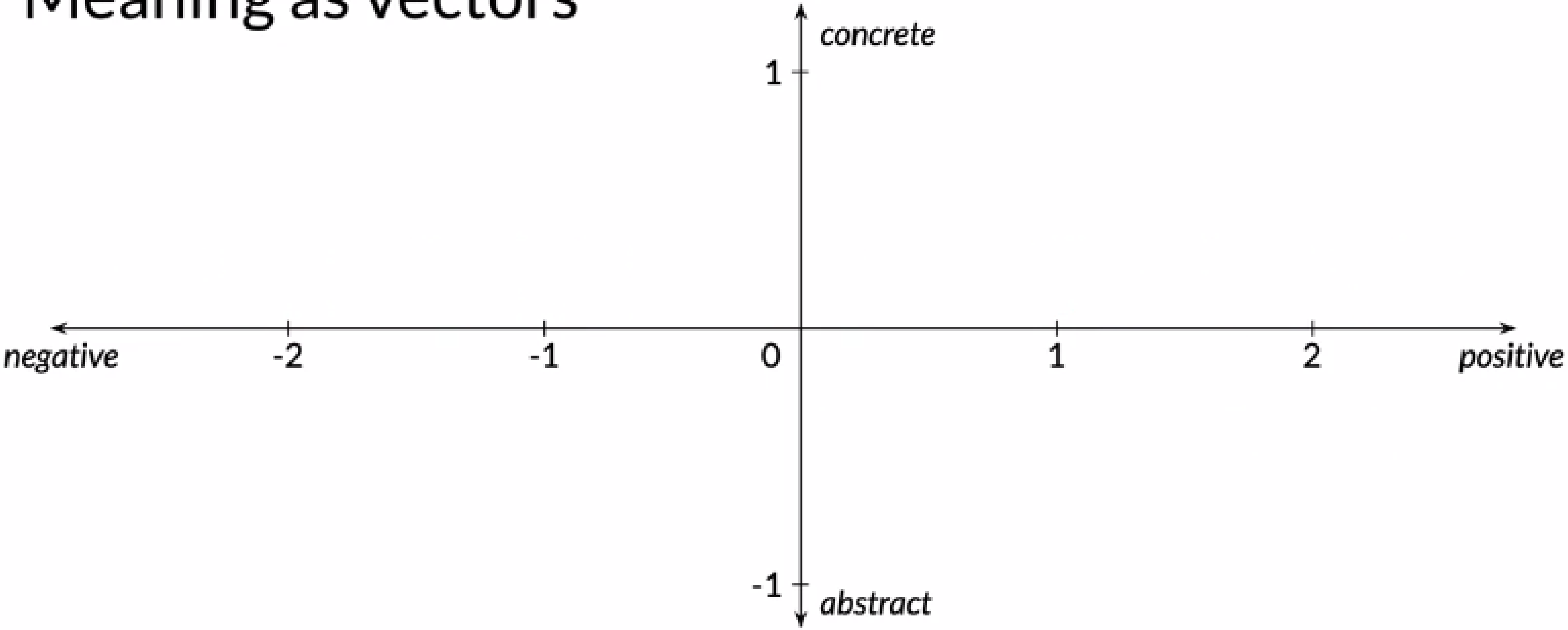
Meaning as vectors



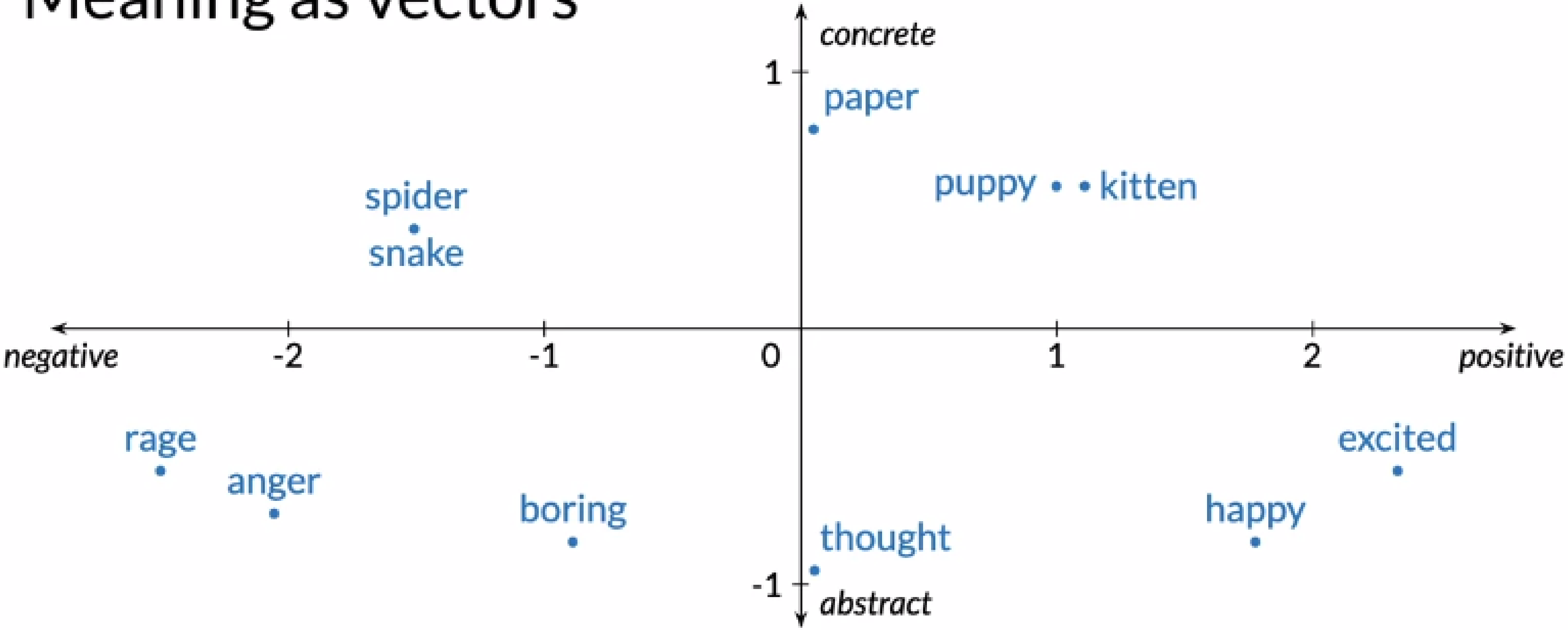
Meaning as vectors



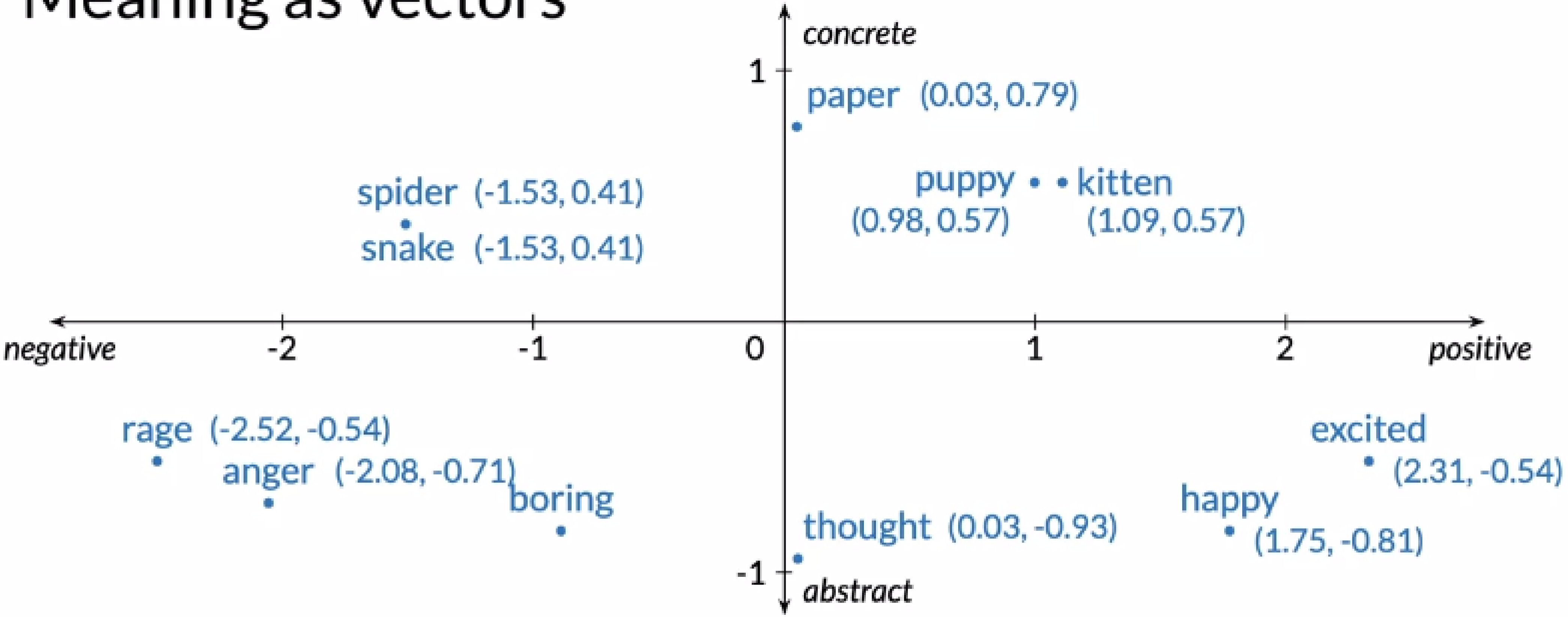
Meaning as vectors



Meaning as vectors



Meaning as vectors



Word embedding vectors

Word embedding vectors

+ Low dimension

Word embedding vectors

+ Low dimension

"happy"

$$\text{\textasciitilde 100---\textasciitilde 1000 rows} \; \updownarrow \begin{pmatrix} 0.123 \\ \vdots \\ -4.059 \\ \vdots \\ 1.891 \end{pmatrix}$$

Word embedding vectors

+ Low dimension

+ Embed meaning

"happy"

~100—~1000 rows $\begin{pmatrix} 0.123 \\ \vdots \\ -4.059 \\ \vdots \\ 1.891 \end{pmatrix}$

Word embedding vectors

+ Low dimension

+ Embed meaning
    - e.g. semantic distance

"happy"

~100—~1000 rows

$$\begin{pmatrix} 0.123 \\ \vdots \\ -4.059 \\ \vdots \\ 1.891 \end{pmatrix}$$

Word embedding vectors

+ Low dimension

+ Embed meaning
    - e.g. semantic distance

forest ≈ tree   forest ≉ ticket

"happy"

~100—~1000 rows

$$\begin{pmatrix} 0.123 \\ \vdots \\ -4.059 \\ \vdots \\ 1.891 \end{pmatrix}$$

Word embedding vectors

+ Low dimension

+ Embed meaning
  - e.g. semantic distance

    forest ≈ tree   forest ≉ ticket

  - e.g. analogies

“happy”

$$\text{\textasciitilde 100—\textasciitilde 1000 rows} \quad \begin{pmatrix} 0.123 \\ \vdots \\ -4.059 \\ \vdots \\ 1.891 \end{pmatrix}$$

Word embedding vectors

+ Low dimension

+ Embed meaning
  - e.g. semantic distance

    forest ≈ tree   forest ≉ ticket

  - e.g. analogies

    Paris:France :: Rome:?

"happy"

~100—~1000 rows $\begin{pmatrix} 0.123 \\ \vdots \\ -4.059 \\ \vdots \\ 1.891 \end{pmatrix}$

Terminology

integers | one-hot vectors | word embedding vectors

Terminology

word vectors

integers

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

Corpus

General-purpose
e.g. Wikipedia

Words in context

Word embedding process

Corpus

General-purpose
e.g. Wikipedia

Specialized

Words in context

Word embedding process

Corpus

General-purpose
e.g. Wikipedia

Specialized
e.g. contracts, law books

Words in context

Word embedding process

Corpus

General-purpose
e.g. Wikipedia

Specialized
e.g. contracts, law books

Words in context

Embedding method

Word embedding process

Corpus

General-purpose
e.g. Wikipedia

Specialized
e.g. contracts, law books

Words in context

Embedding method

Machine learning model

Word embedding process

Corpus

General-purpose
e.g. Wikipedia

Specialized
e.g. contracts, law books

Words in context

Embedding method

Machine learning model

Learning task

Word embeddings

Word embedding process

Corpus

General-purpose
e.g. Wikipedia

Specialized
e.g. contracts, law books

Words in context

Embedding method

Machine learning model

Learning task

"I think [???] I am"

Word embeddings

Word embedding process

Corpus

General-purpose
e.g. Wikipedia

Specialized
e.g. contracts, law books

Words in context

Embedding method

Machine learning model

Learning task
"I think [???] I am"

Meaning

Word embeddings

Word embedding process

Corpus

General-purpose
e.g. Wikipedia

Specialized
e.g. contracts, law books

Words in context

Embedding method

Machine learning model

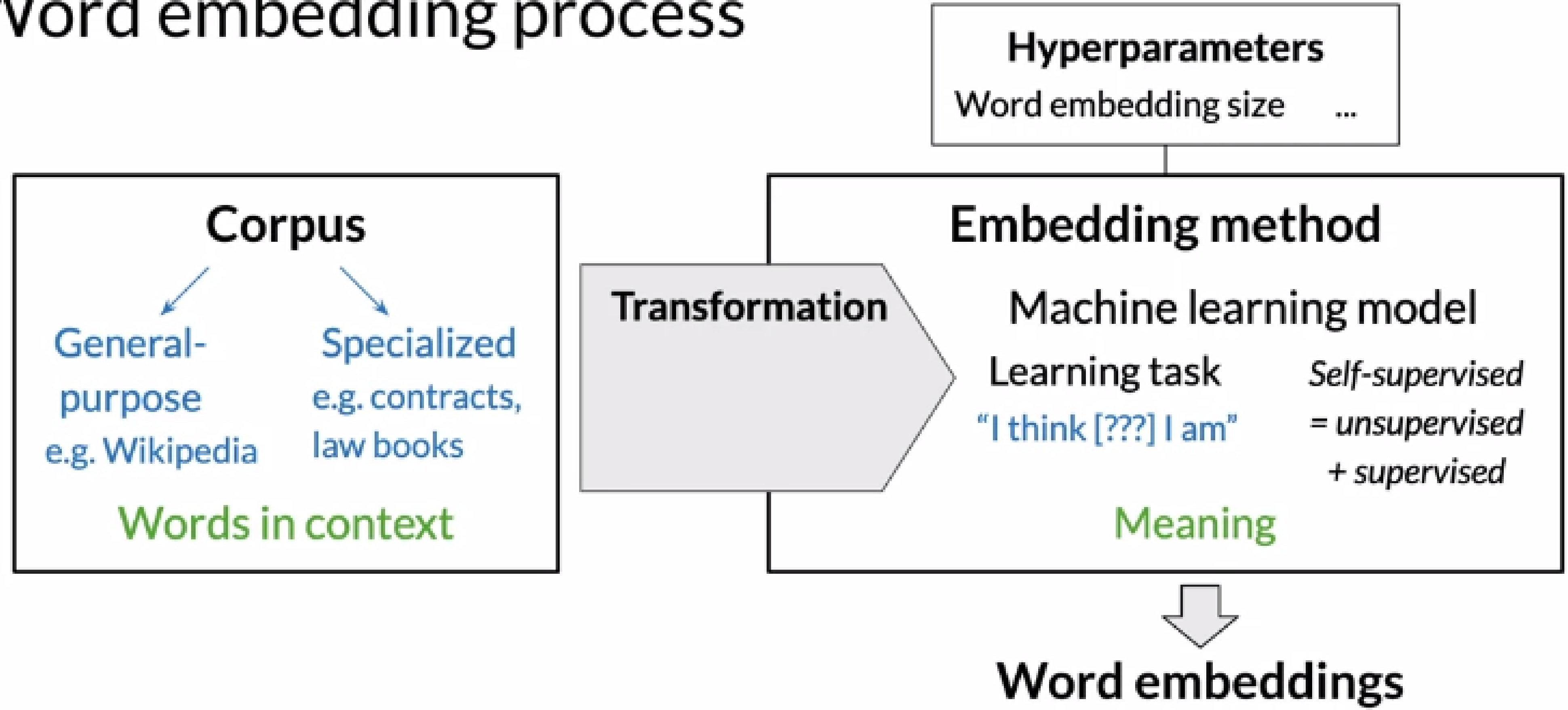
Learning task
"I think [???] I am"

Self-supervised
= unsupervised
+ supervised

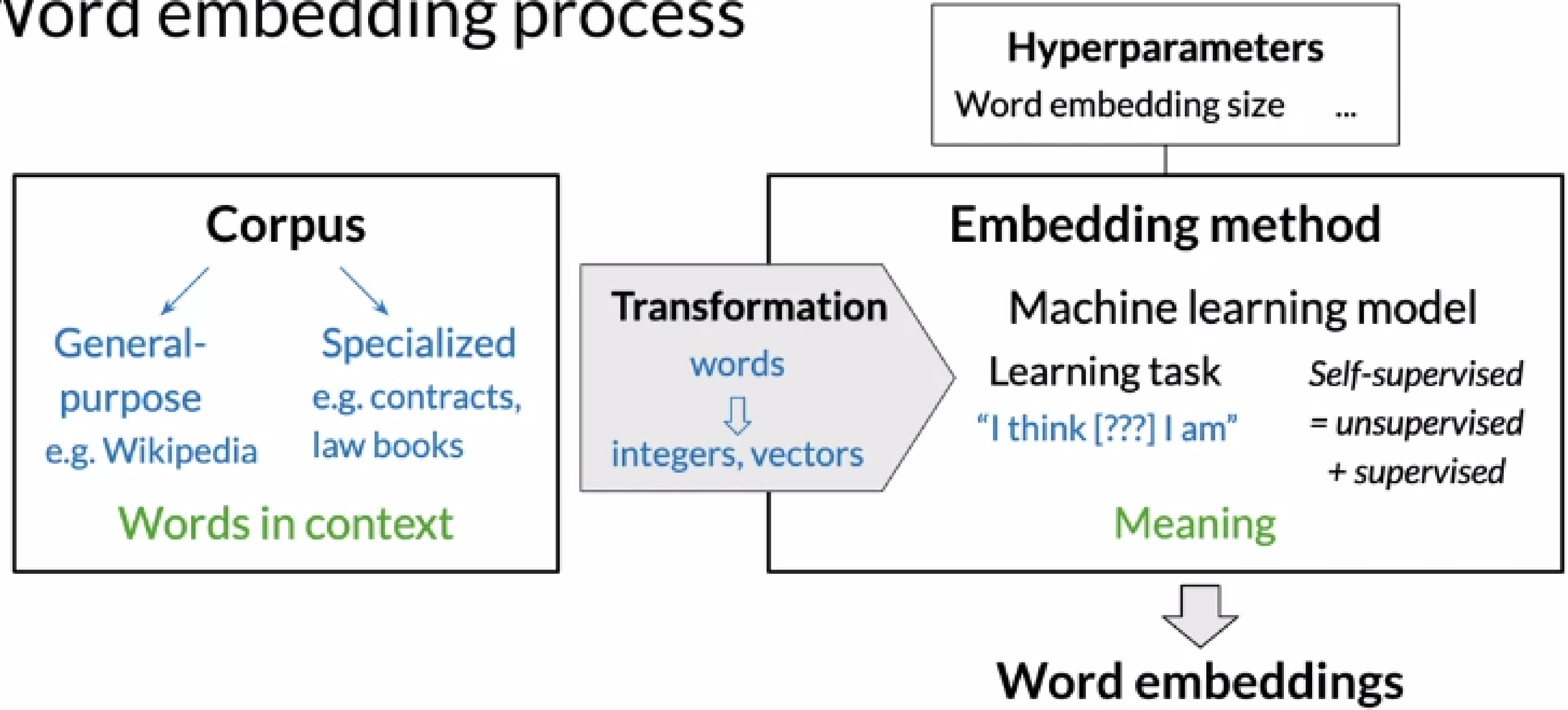
Meaning

Word embeddings

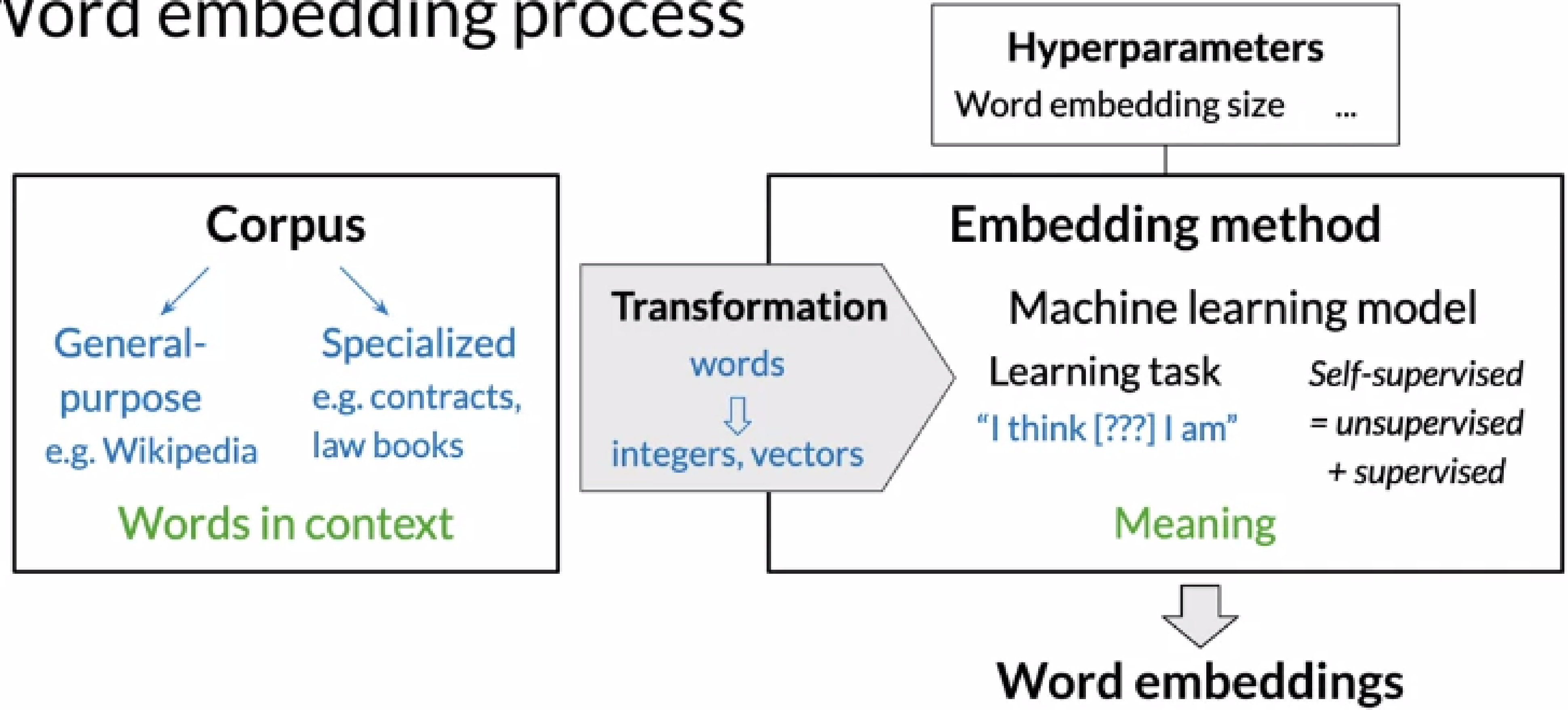
Word embedding process



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

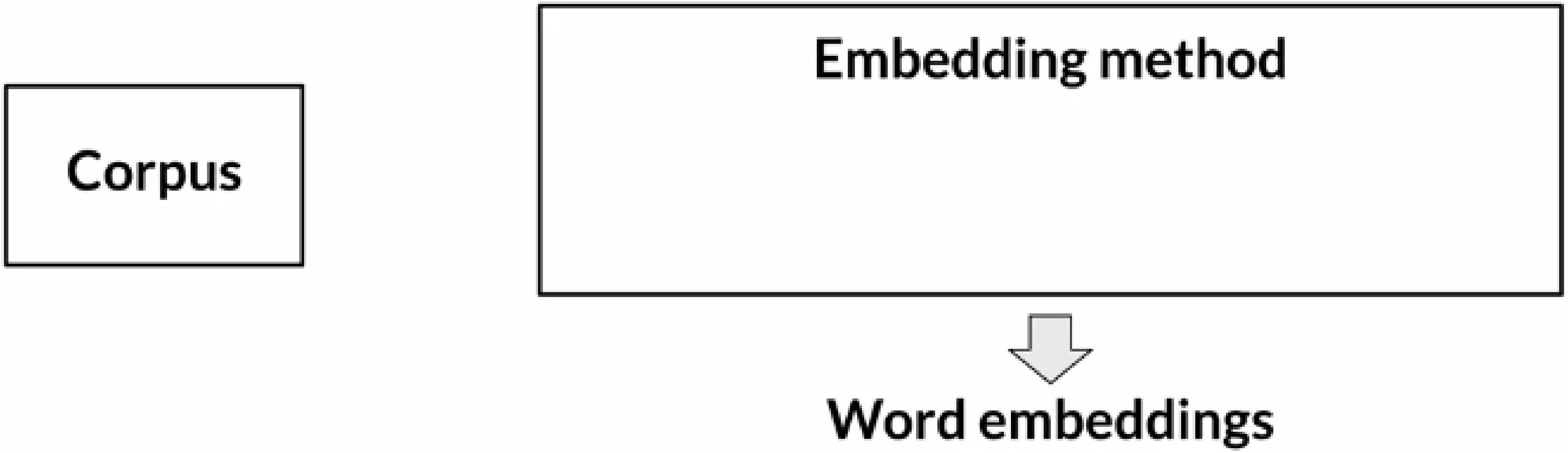
Continuous bag-of-words word embedding process

Continuous bag-of-words word embedding process

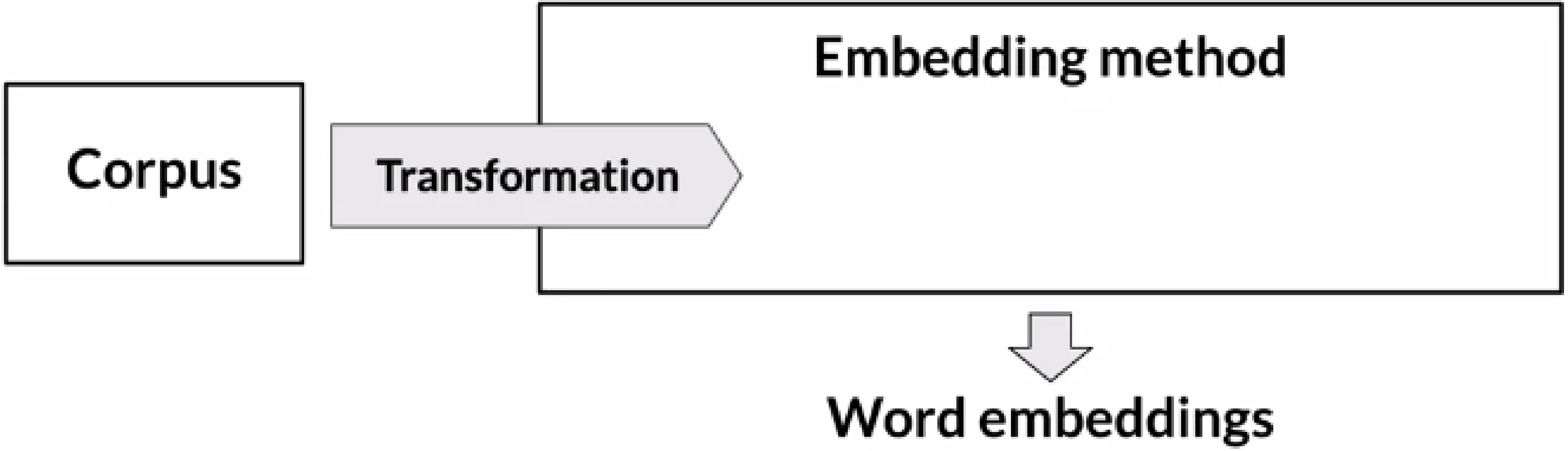
Corpus

Embedding method

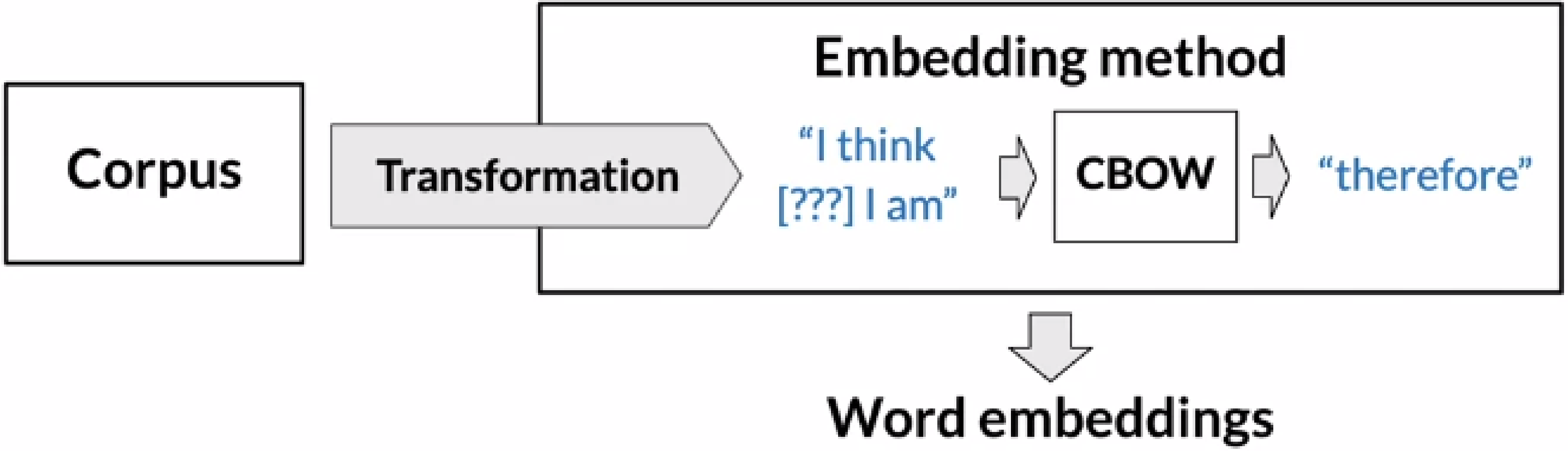
Continuous bag-of-words word embedding process



Continuous bag-of-words word embedding process



Continuous bag-of-words word embedding process



Center word prediction: rationale

Center word prediction: rationale

The little ______?______ is barking

Corpus → Transformation → CBOW

Center word prediction: rationale

The little ________?________ is barking

dog

puppy

hound

terrier

...

Creating a training example

I am happy because I am learning

Creating a training example

center word

I am **happy** because I am learning

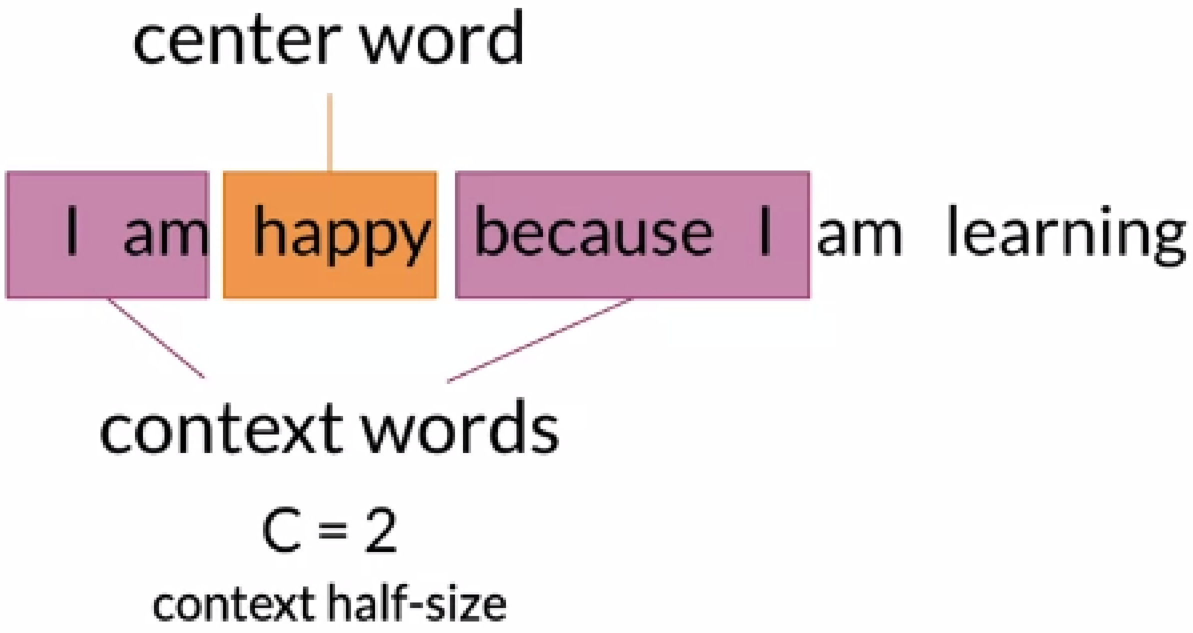
Creating a training example

center word

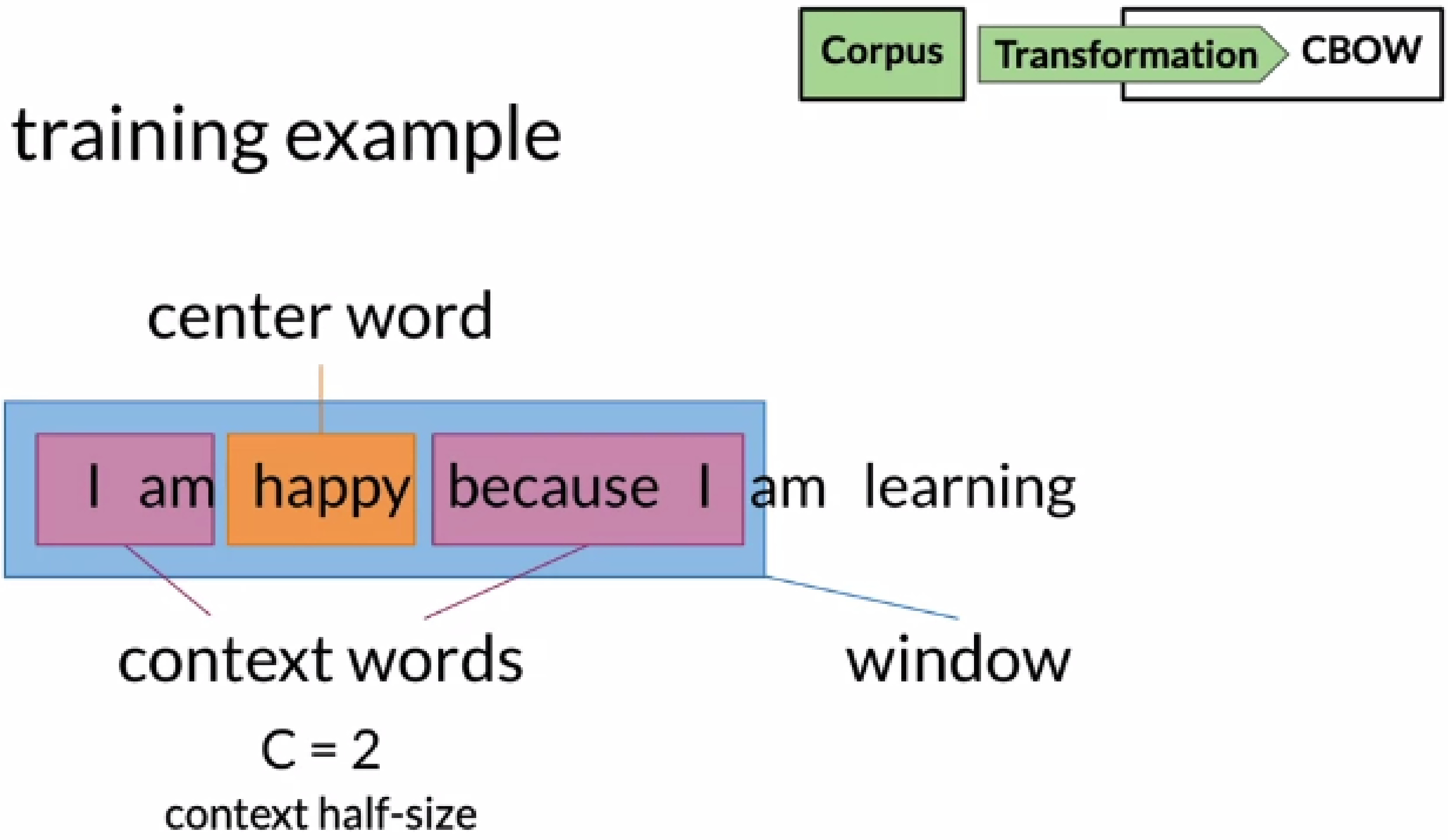
I am happy because I am learning

context words

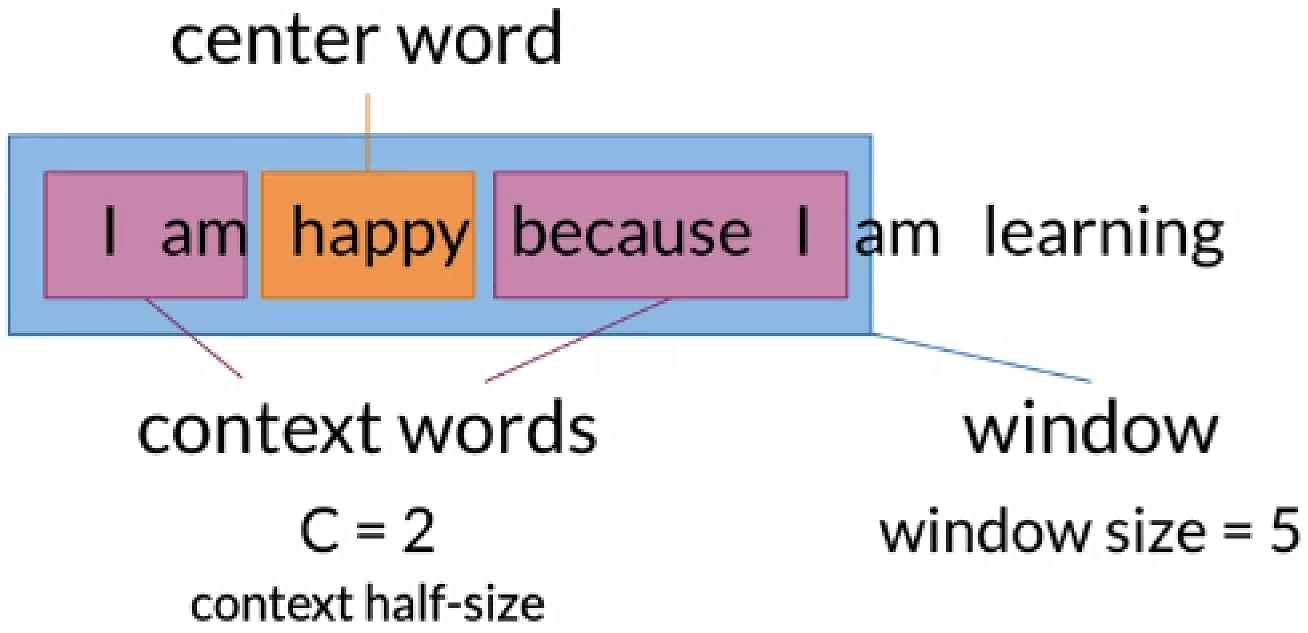
Creating a training example



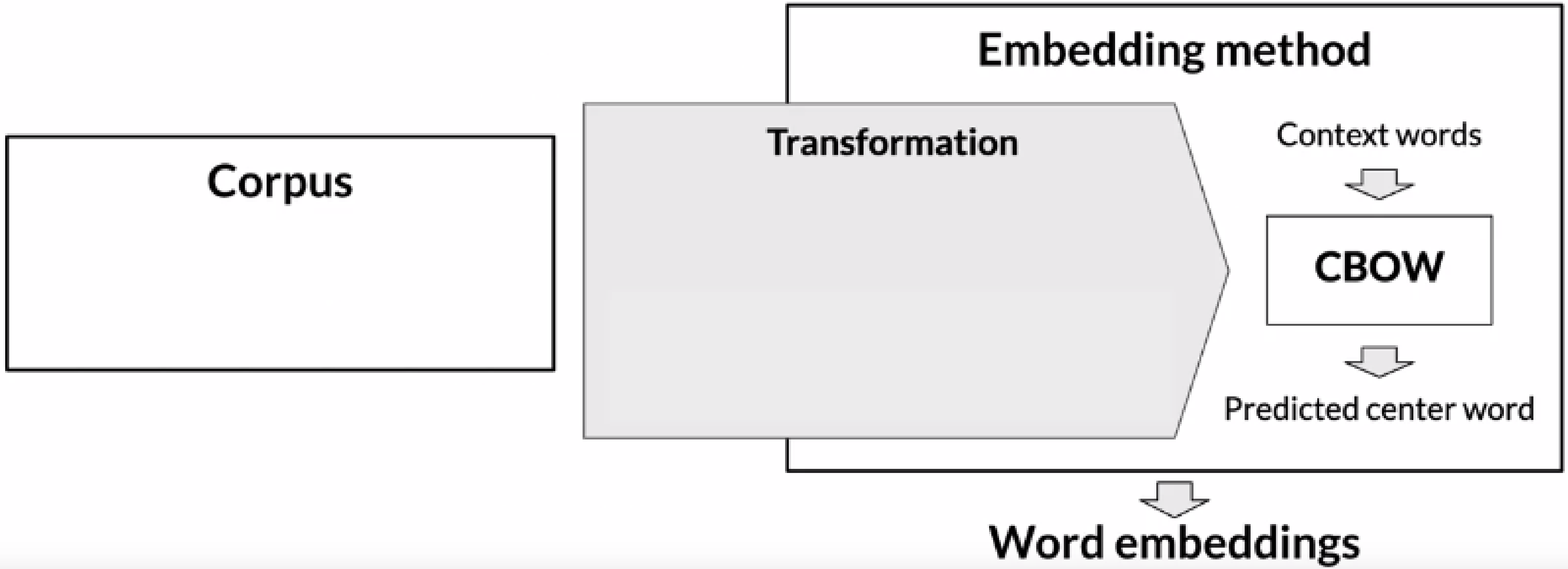
Creating a training example



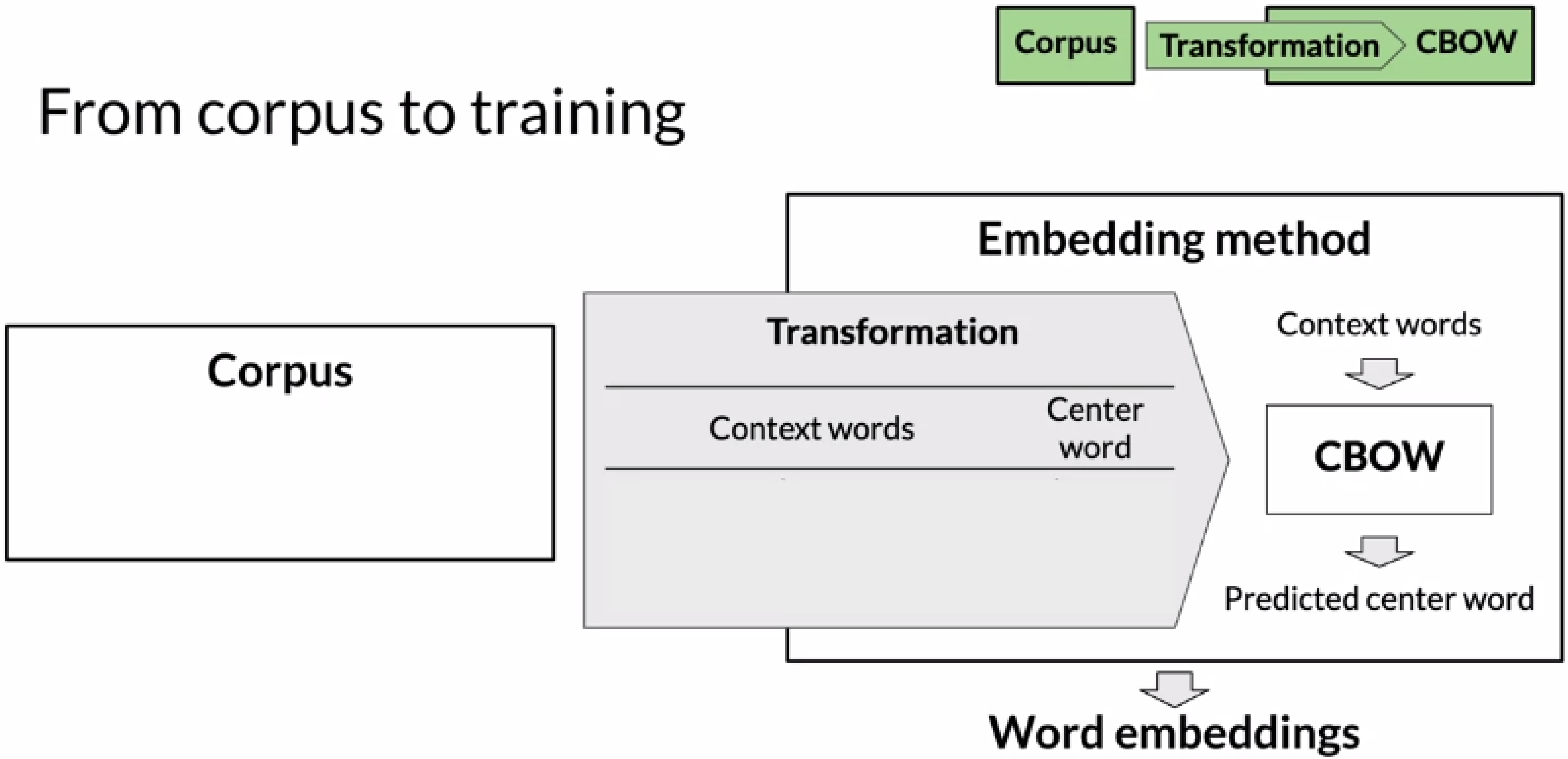
Creating a training example



From corpus to training

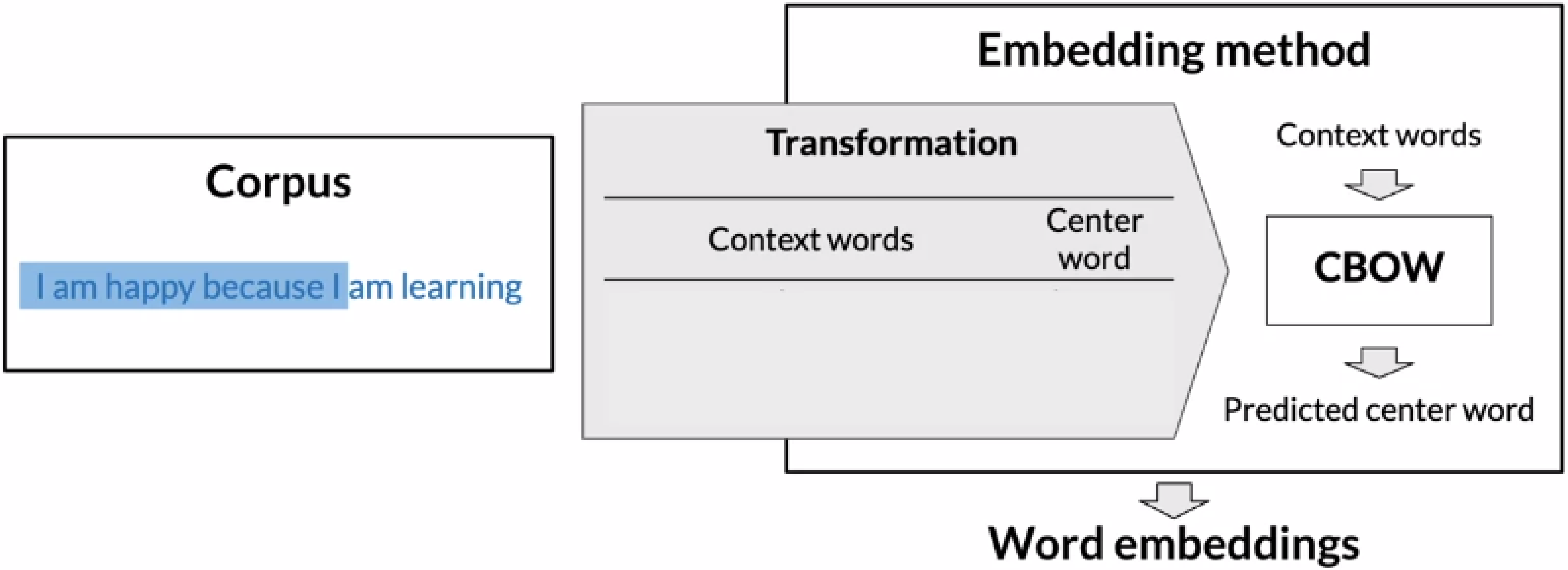


From corpus to training



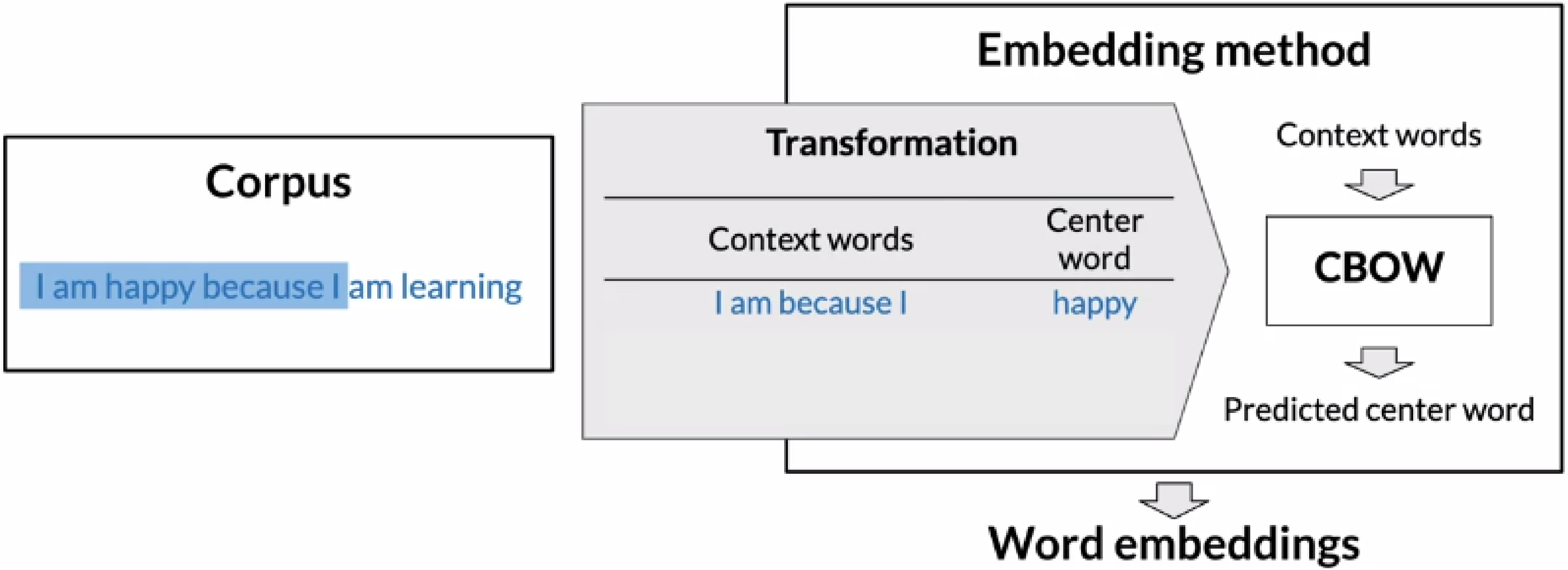
Corpus Transformation CBOW

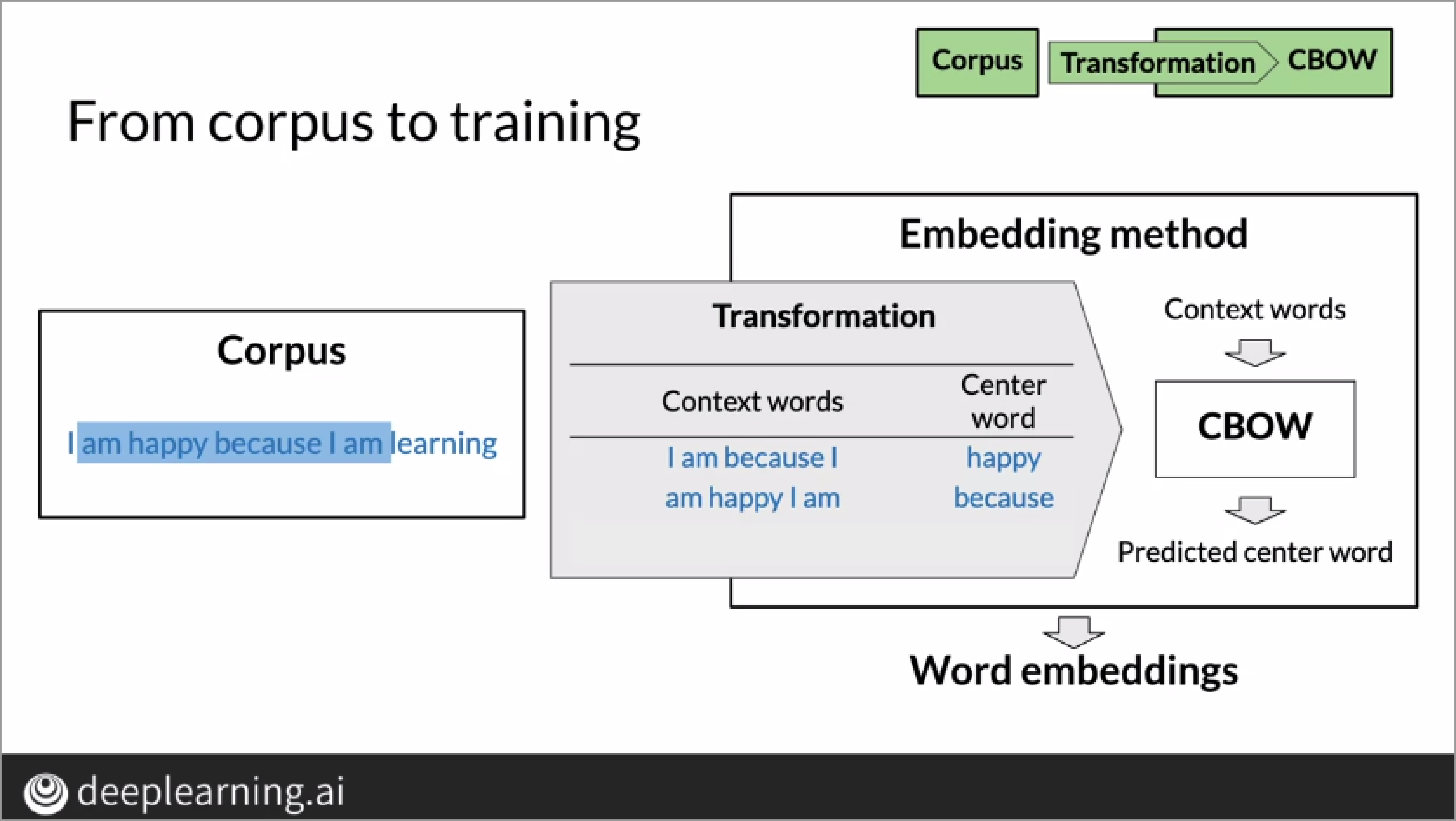
From corpus to training



Corpus Transformation CBOW

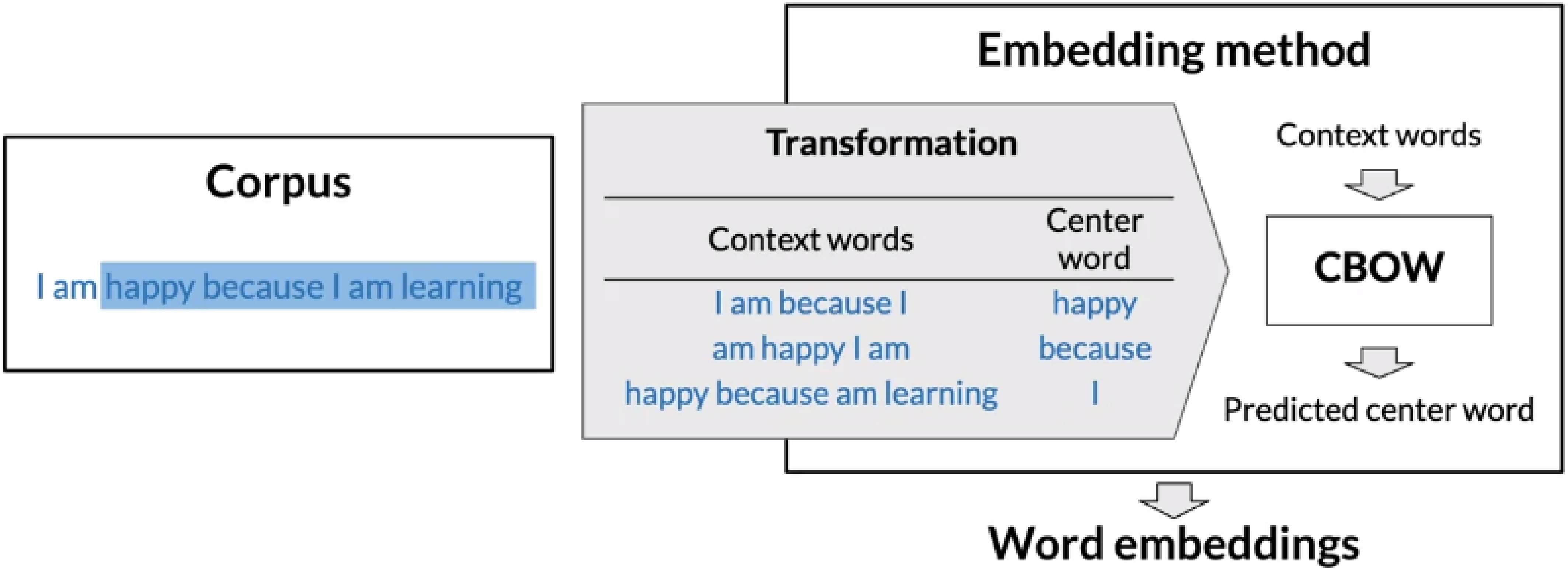
From corpus to training



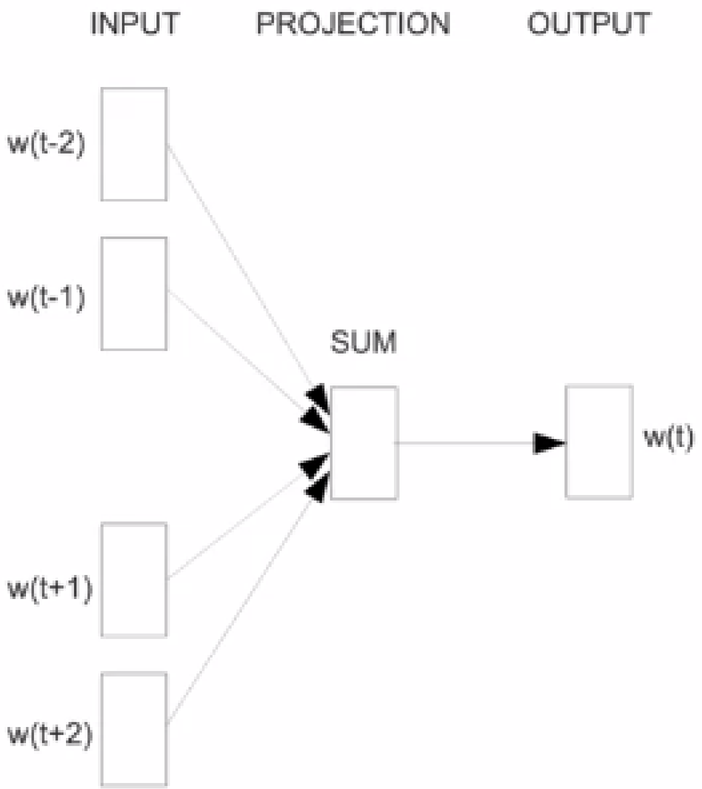


Corpus
Transformation
CBOW
From corpus to training
Embedding method
Corpus
I am happy because I am learning
Transformation
Context words
Center word
I am because I
happy
am happy I am
because
Context words
CBOW
Predicted center word
Word embeddings
deeplearning.ai

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 / upper case*

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → *lowercase / upper case*
- Punctuation

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → *lowercase / upper case*

- Punctuation , ! . ? → .

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → *lowercase / upper case*
- Punctuation , ! . ? → . " ' « » ' " → ∅

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → *lowercase / upper case*
- Punctuation , ! . ? → . " ' « » ' " → ∅ ... !! ??? → .

Cleaning and tokenization matters

- Letter case "The" == "the" == "THE" → *lowercase / upper case*
- Punctuation , ! . ? → . “ ‘ « » ’ ” → ∅ ... !! ??? → .
- Numbers

Cleaning and tokenization matters

- Letter case — "The" == "the" == "THE" → *lowercase / upper case*
- Punctuation — , ! . ? → .   " ' « » ' " → ∅   ... !! ??? → .
- Numbers — 1 2 3 5 8 → ∅   3.14159 90210

Cleaning and tokenization matters

- Letter case — "The" == "the" == "THE" → *lowercase / upper case*
- Punctuation — , ! . ? → .   " ' « » ' " → ∅   ... !! ??? → .
- Numbers — 1 2 3 5 8 → ∅   3.14159 90210

Cleaning and tokenization matters

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

Cleaning and tokenization matters

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

Cleaning and tokenization matters

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

Cleaning and tokenization matters

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

Cleaning and tokenization matters

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

Cleaning and tokenization matters

- Letter case — "The" == "the" == "THE" → *lowercase / upper case*
- Punctuation — , ! . ? → .   " ' « » ' " → ∅   ... !! ??? → .
- Numbers — 1 2 3 5 8 → ∅   3.14159 90210 → *as is/<NUMBER>*
- Special characters — ∇ $ € § ¶ ** → ∅
- Special words — 😊 #nlp

Cleaning and tokenization matters

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

```python
# 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

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

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

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

Example in Python: code

```python
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', '.']

Sliding Window of Words in Python

deeplearning.ai

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

```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

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

```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

```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

```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):
    ...
        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
```

Transforming Words into Vectors

deeplearning.ai

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

$$\begin{array}{r} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}$$

Transforming center words into vectors

Corpus: I am happy because I am learning

Vocabulary: am, because, happy, I, learning

One-hot vector: am

$$\begin{array}{r} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Transforming center words into vectors

Corpus: I am happy because I am learning

Vocabulary: am, because, happy, I, learning

One-hot vector

$$\begin{array}{r c c} & \text{am} & \text{because} \\ \text{am} & \multirow{5}{*}{$\begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix}$} & \multirow{5}{*}{$\begin{pmatrix}0\\1\\0\\0\\0\end{pmatrix}$} \\ \text{because} & & \\ \text{happy} & & \\ \text{I} & & \\ \text{learning} & & \end{array}$$

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

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 I

Transforming context words into vectors

Average of individual one-hot vectors

I am because I

$$\begin{array}{r} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Transforming context words into vectors

Average of individual one-hot vectors

$$\begin{array}{r c c c c} & \text{I} & \text{am} & \text{because} & \text{I} \\ \begin{array}{r} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} & & & \end{array}$$

Transforming context words into vectors

Average of individual one-hot vectors

I am because I

$$
\begin{array}{r}
\text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning}
\end{array}
\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}
\quad
\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

Transforming context words into vectors

Average of individual one-hot vectors

$$
\begin{array}{rcccc}
 & \text{I} & \text{am} & \text{because} & \text{I} \\
\begin{array}{r}\text{am}\\ \text{because}\\ \text{happy}\\ \text{I}\\ \text{learning}\end{array} &
\begin{pmatrix}0\\0\\0\\1\\0\end{pmatrix} &
\begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix} &
\begin{pmatrix}0\\1\\0\\0\\0\end{pmatrix} &
\begin{pmatrix}0\\0\\0\\1\\0\end{pmatrix}
\end{array}
$$

Transforming context words into vectors

Average of individual one-hot vectors

$$\left(\begin{array}{r} \\ \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} \begin{array}{c} \text{I} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{array} + \begin{array}{c} \text{am} \\ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{array} + \begin{array}{c} \text{because} \\ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{array} + \begin{array}{c} \text{I} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{array} \right) / 4 = \begin{array}{c} \text{I am because I} \\ \begin{pmatrix} 0.25 \\ 0.25 \\ 0 \\ 0.5 \\ 0 \end{pmatrix} \end{array}$$

Transforming context words into vectors

Average of individual one-hot vectors

$$
\left(
\begin{array}{r}
\\ \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning}
\end{array}
\begin{array}{c}
\text{I} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}
\end{array}
+
\begin{array}{c}
\text{am} \\ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\end{array}
+
\begin{array}{c}
\text{because} \\ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\end{array}
+
\begin{array}{c}
\text{I} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}
\end{array}
\right) / 4
=
\begin{array}{c}
\text{I am because I} \\ \begin{pmatrix} 0.25 \\ 0.25 \\ 0 \\ 0.5 \\ 0 \end{pmatrix}
\end{array}
$$

Final prepared training set

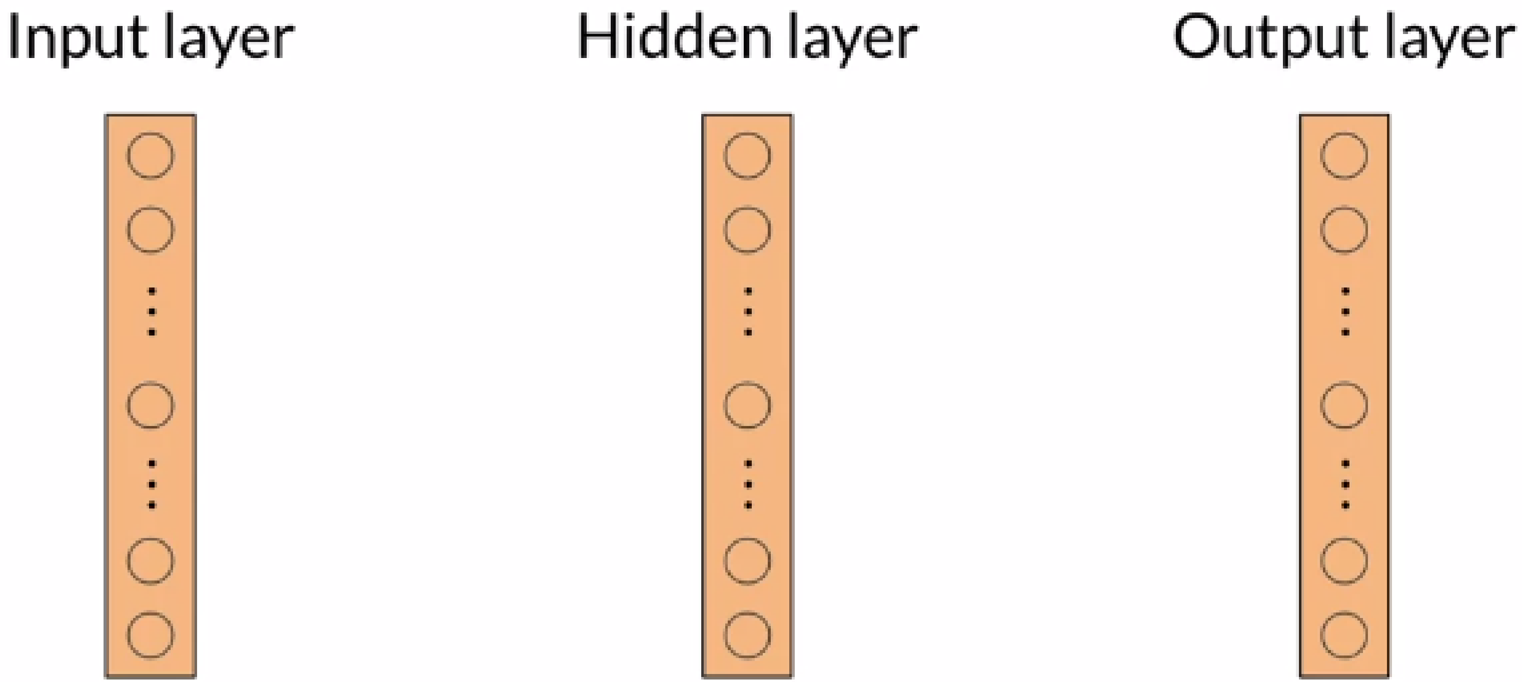
Context words	Context words vector	*Center word*	Center word vector
I am because I	[0.25; 0.25; 0; 0.5; 0]	*happy*	[0; 0; 1; 0; 0]

Final prepared training set

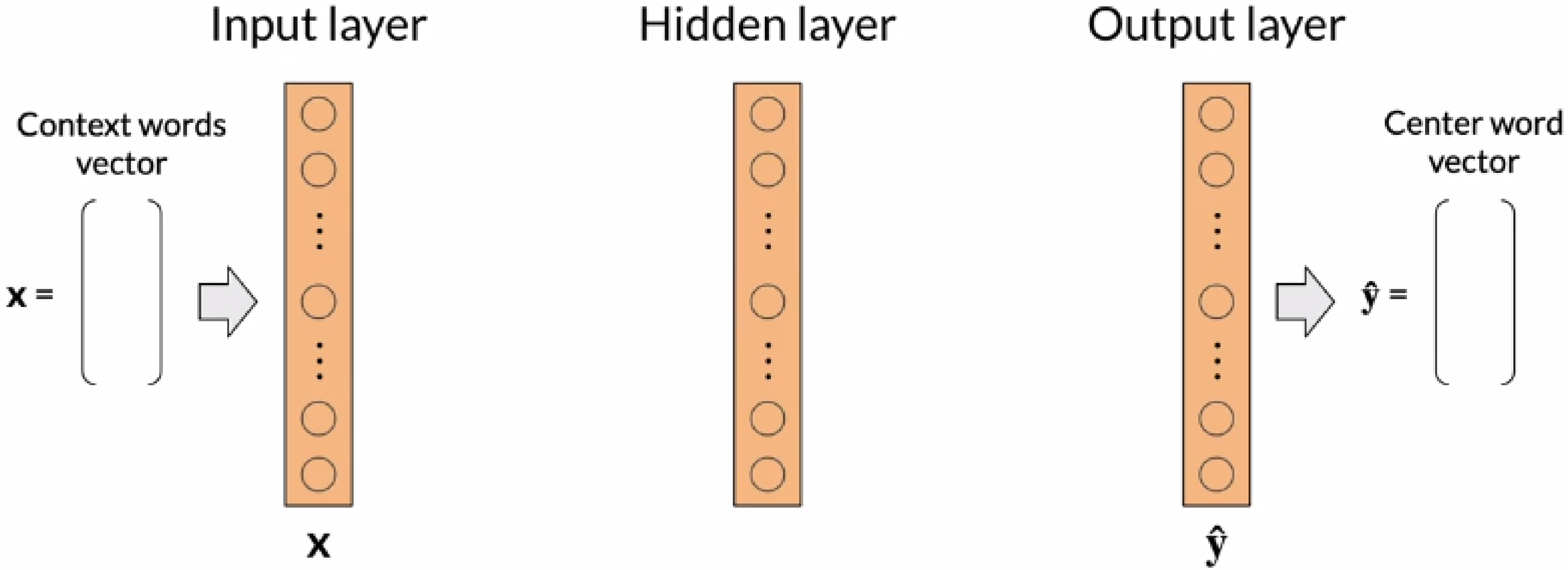
Context words	Context words vector	*Center word*	Center word vector
I am because I	[0.25; 0.25; 0; 0.5; 0]	*happy*	[0; 0; 1; 0; 0]
am happy I am	[0.5; 0; 0.25; 0.25; 0]	*because*	[0; 1; 0; 0; 0]
happy because am learning	[0.25; 0.25; 0.25; 0; 0.25]	*I*	[0; 0; 0; 1; 0]

Architecture of the CBOW model

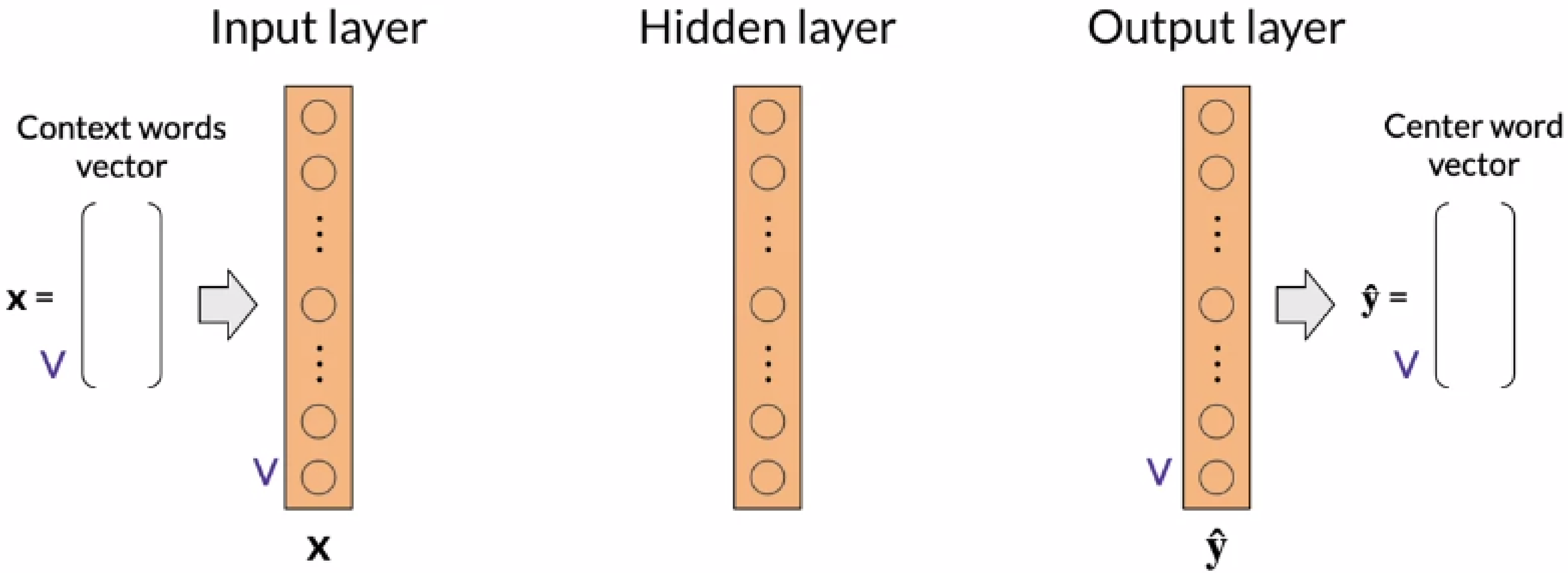
Architecture of the CBOW model



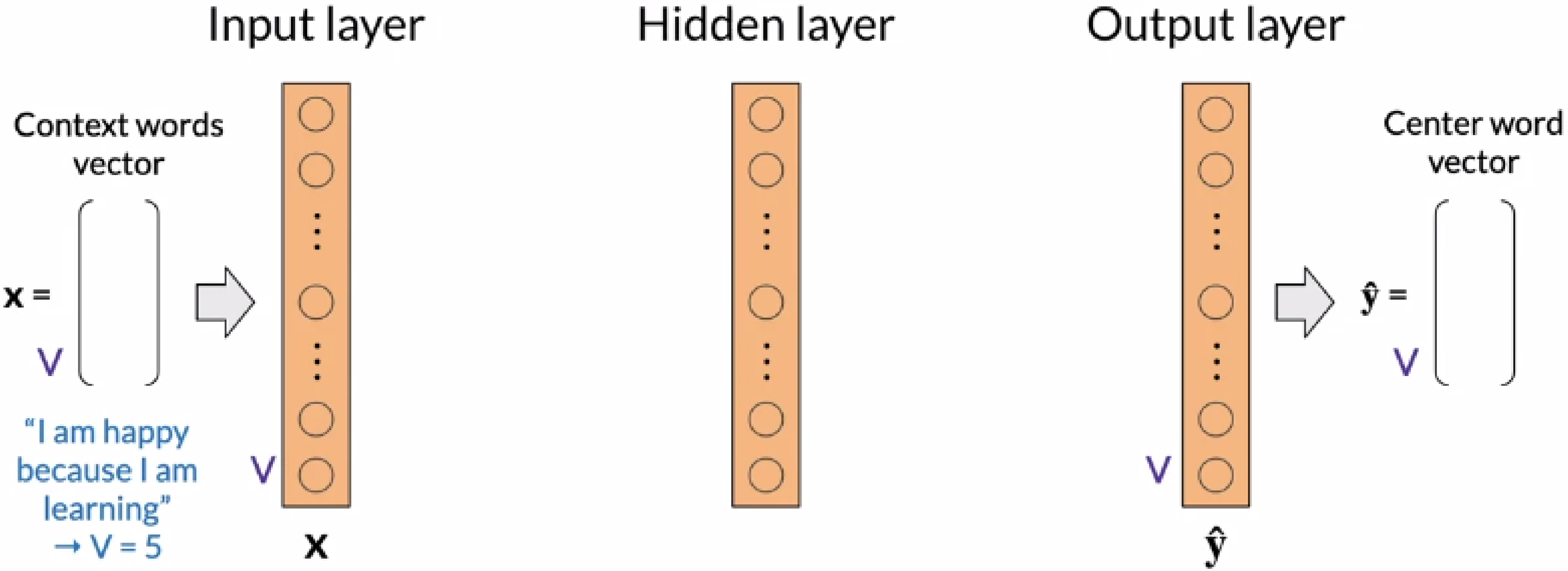
Architecture of the CBOW model



Architecture of the CBOW model



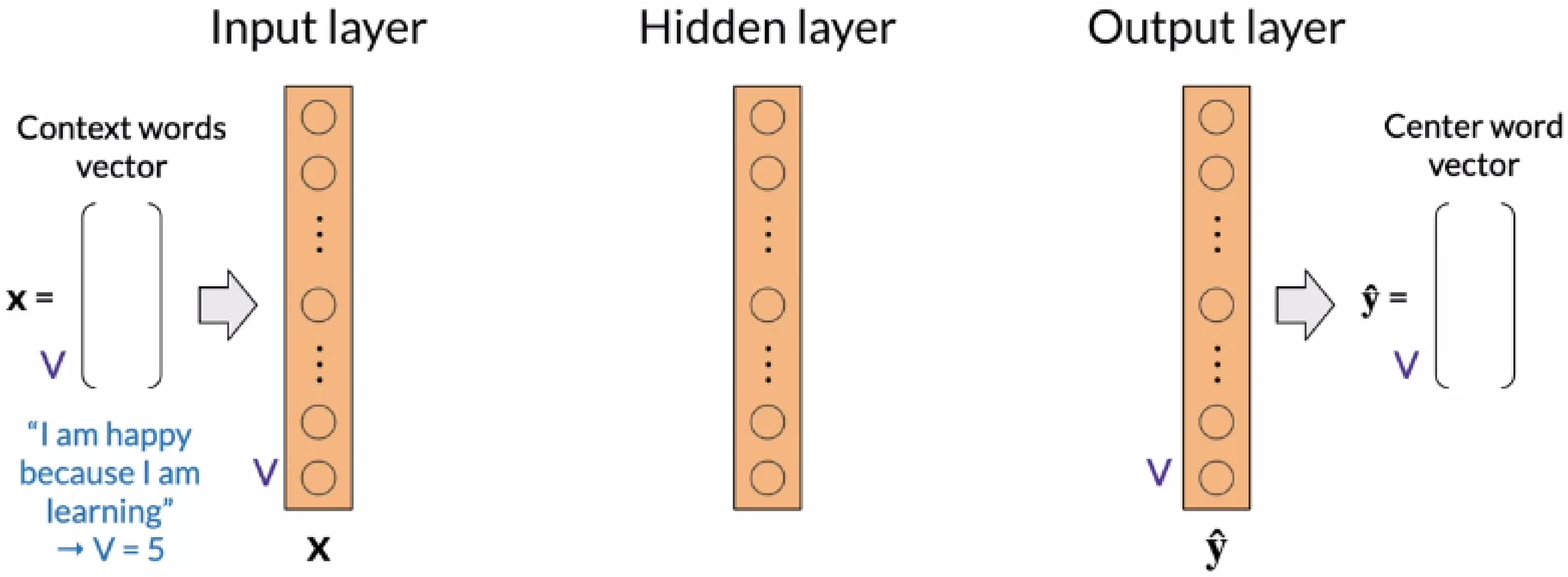
Architecture of the CBOW model



Architecture of the CBOW model

Hyperparameters

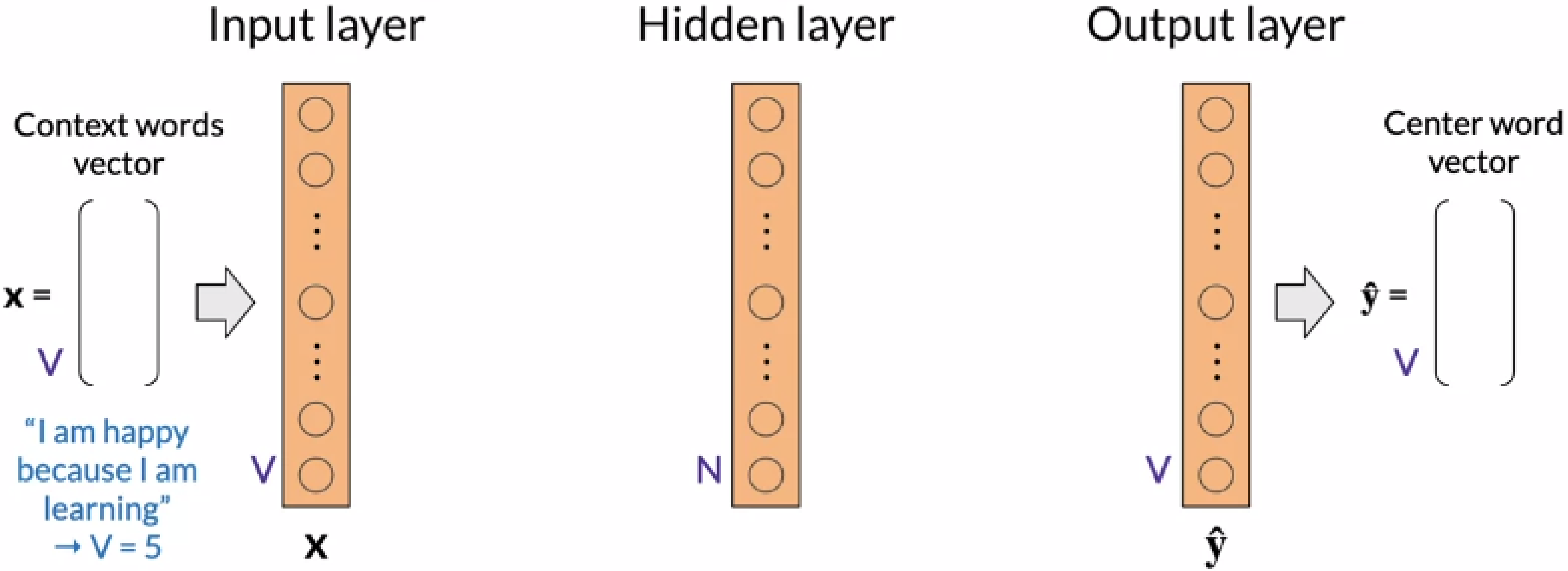
Word embedding size ...



Architecture of the CBOW model

Hyperparameters

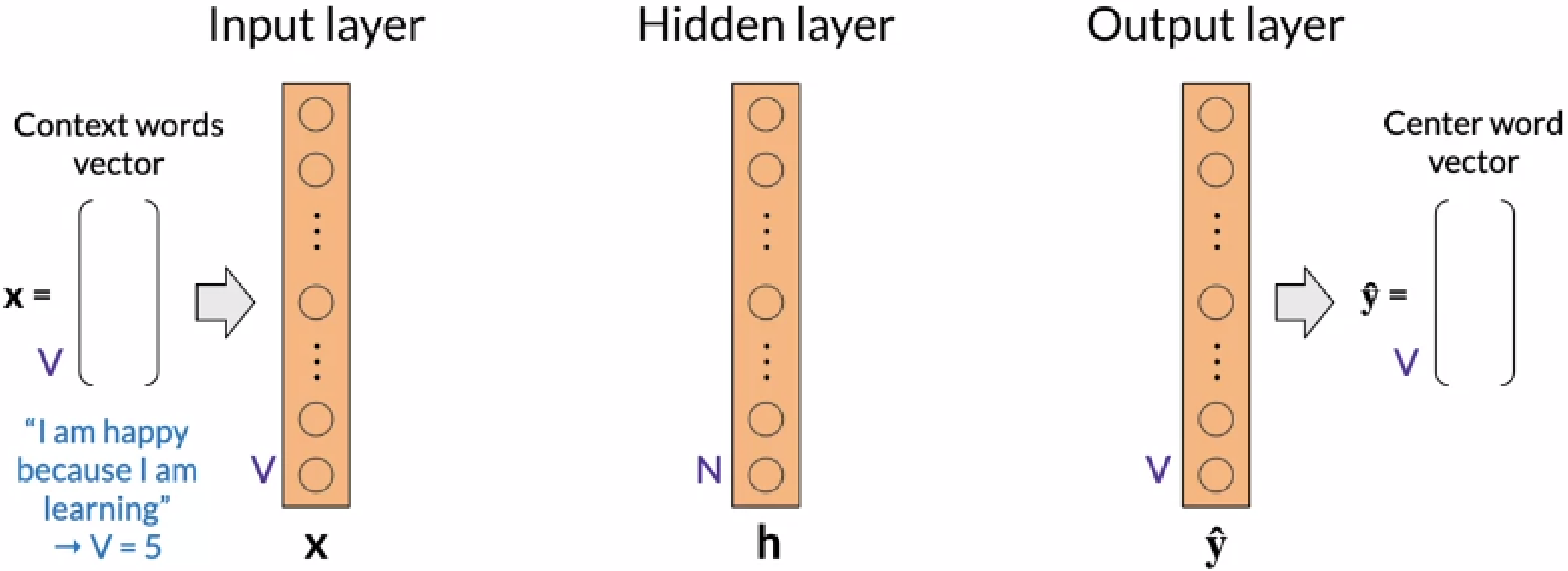
N: Word embedding size ...



Architecture of the CBOW model

Hyperparameters

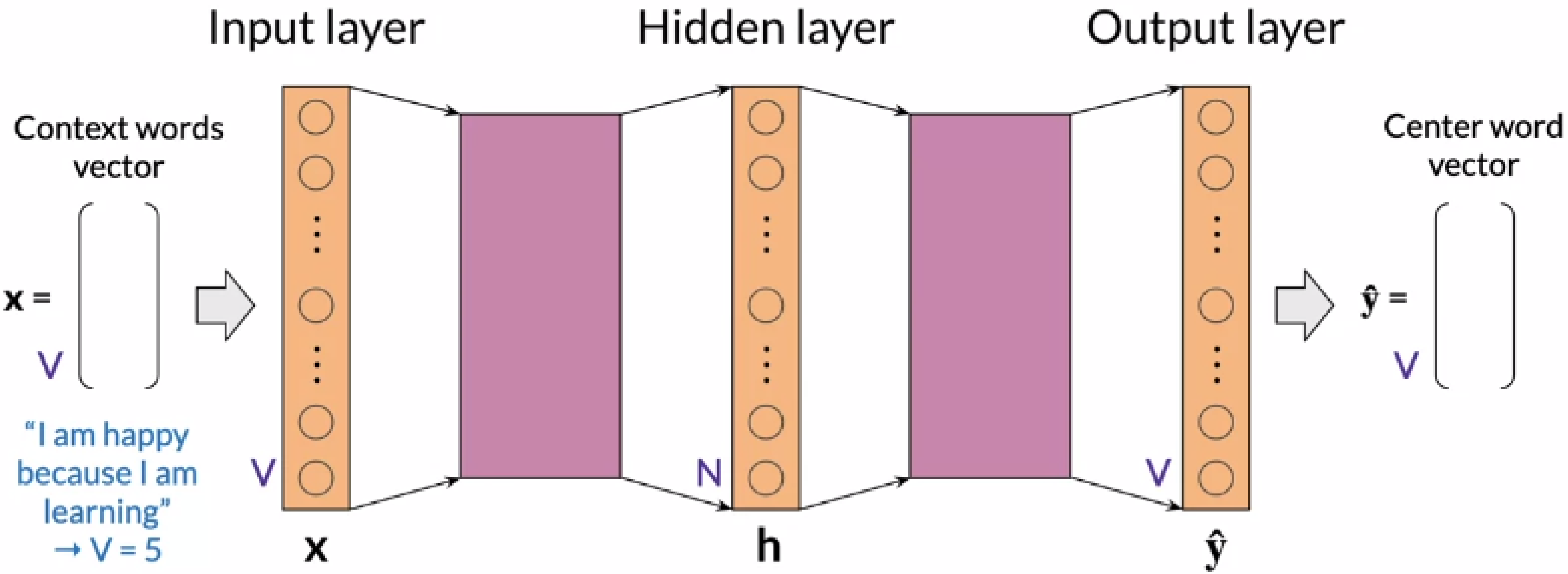
N: Word embedding size ...



Architecture of the CBOW model

Hyperparameters

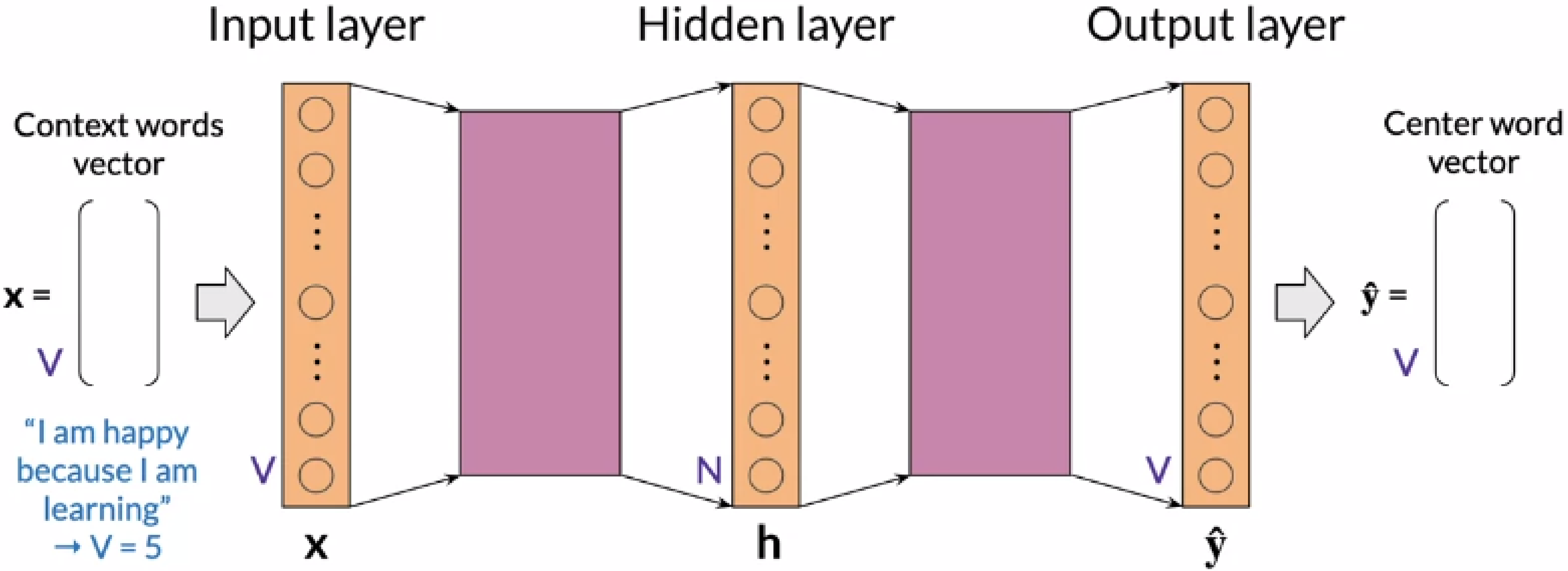
N: Word embedding size ...



Architecture of the CBOW model

Hyperparameters

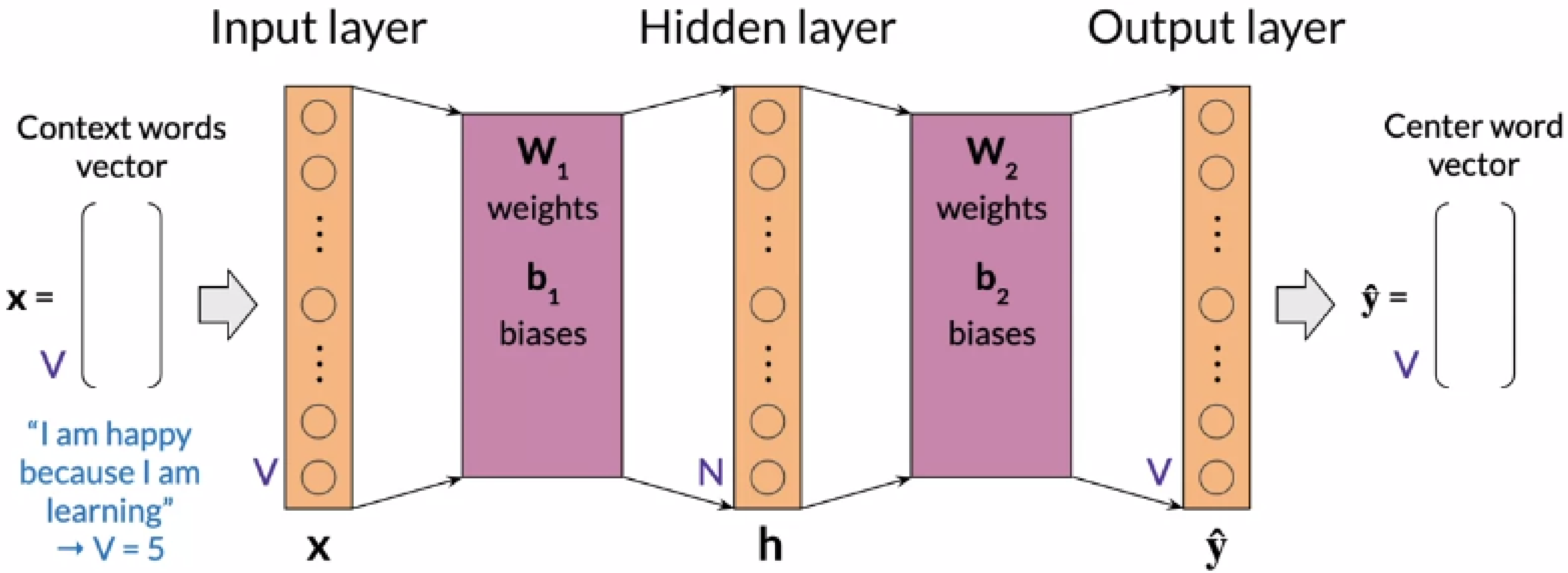
N: Word embedding size ...



Architecture of the CBOW model

Hyperparameters

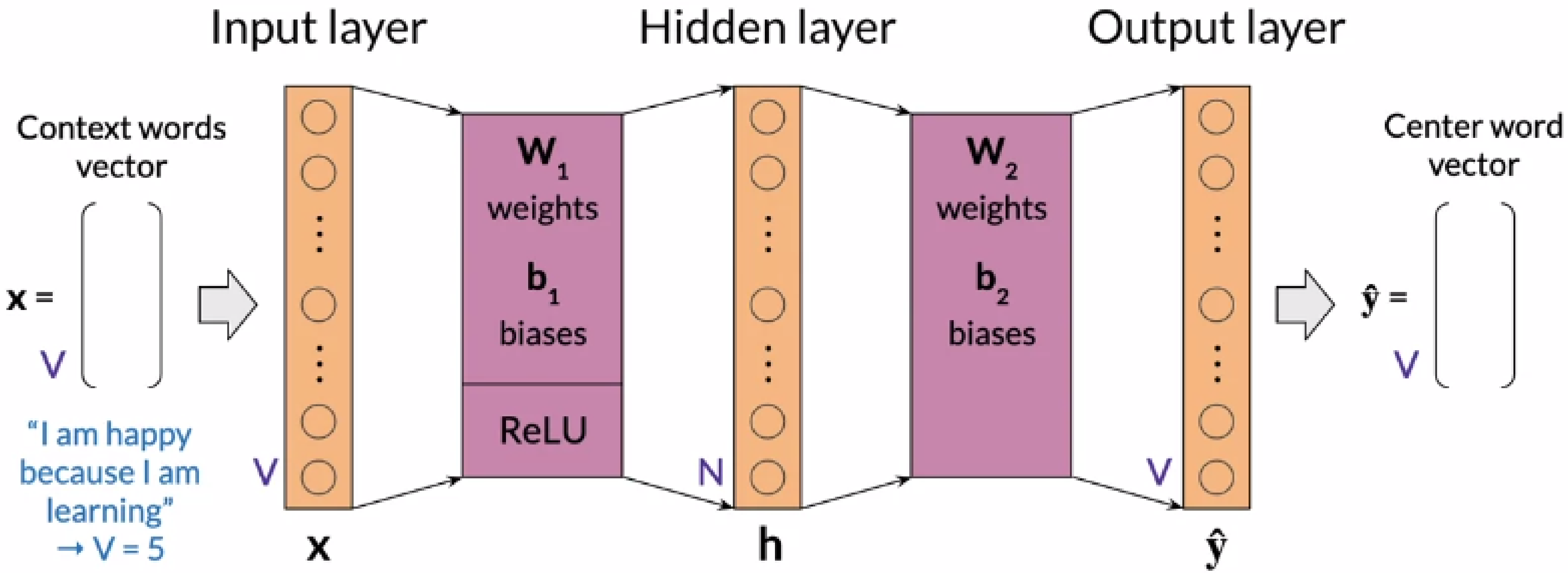
N: Word embedding size ...



Architecture of the CBOW model

Hyperparameters

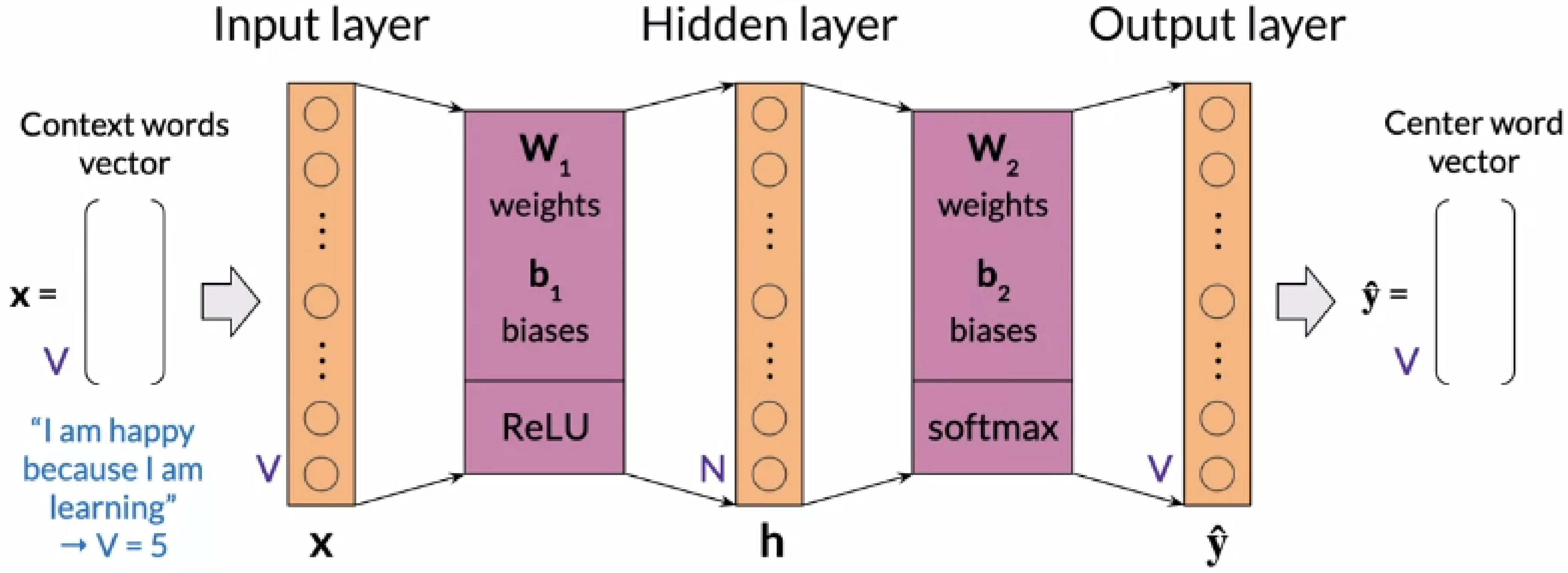
N: Word embedding size ...



Architecture of the CBOW model

Hyperparameters

N: Word embedding size ...

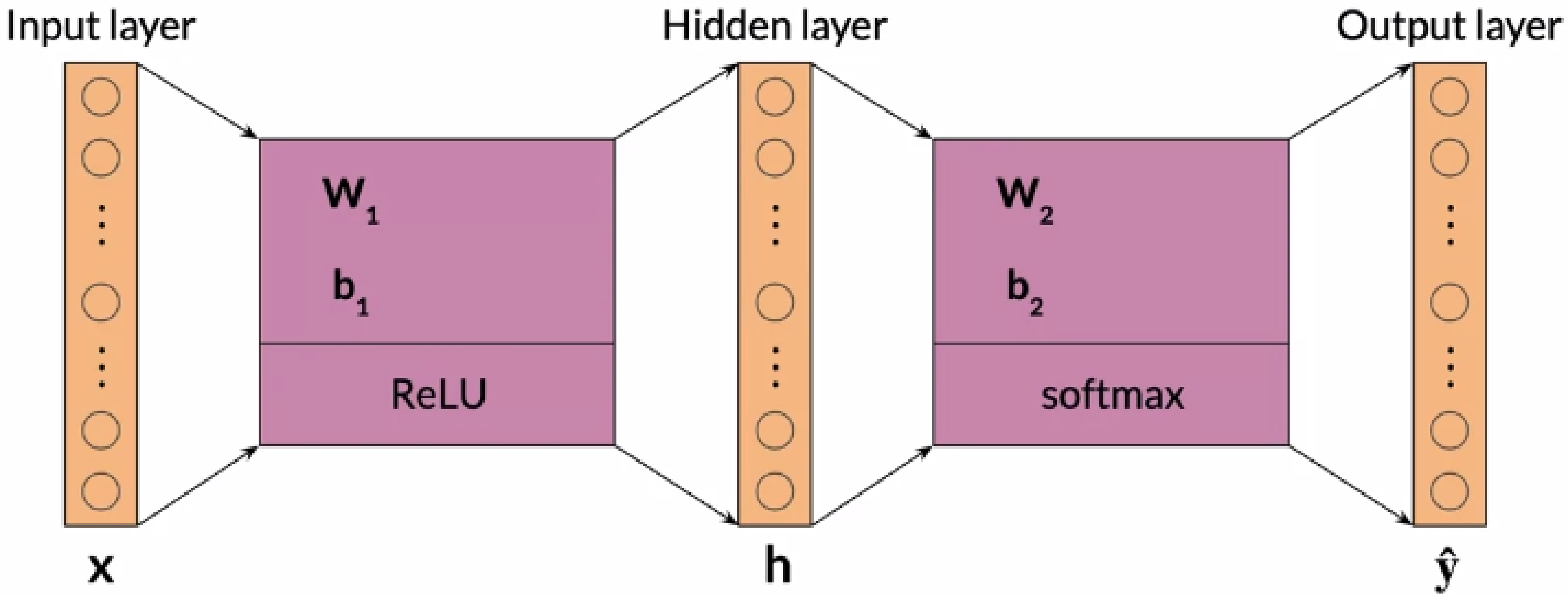


deeplearning.ai

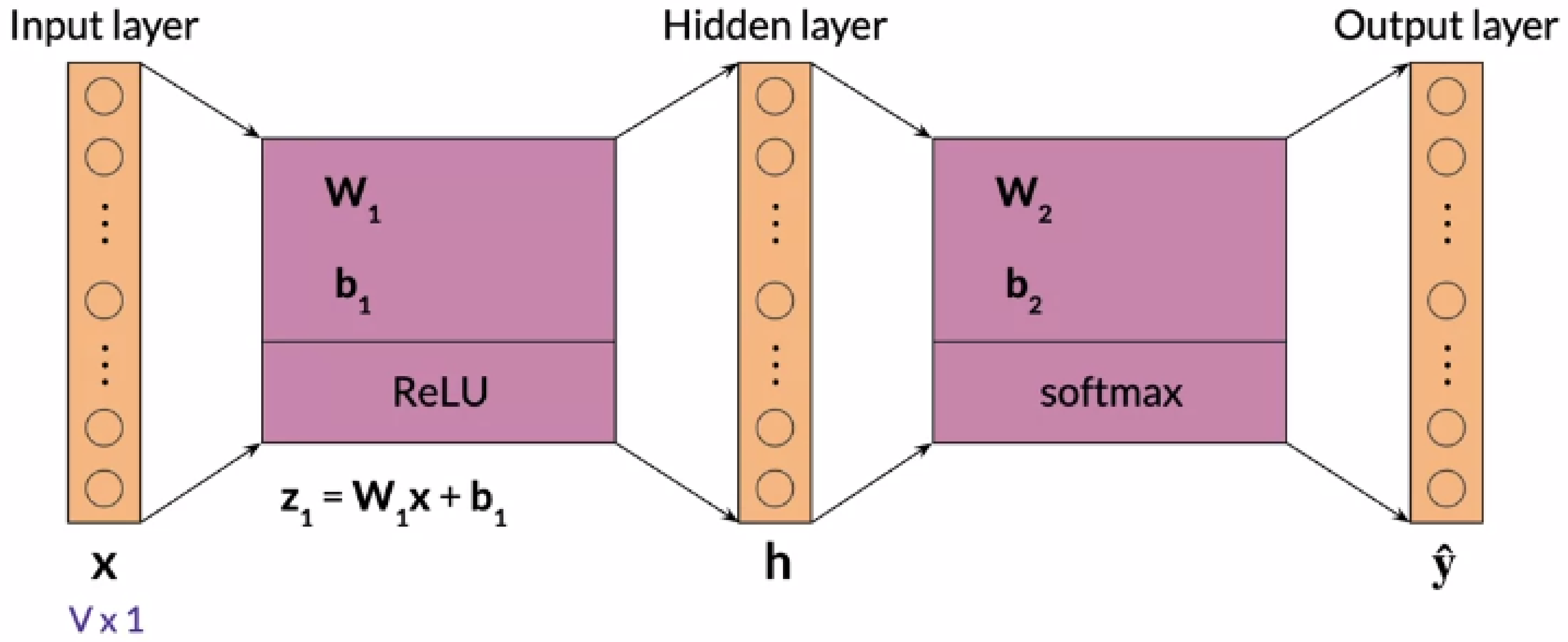
Architecture of the CBOW Model:

Dimensions

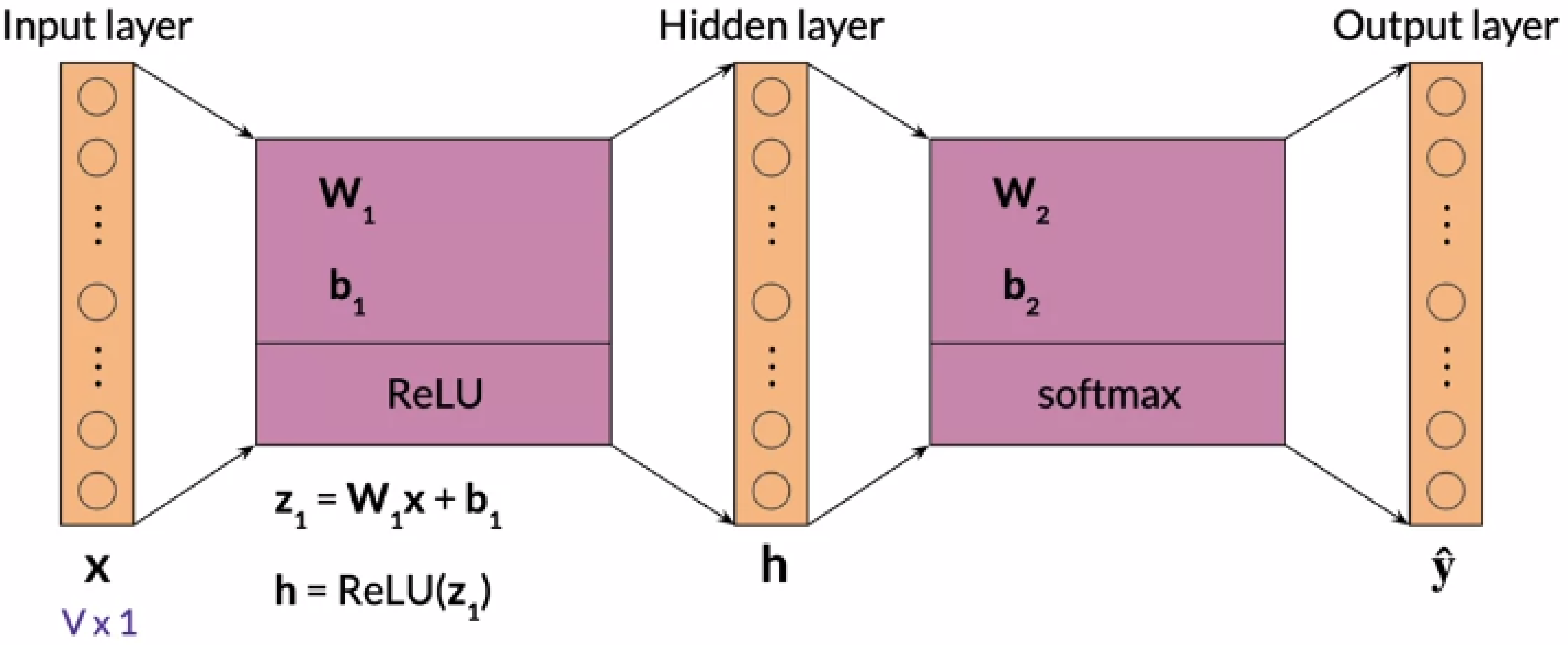
Dimensions (single input)



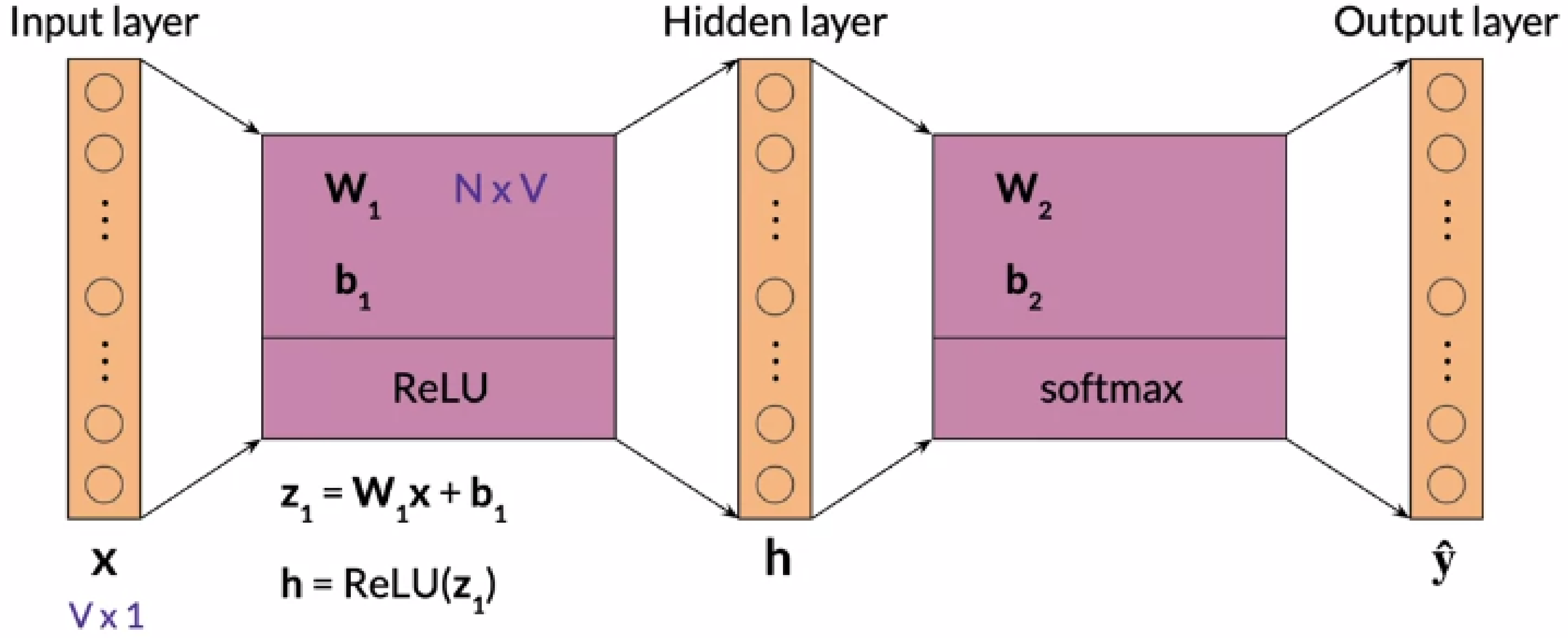
Dimensions (single input)



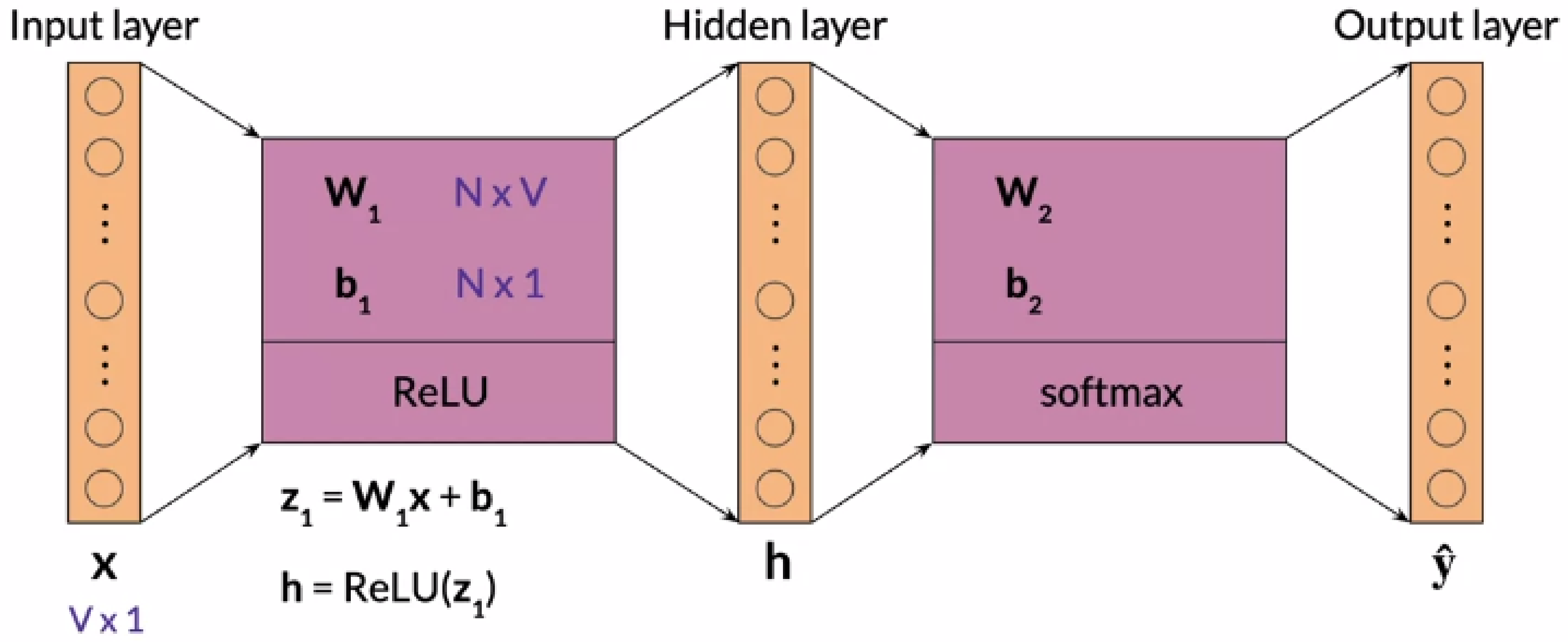
Dimensions (single input)



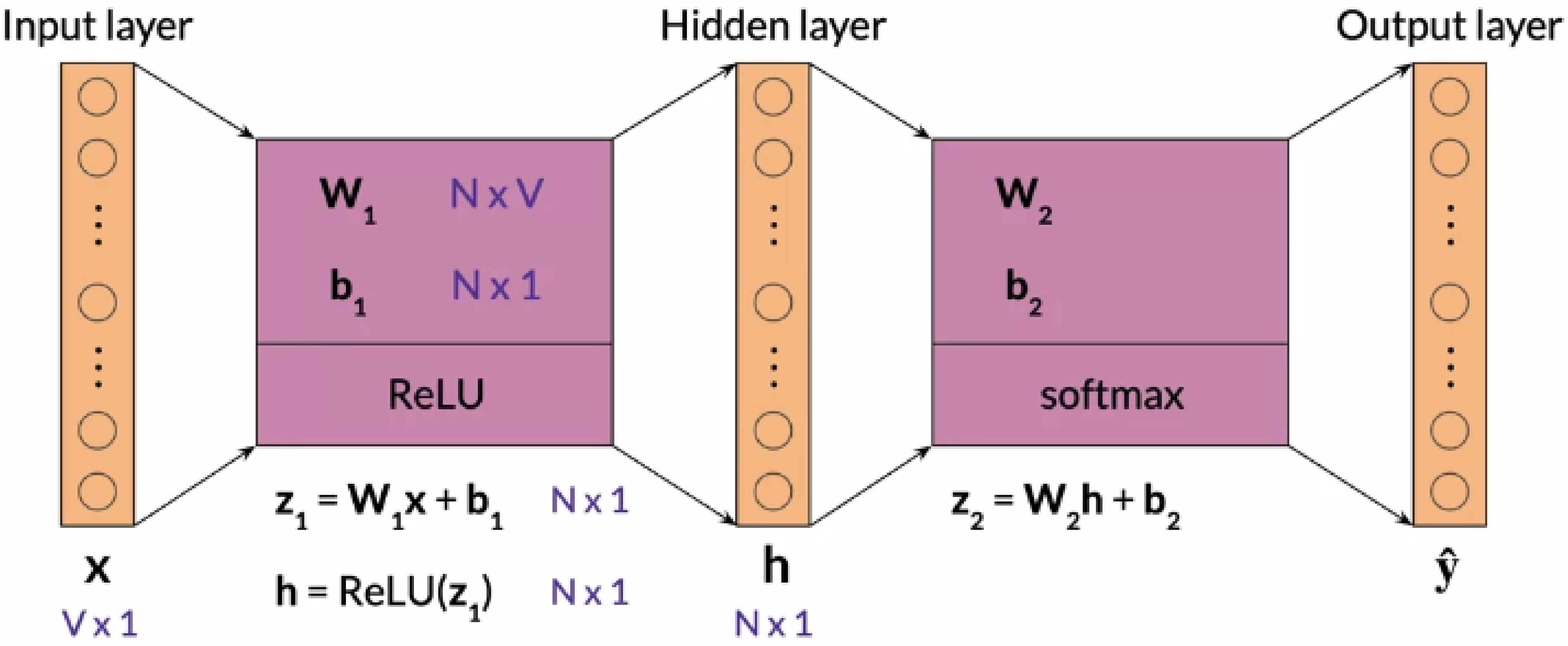
Dimensions (single input)



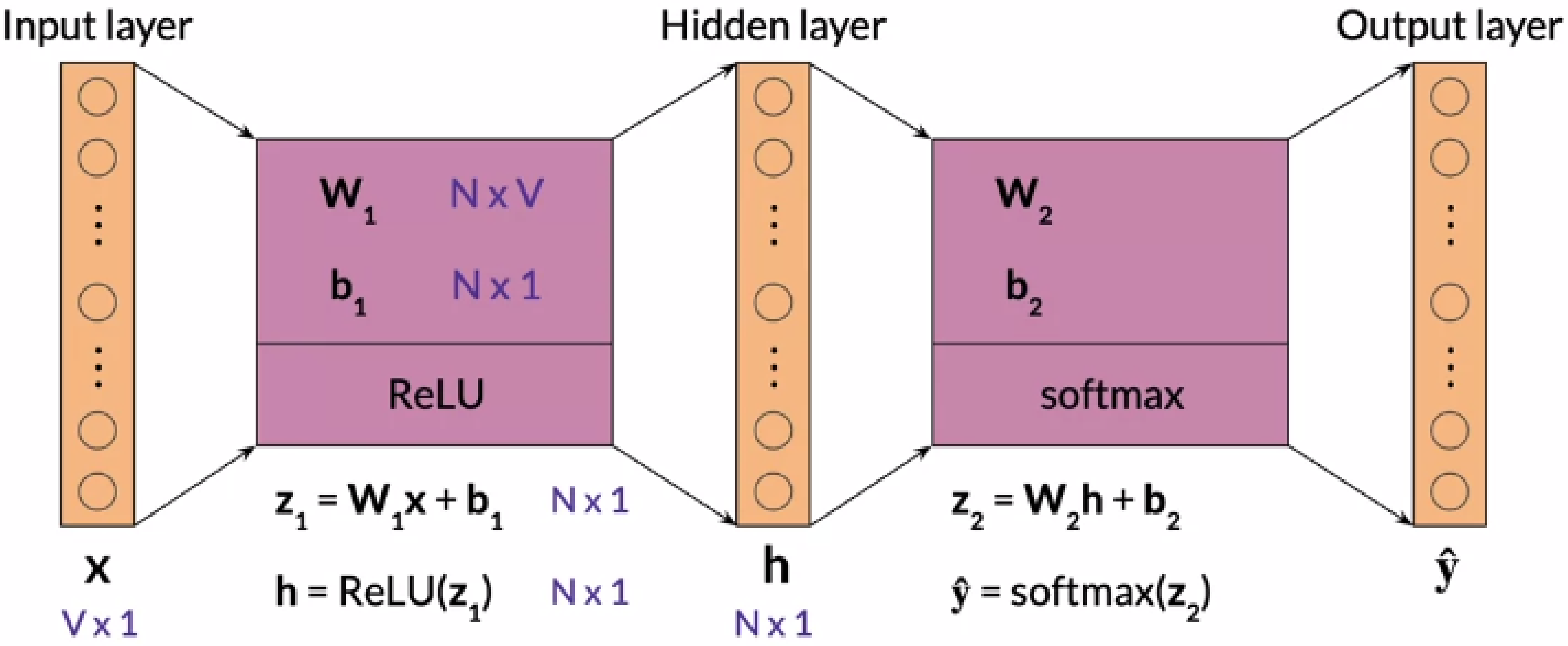
Dimensions (single input)



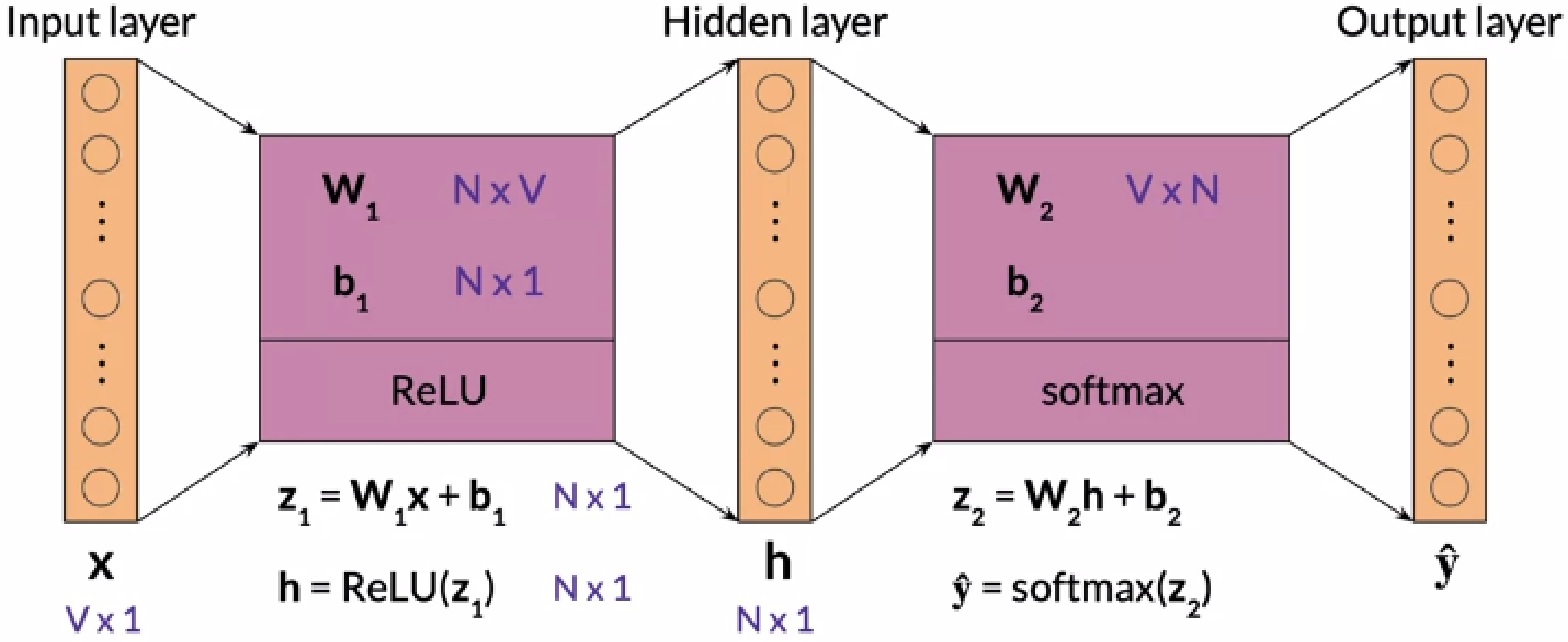
Dimensions (single input)



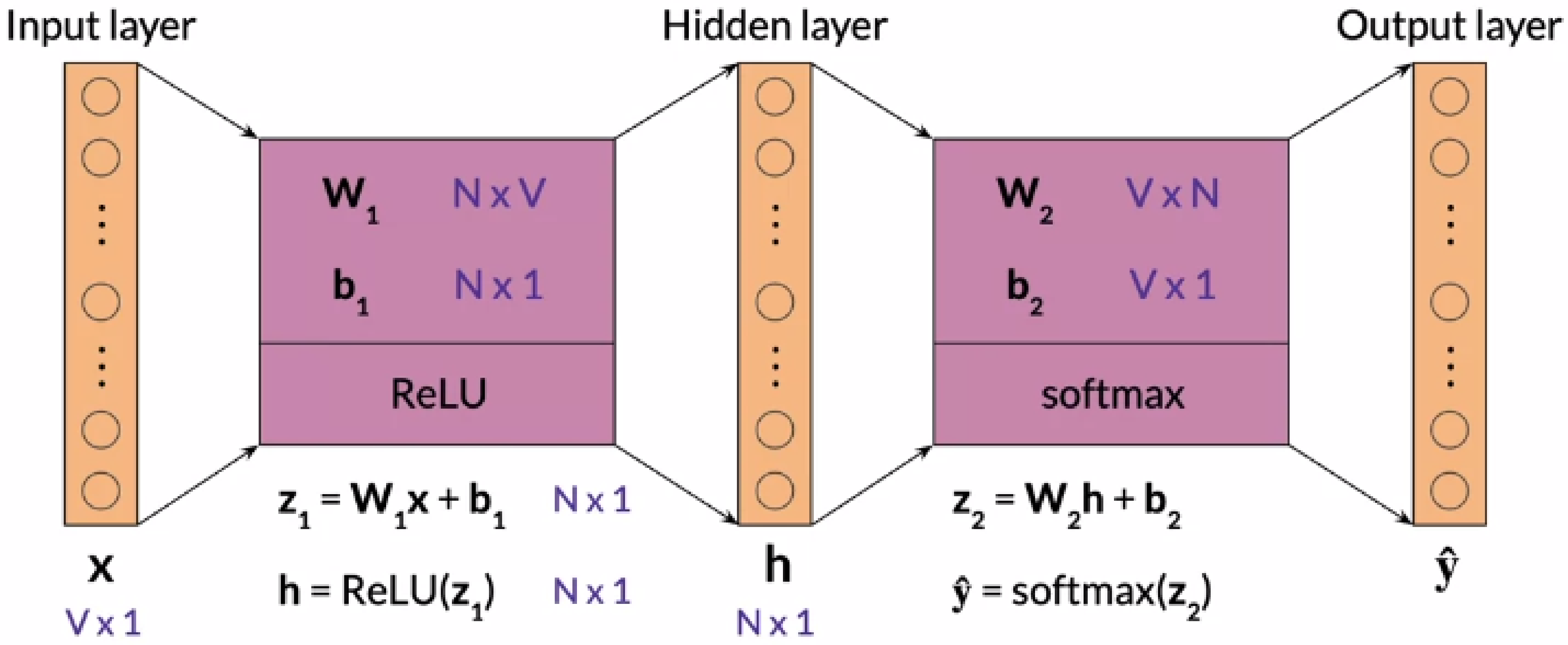
Dimensions (single input)



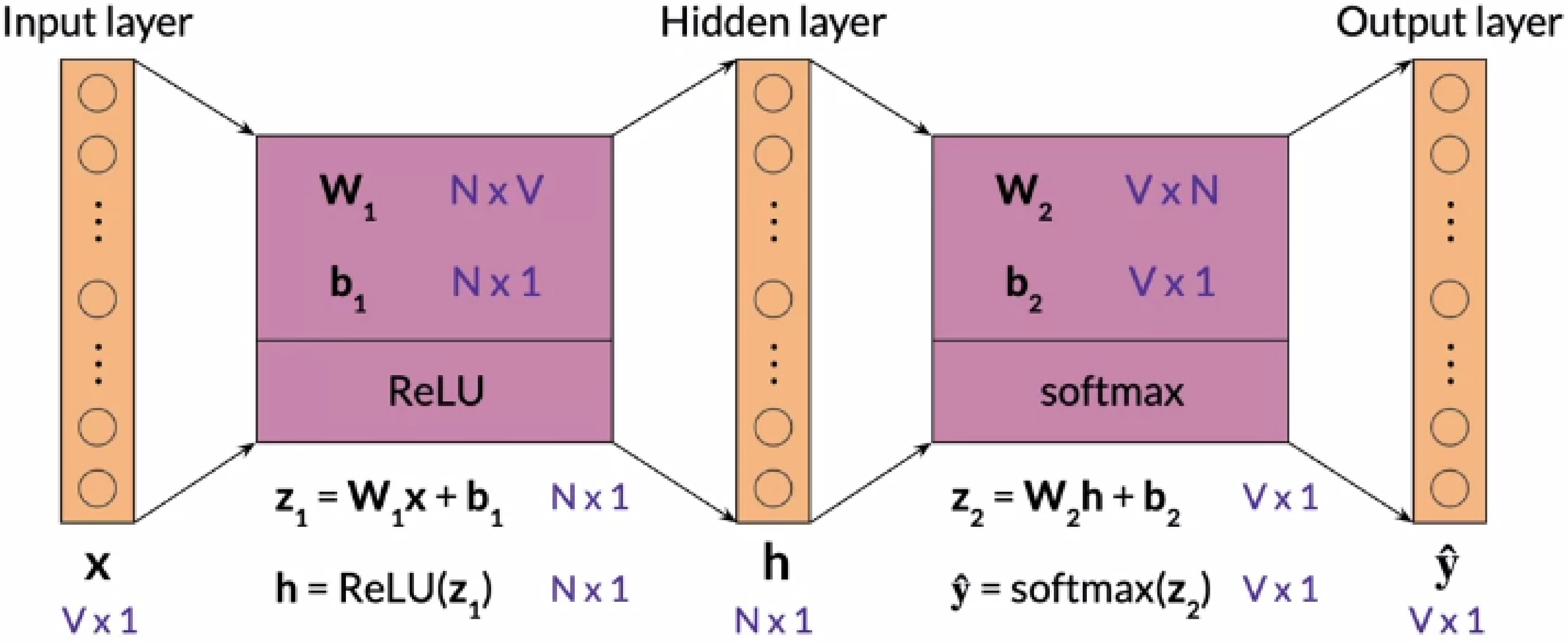
Dimensions (single input)



Dimensions (single input)



Dimensions (single input)



Dimensions (single input)

Column vectors

$$\mathbf{z_1} = \mathbf{W_1}\mathbf{x} + \mathbf{b_1}$$

$$\underset{N \times 1}{\mathbf{z_1} = \begin{pmatrix} \\ \\ \end{pmatrix}} \quad \mathbf{W_1} = \begin{pmatrix} & N \times V & \end{pmatrix} \quad \underset{V \times 1}{\mathbf{x} = \begin{pmatrix} \\ \\ \\ \end{pmatrix}} \quad \underset{N \times 1}{\mathbf{b_1} = \begin{pmatrix} \\ \\ \end{pmatrix}}$$

Dimensions (single input)

Column vectors

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

$$\mathbf{z}_1 = \underset{N \times 1}{\begin{pmatrix} \\ \\ \end{pmatrix}} \quad \mathbf{W}_1 = \begin{pmatrix} \quad N \times V \quad \end{pmatrix} \quad \mathbf{x} = \underset{V \times 1}{\begin{pmatrix} \\ \\ \\ \end{pmatrix}} \quad \mathbf{b}_1 = \underset{N \times 1}{\begin{pmatrix} \\ \\ \end{pmatrix}}$$

Dimensions (single input)

Column vectors

$$\mathbf{z_1} = \mathbf{W_1}\mathbf{x} + \mathbf{b_1}$$

$$\mathbf{z_1} = \underset{N \times 1}{\begin{pmatrix} \\ \\ \end{pmatrix}} \quad \mathbf{W_1} = \begin{pmatrix} & N \times V & \end{pmatrix} \quad \mathbf{x} = \underset{V \times 1}{\begin{pmatrix} \\ \\ \\ \end{pmatrix}} \quad \mathbf{b_1} = \underset{N \times 1}{\begin{pmatrix} \\ \\ \end{pmatrix}}$$

Row vectors

$$\mathbf{z_1} = \mathbf{x}\mathbf{W_1}^T + \mathbf{b_1}$$

$$\mathbf{b_1} = \begin{pmatrix} 1 \times N \end{pmatrix} \quad \mathbf{W_1} = \begin{pmatrix} & N \times V & \end{pmatrix} \quad \mathbf{b_1} = \begin{pmatrix} 1 \times N \end{pmatrix}$$

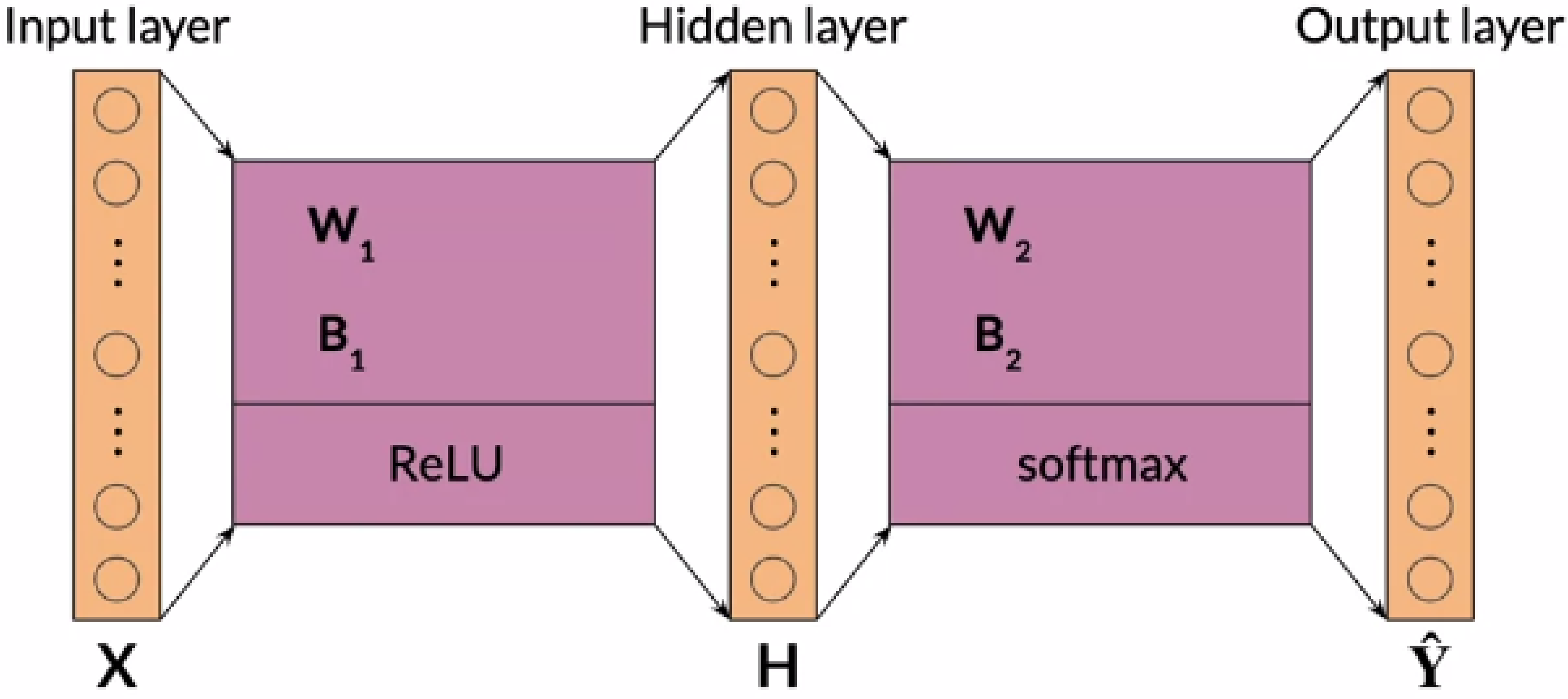
$$\mathbf{x} = \begin{pmatrix} & 1 \times V & \end{pmatrix}$$

Architecture of the CBOW Model:

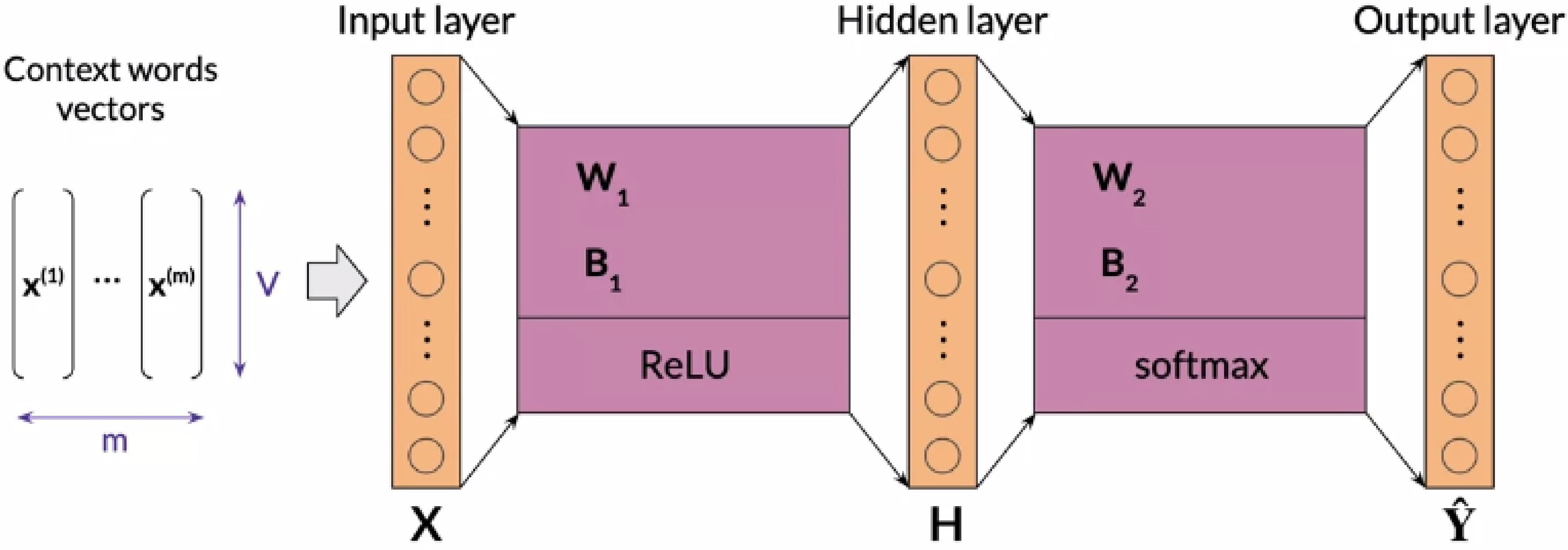
Dimensions 2

deeplearning.ai

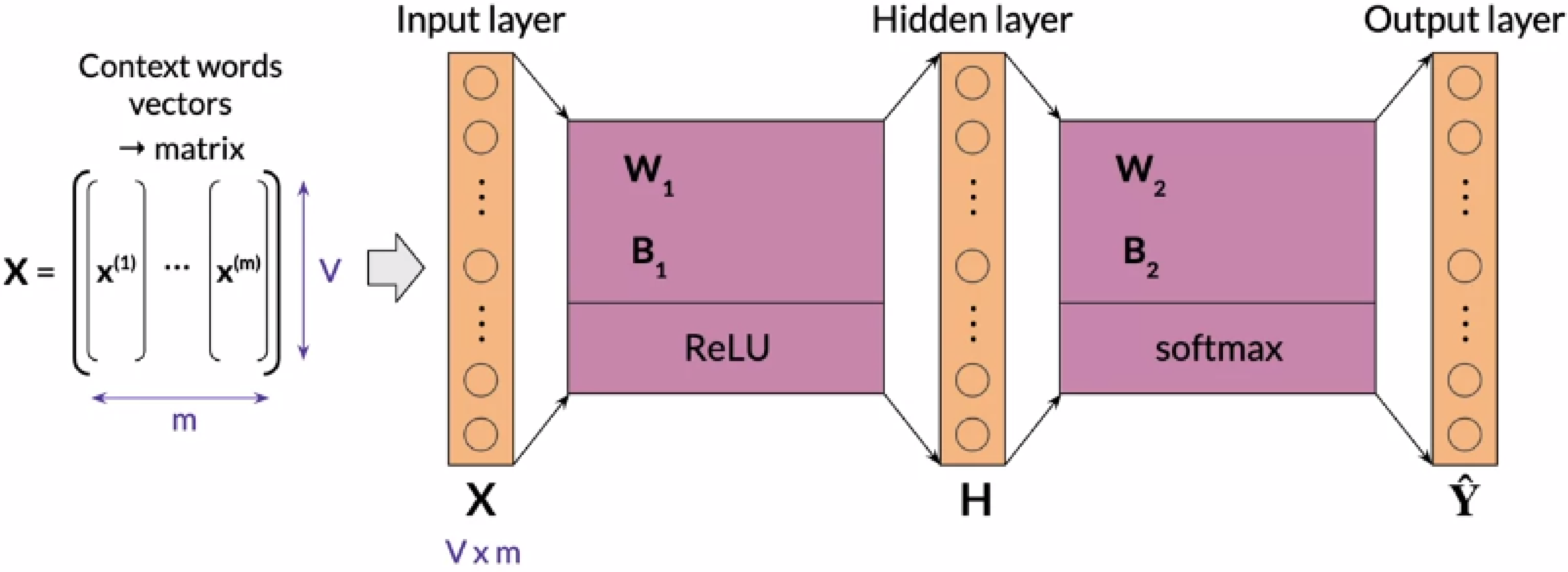
Dimensions (batch input)



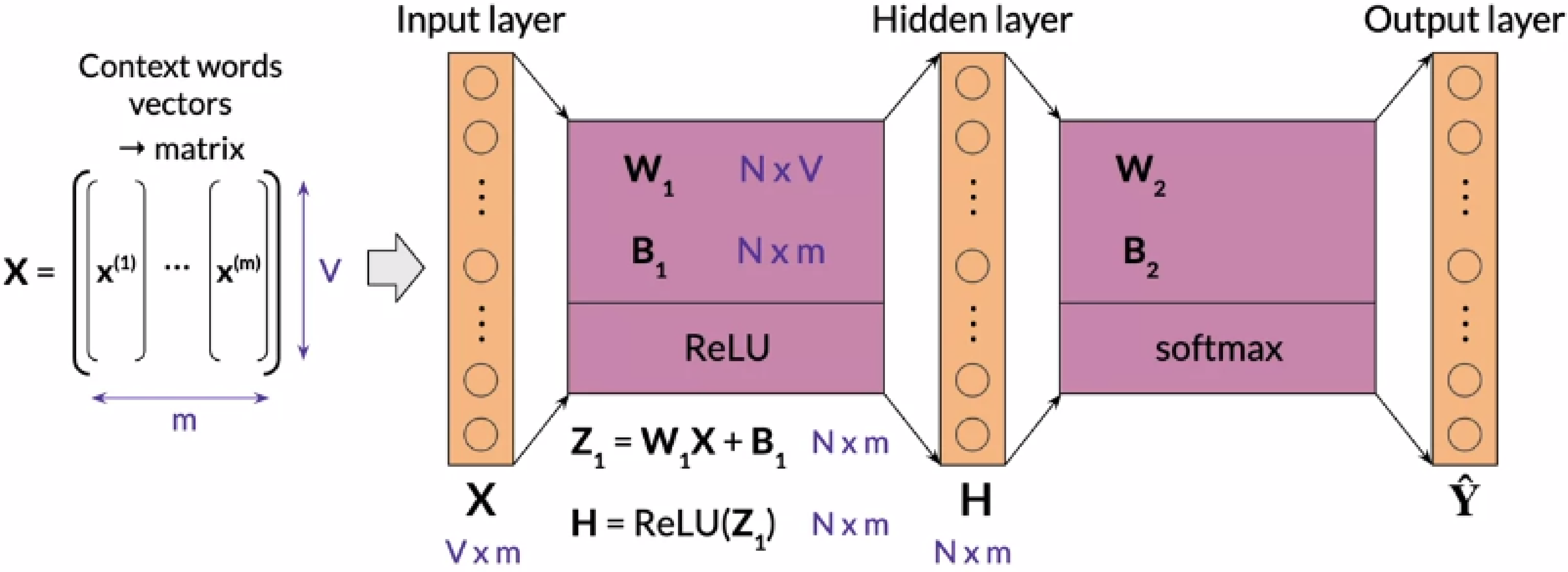
Dimensions (batch input)



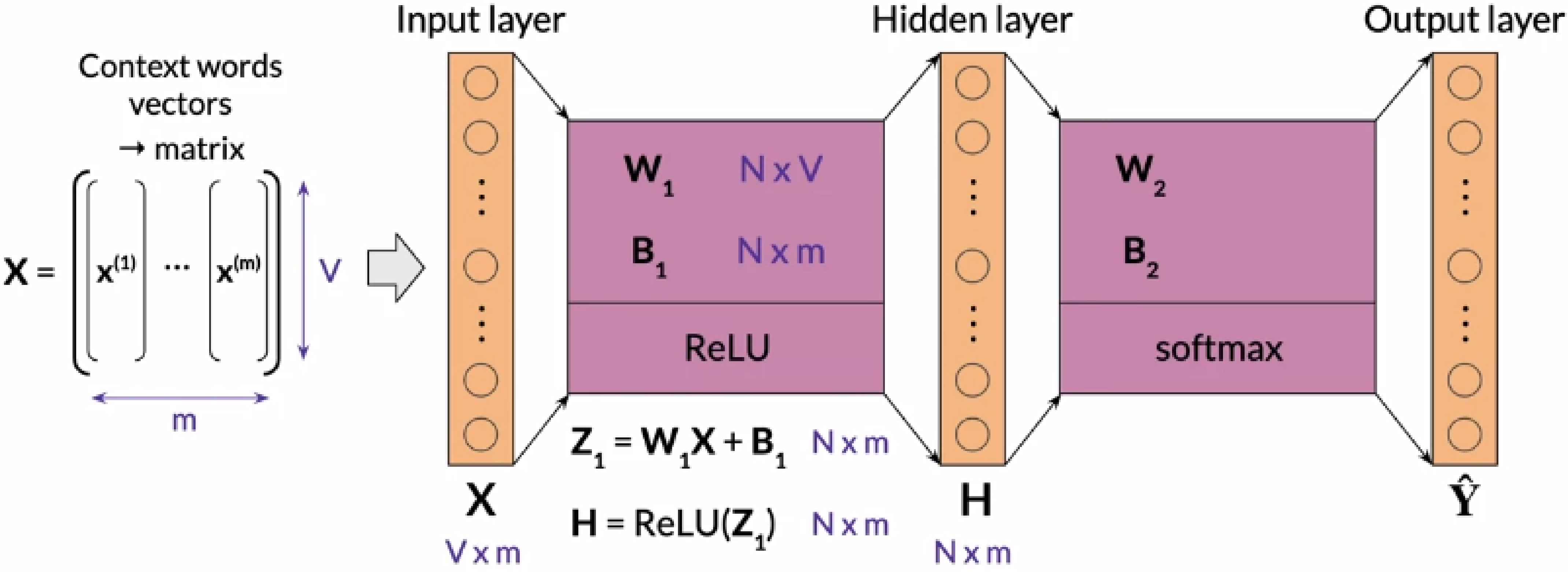
Dimensions (batch input)



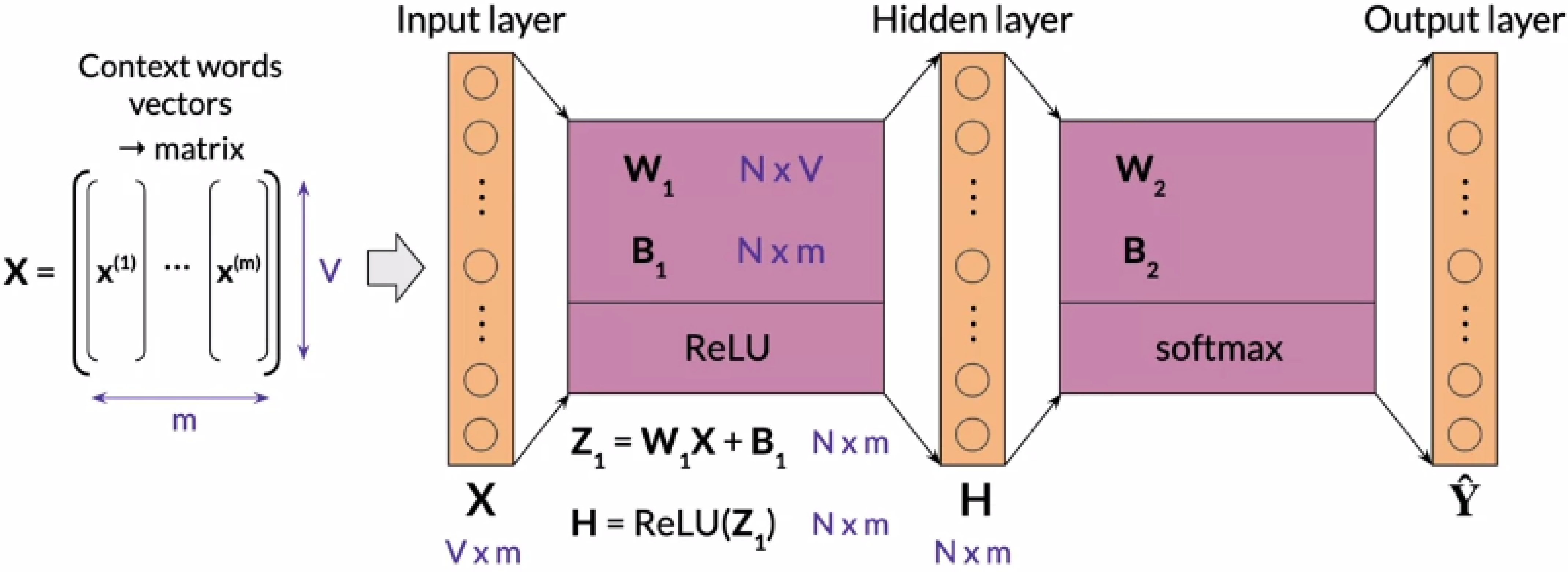
Dimensions (batch input)



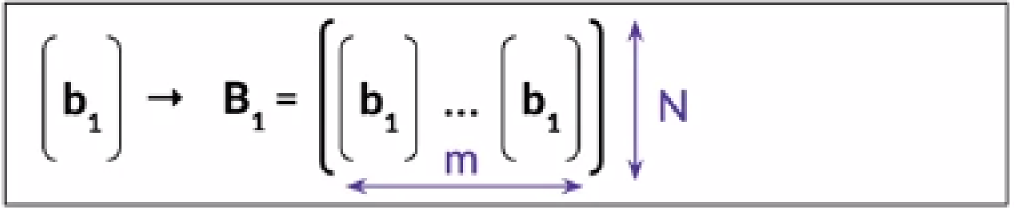
Dimensions (batch input)

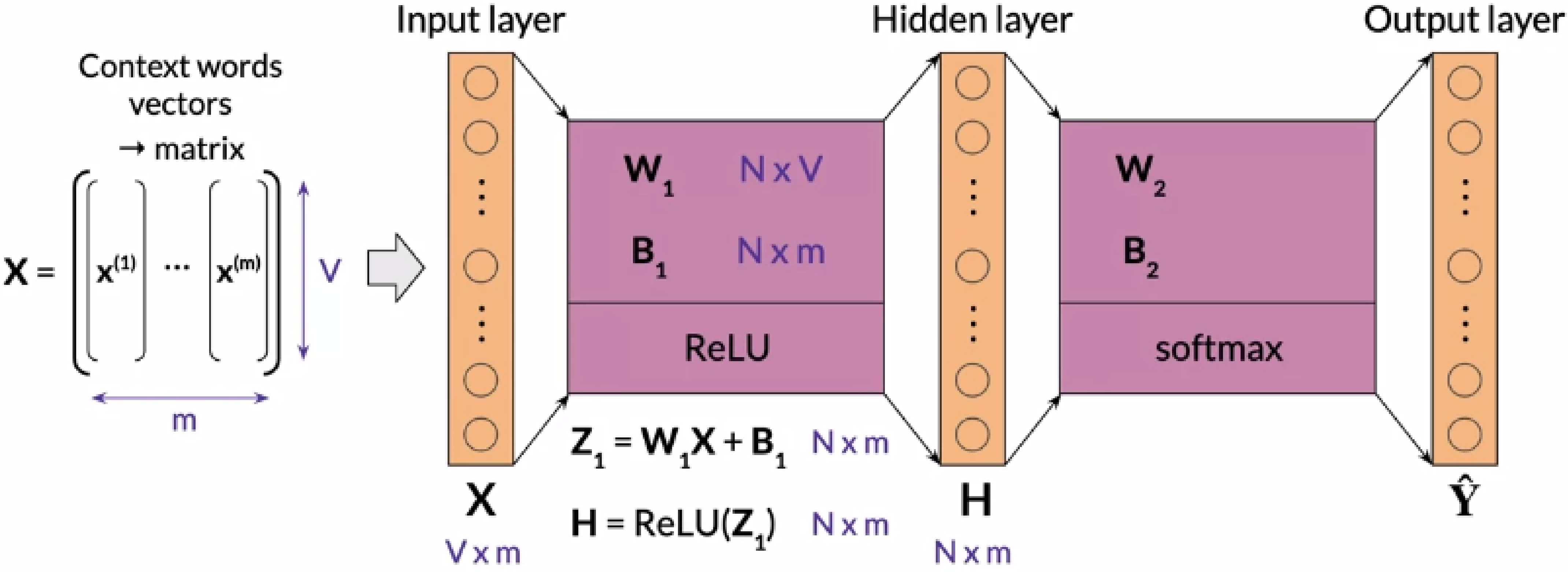


Dimensions (batch input)



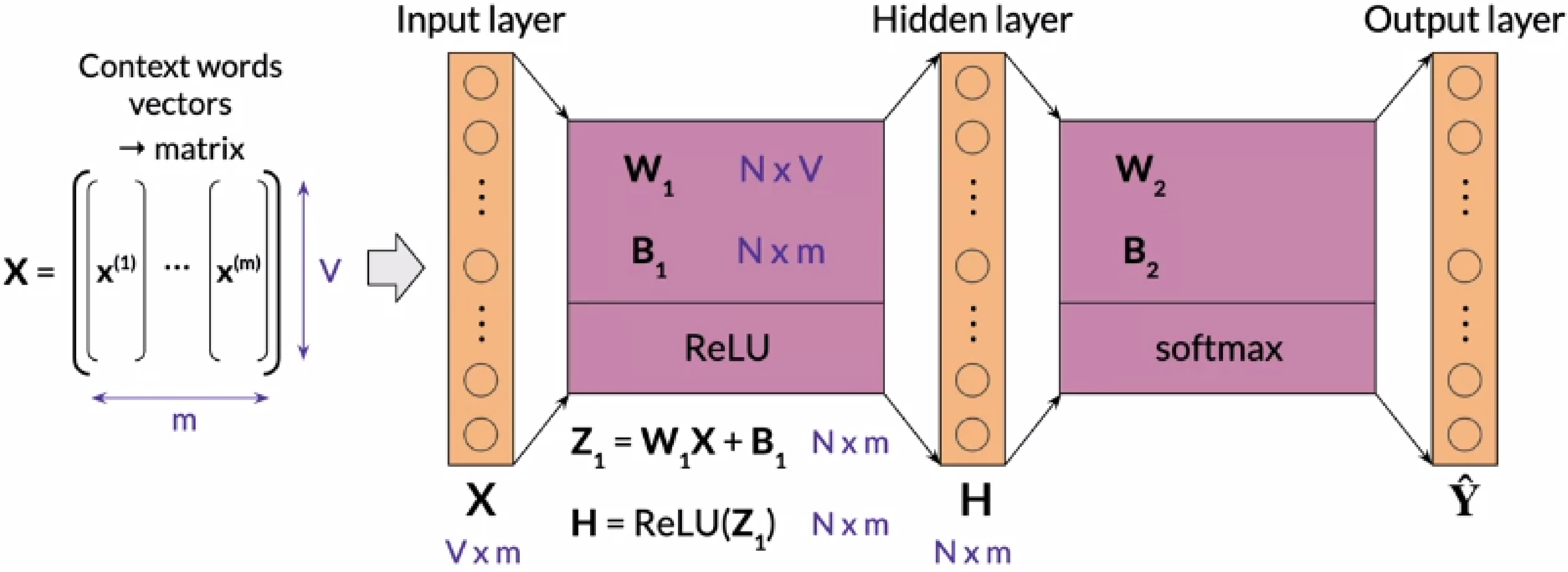
Dimensions (batch input)



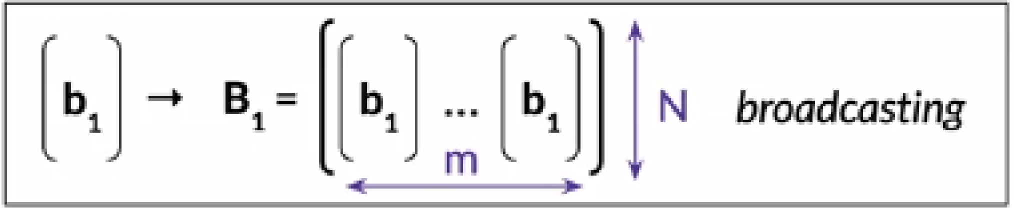


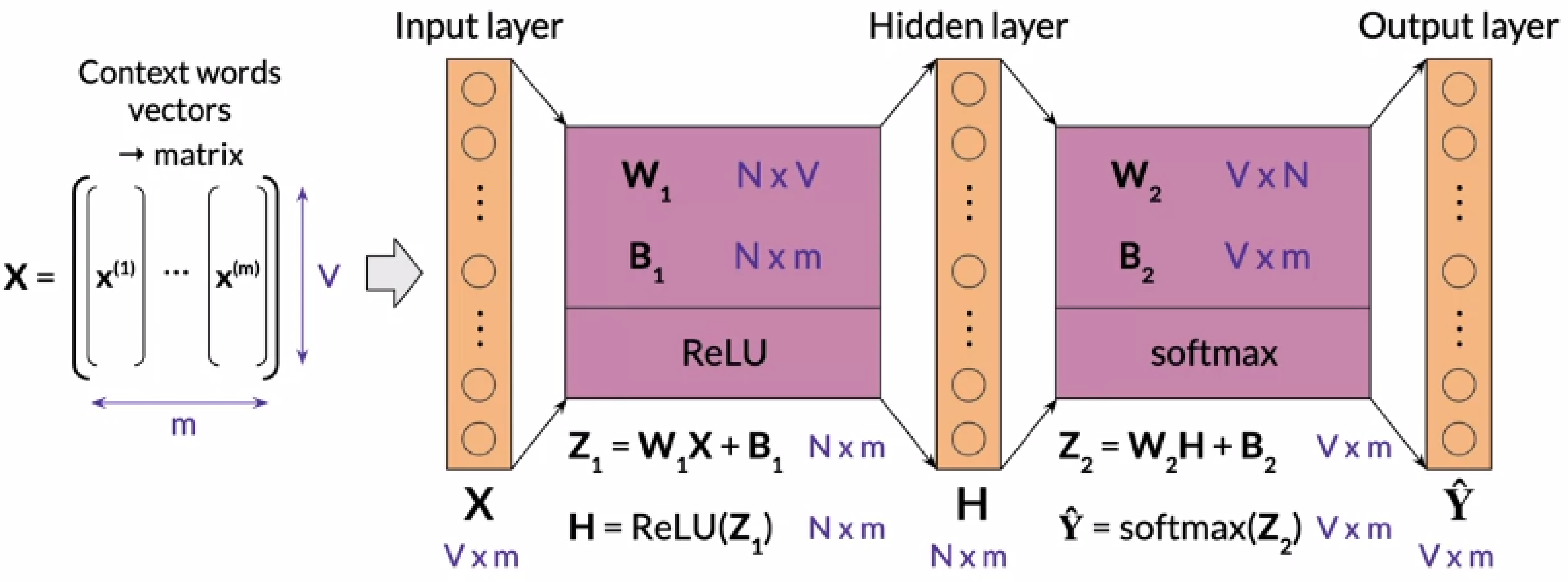
Dimensions (batch input)

$$\begin{bmatrix} b_1 \end{bmatrix} \rightarrow \quad B_1 = \underbrace{\begin{bmatrix} \begin{bmatrix} b_1 \end{bmatrix} & \dots & \begin{bmatrix} b_1 \end{bmatrix} \end{bmatrix}}_{m} \updownarrow N \quad \textit{broadcasting}$$

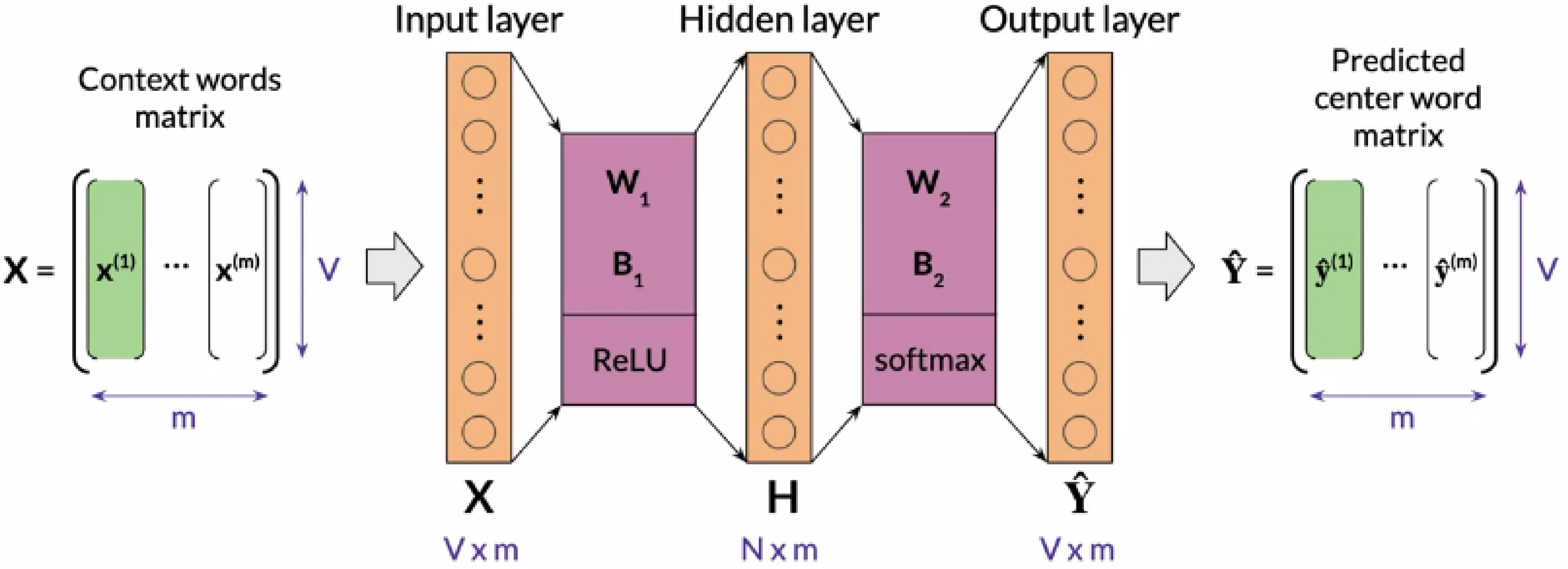


Dimensions (batch input)

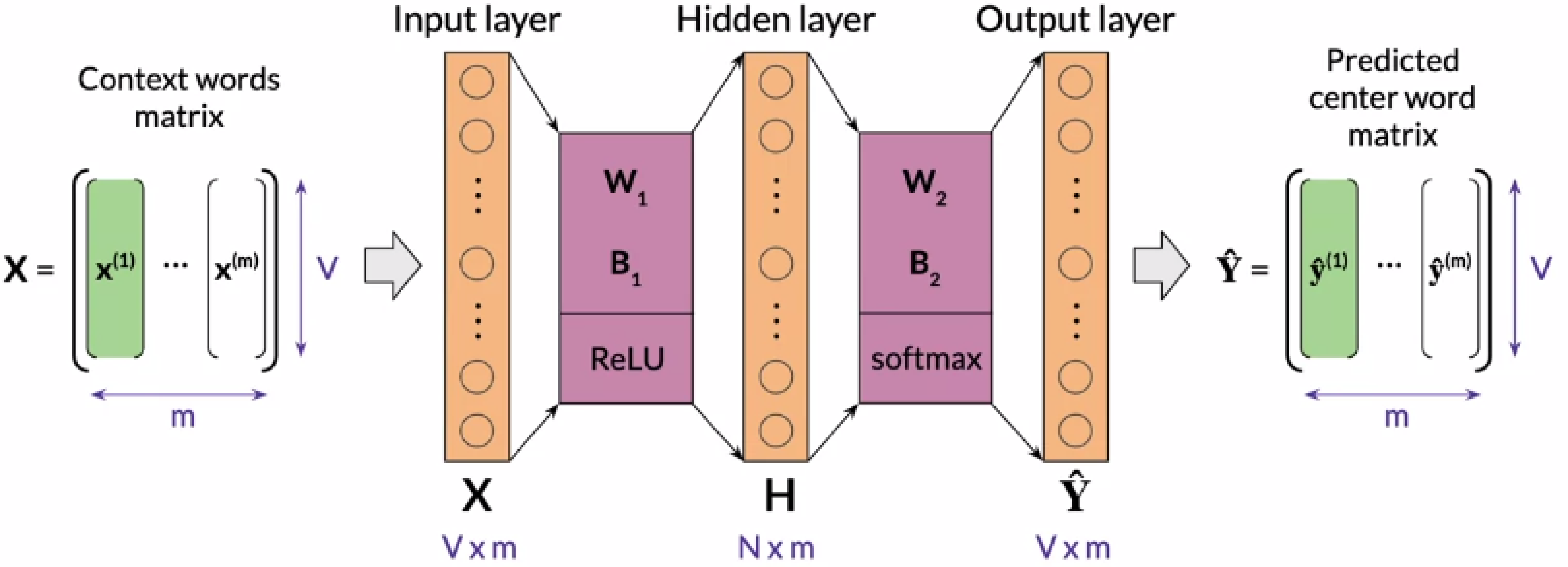




Dimensions (batch input)

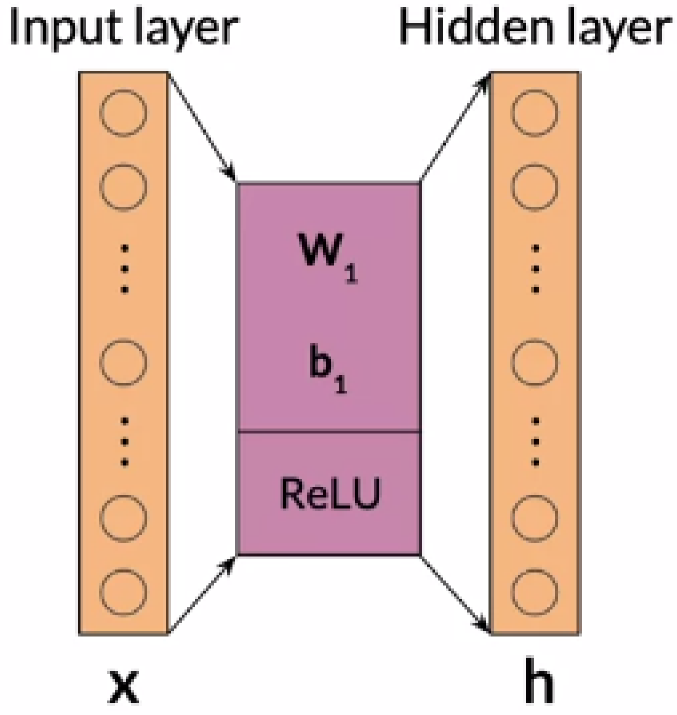


Dimensions (batch input)



Rectified Linear Unit (ReLU)

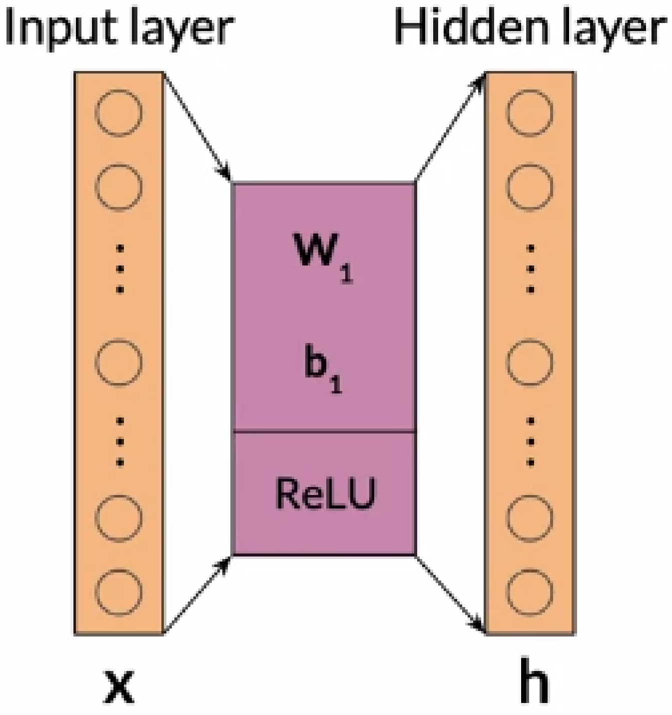
Rectified Linear Unit (ReLU)



$$z_1 = W_1 x + b_1$$

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

Rectified Linear Unit (ReLU)

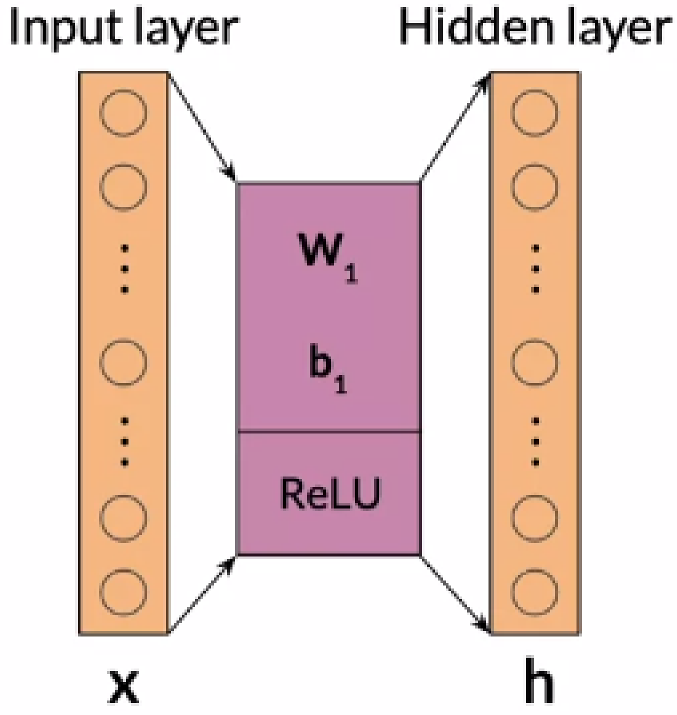


$$z_1 = W_1x + b_1$$

$$h = ReLU(z_1)$$

$$ReLU(x) = max(0, x)$$

Rectified Linear Unit (ReLU)

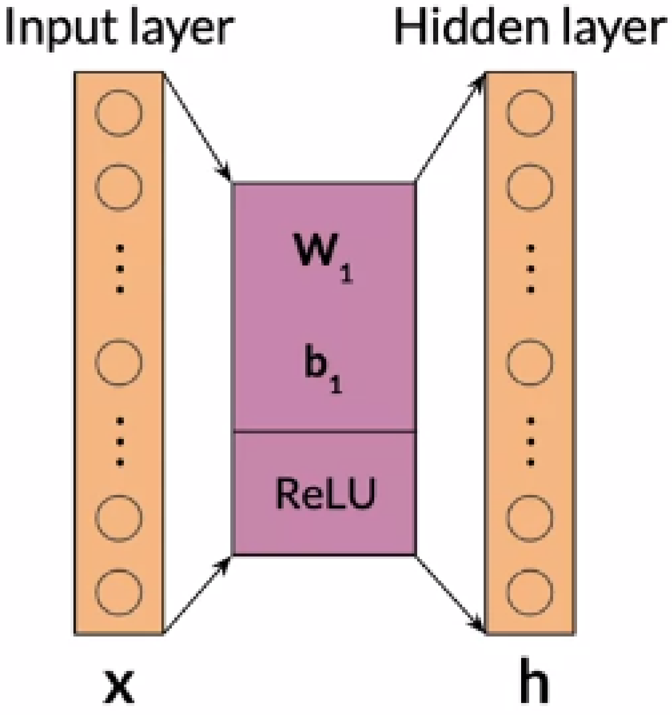


$$\mathbf{z_1} = \mathbf{W_1 x} + \mathbf{b_1}$$

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

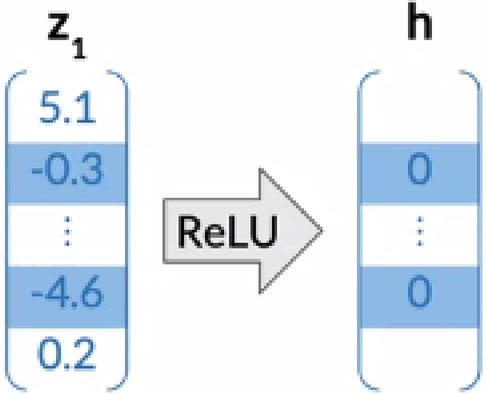
$$\mathrm{ReLU}(x) = \max(0, x)$$

Rectified Linear Unit (ReLU)

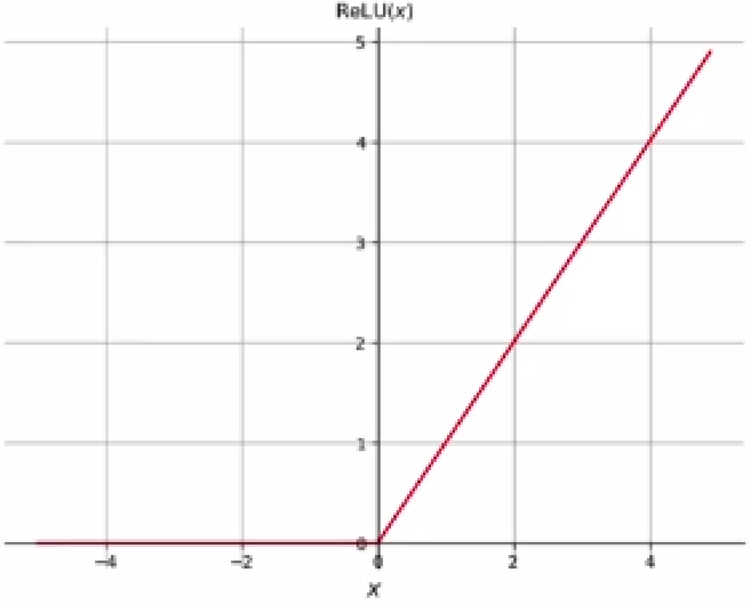


$$z_1 = W_1x + b_1$$

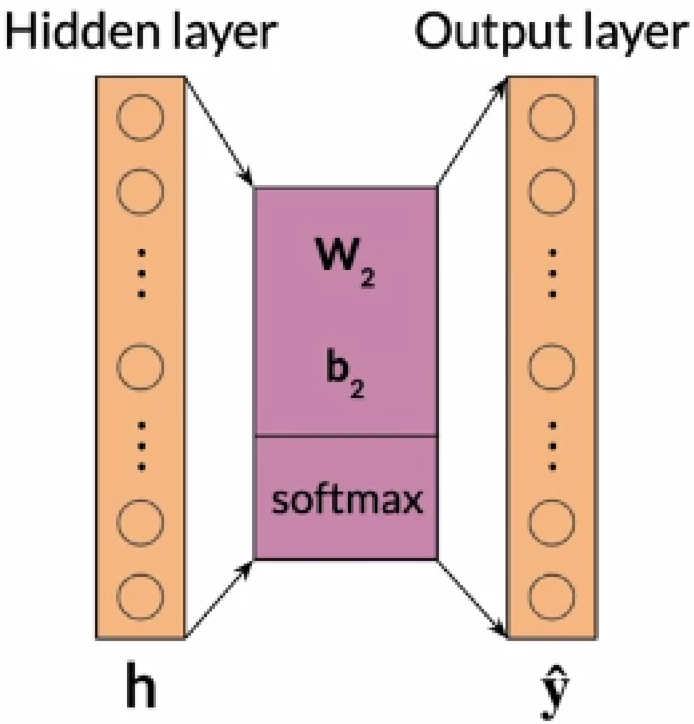
$$h = \mathrm{ReLU}(z_1)$$



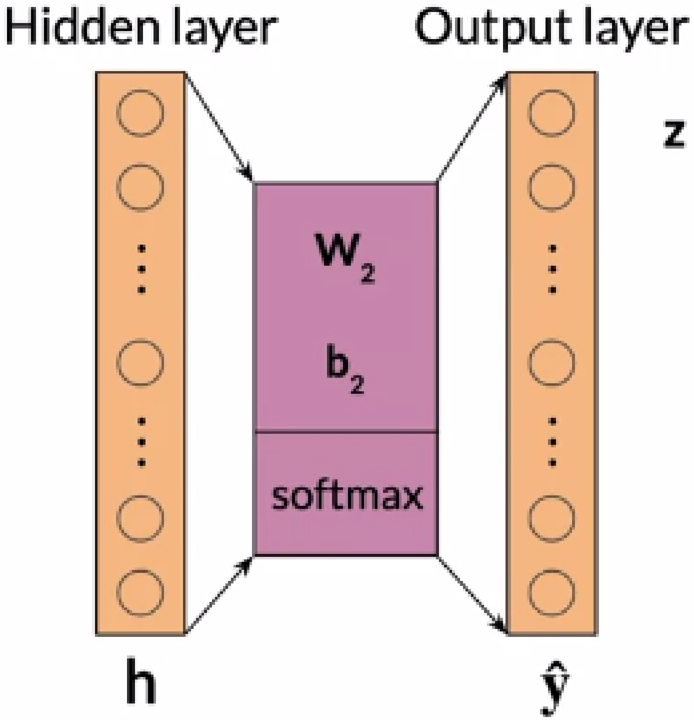
$$\mathrm{ReLU}(x) = \max(0, x)$$



Softmax

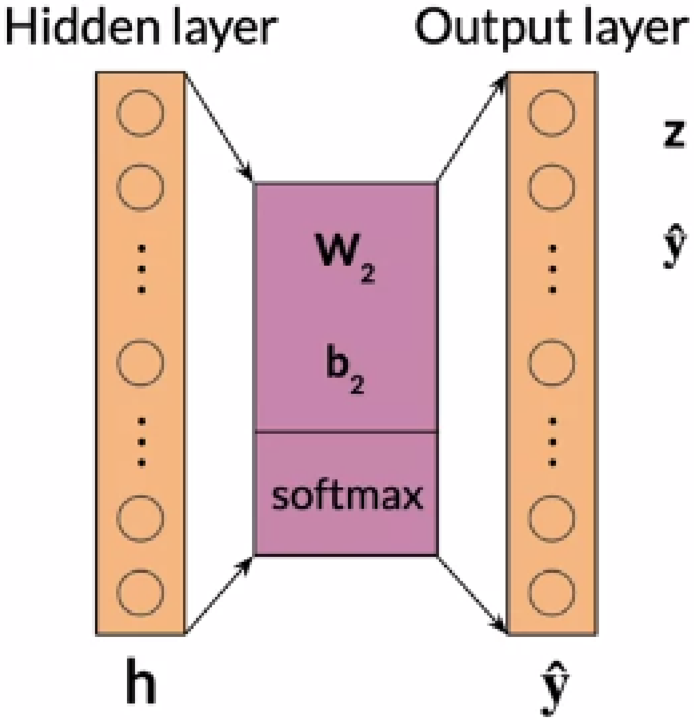


Softmax



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

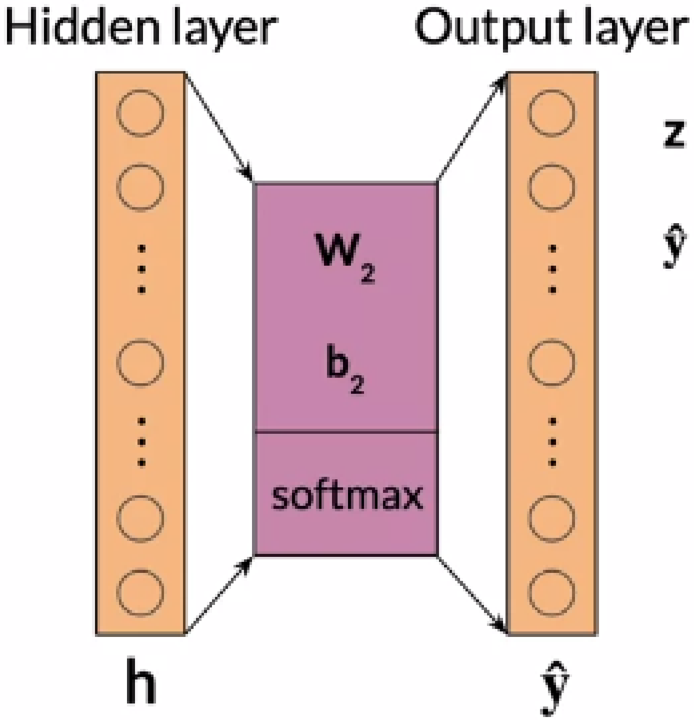
Softmax



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

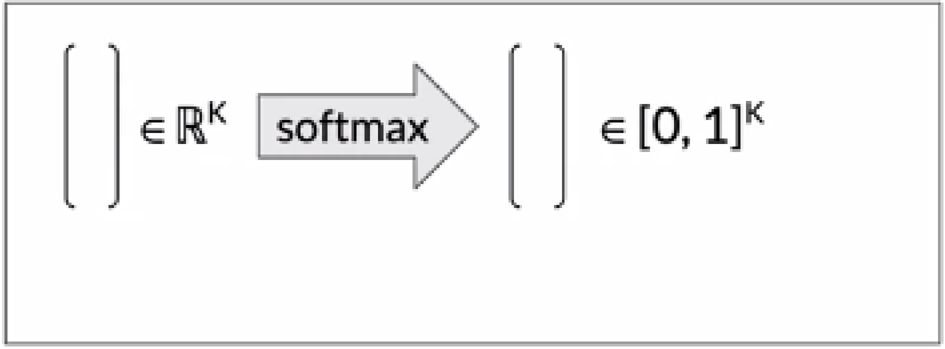
$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{z})$$

Softmax

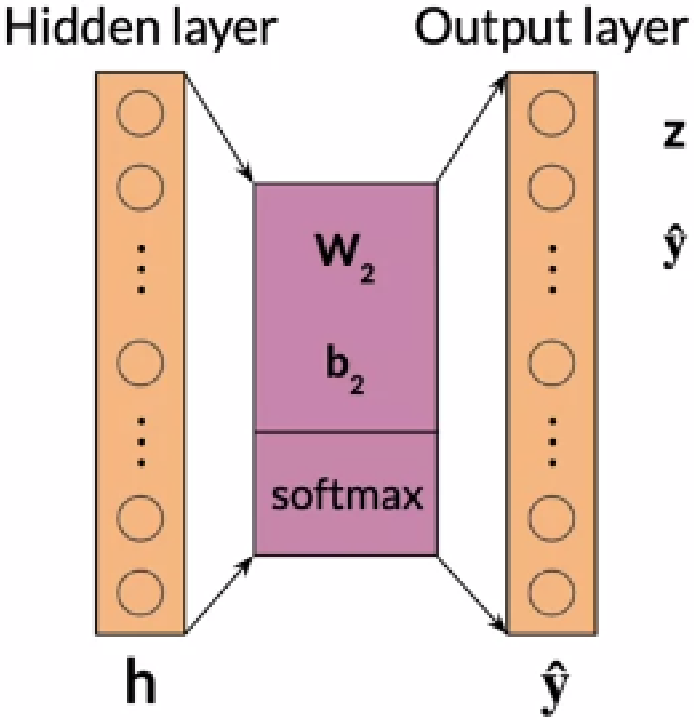


$$\mathbf{z} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

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

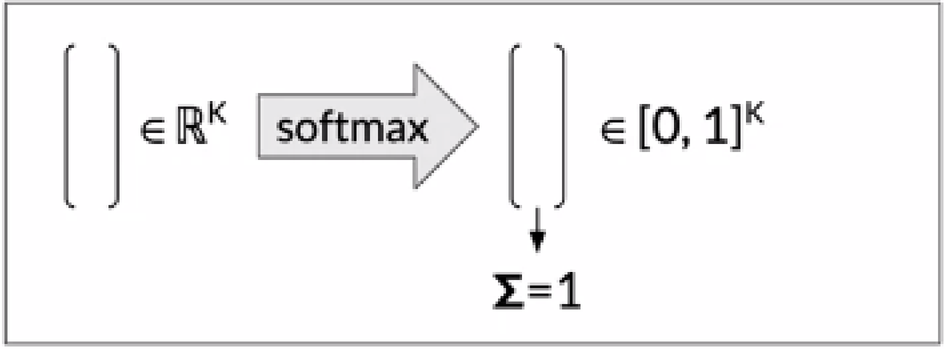


Softmax

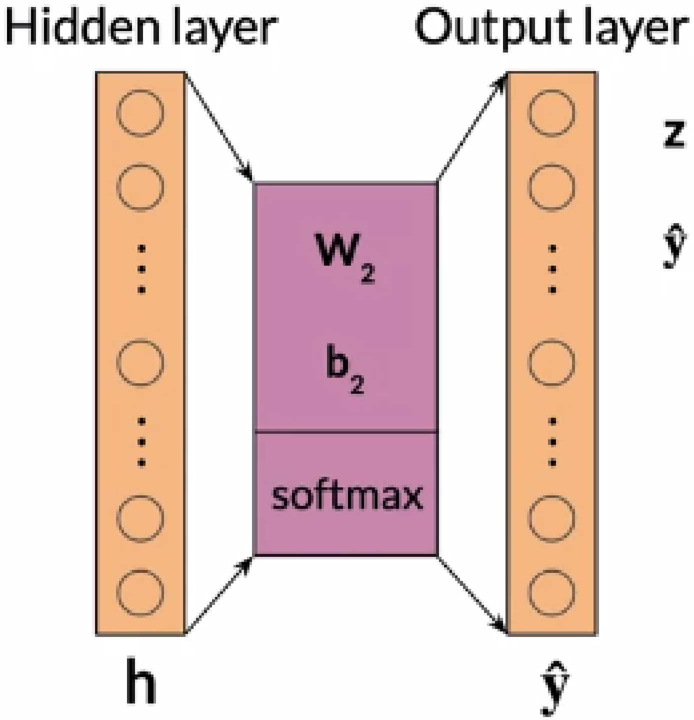


$$\mathbf{z} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

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

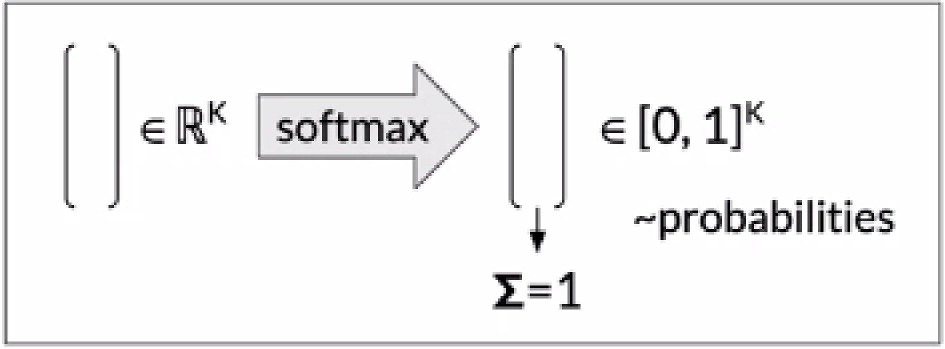


Softmax

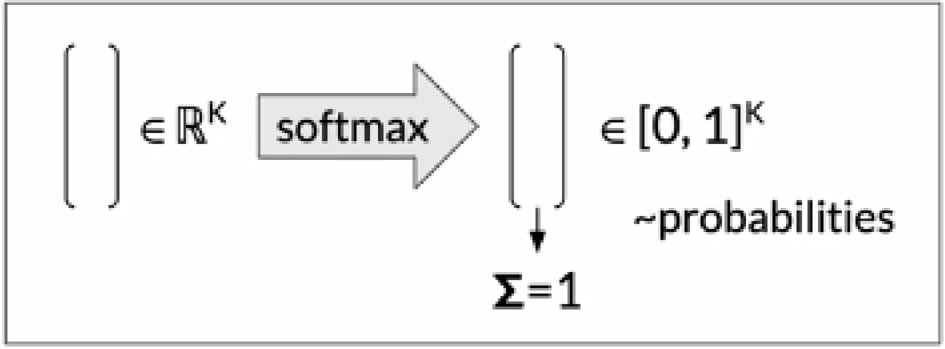


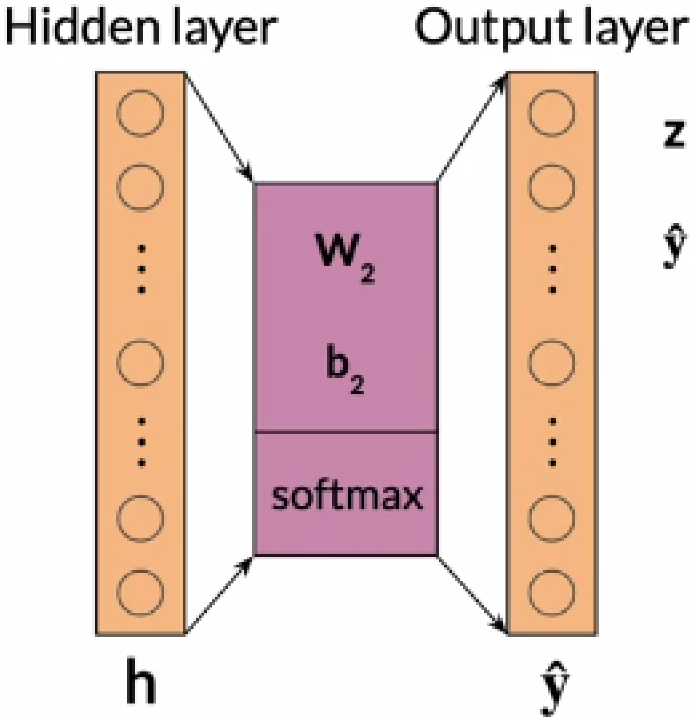
$$\mathbf{z} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

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



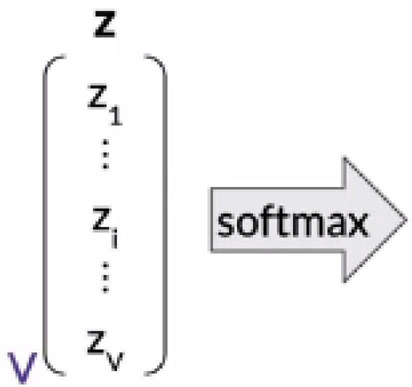
Softmax



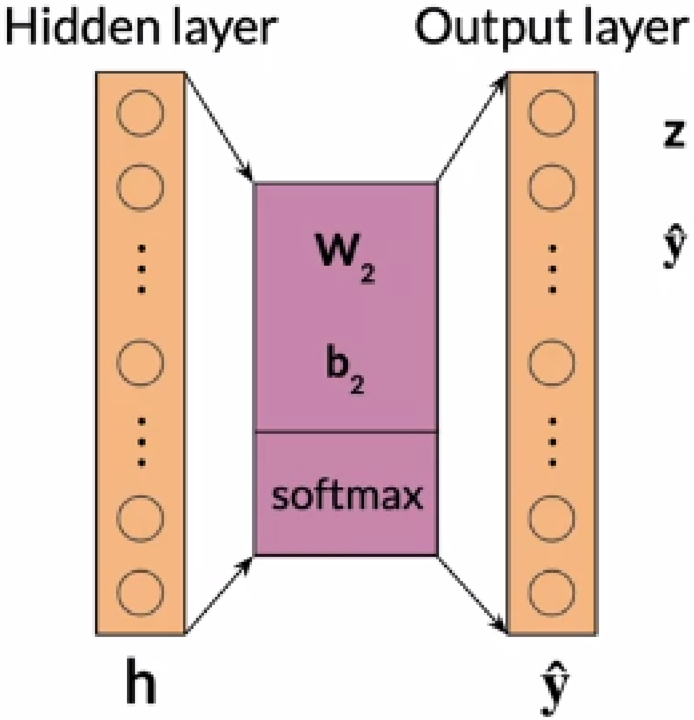


$\mathbf{z} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$

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

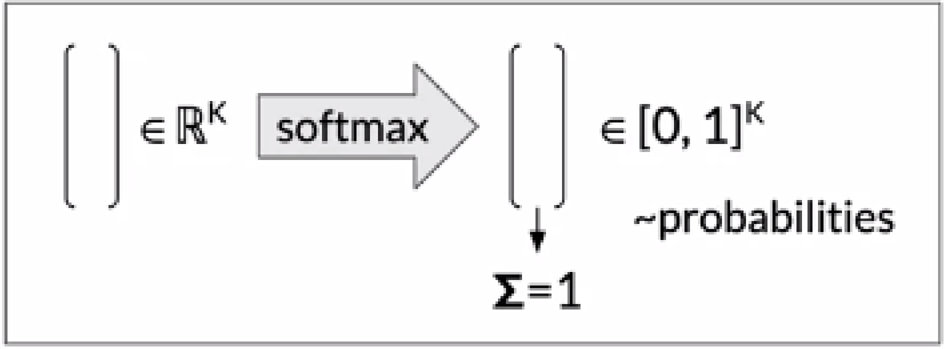


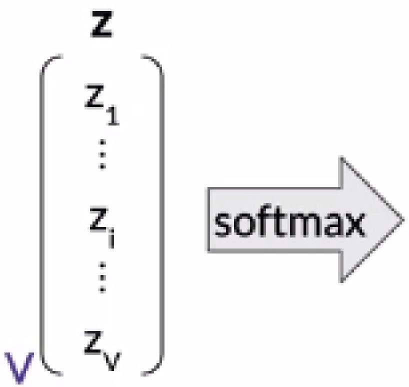
Softmax



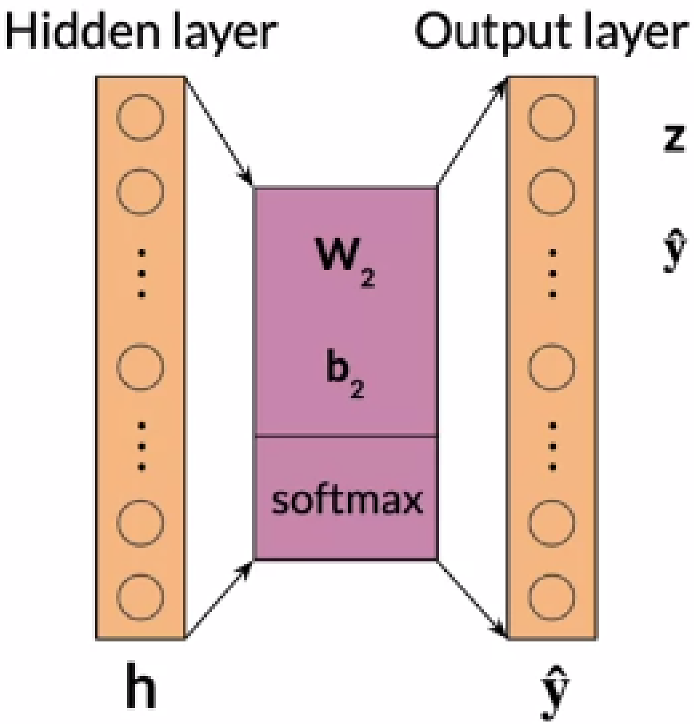
$\mathbf{z} = \mathbf{W_2 h} + \mathbf{b_2}$

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



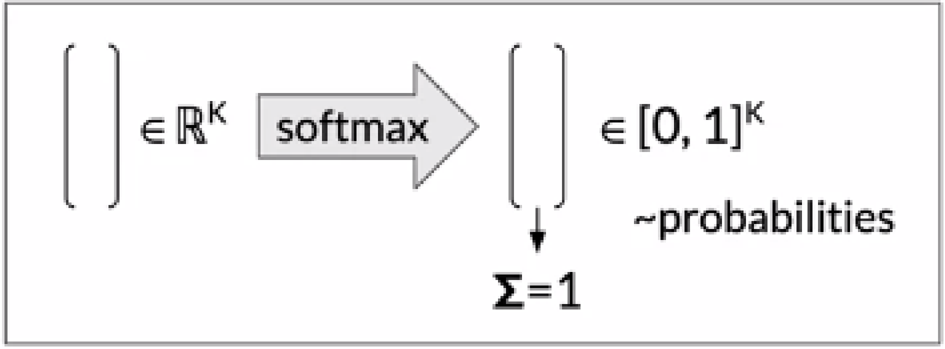


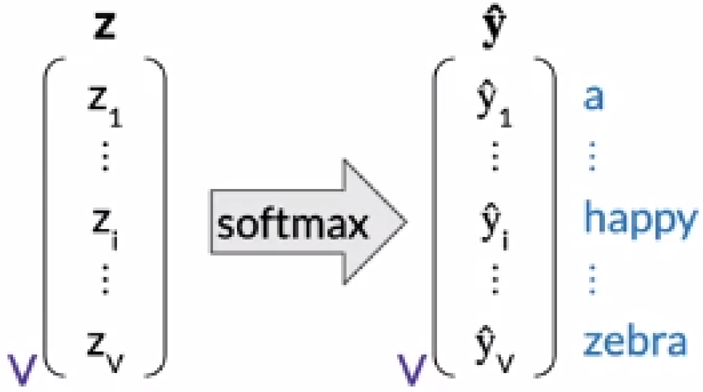
Softmax



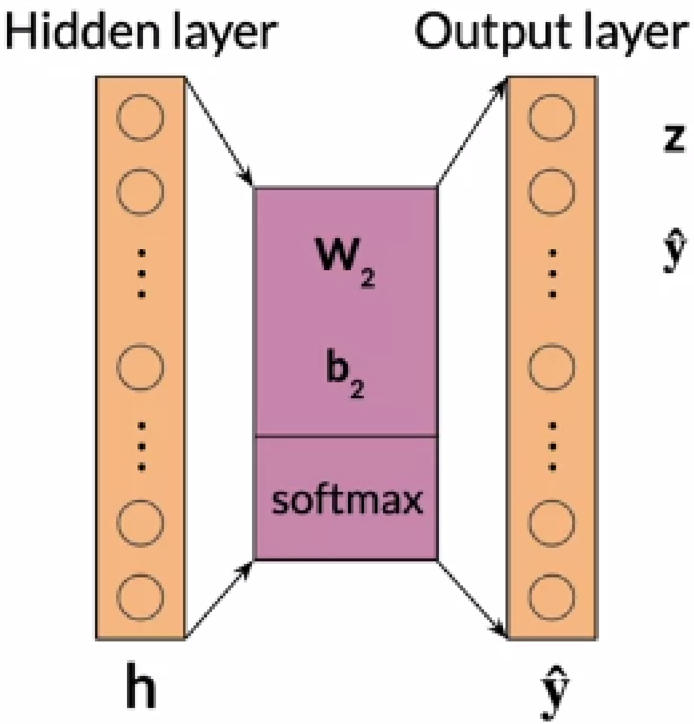
$$\mathbf{z} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

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



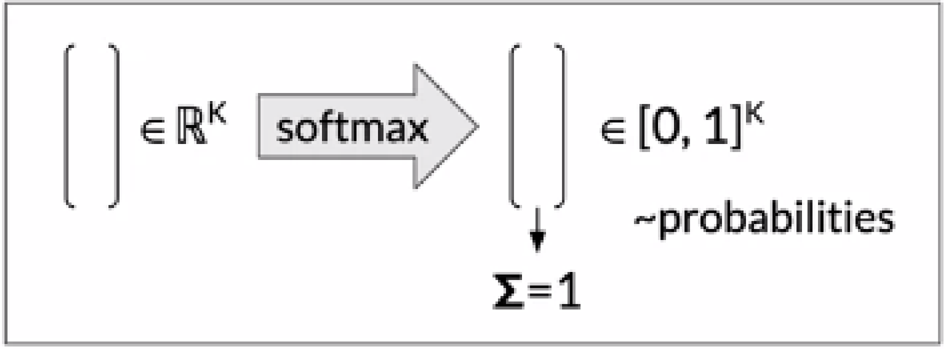


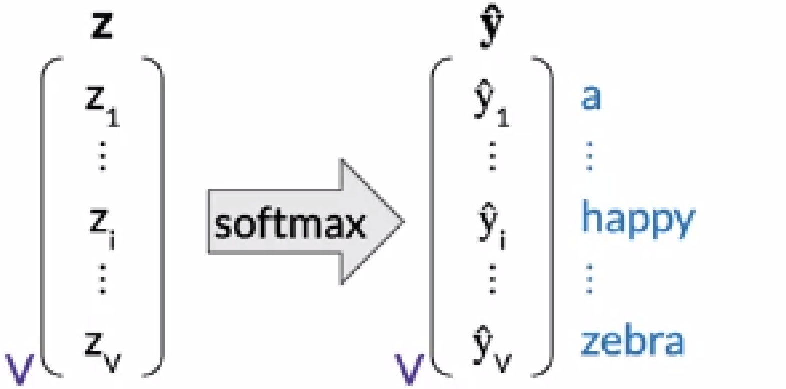
Softmax



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

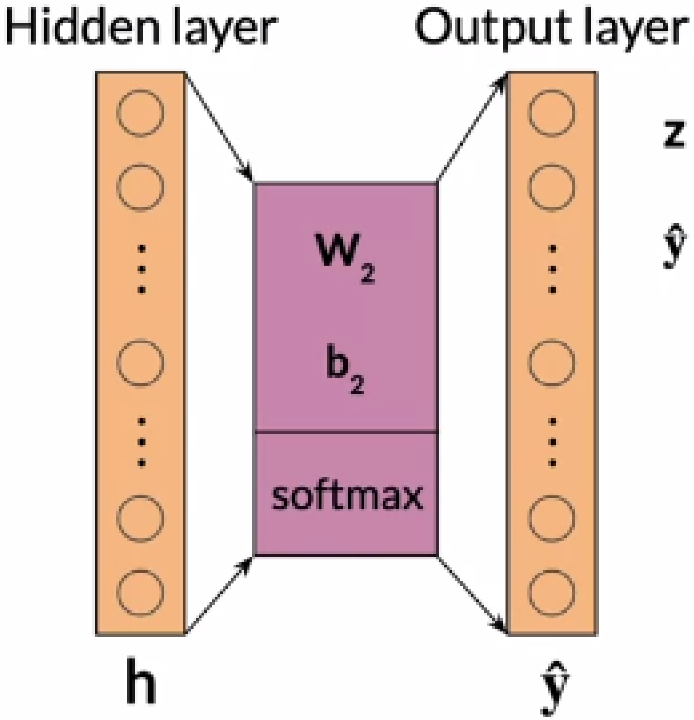
$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$$





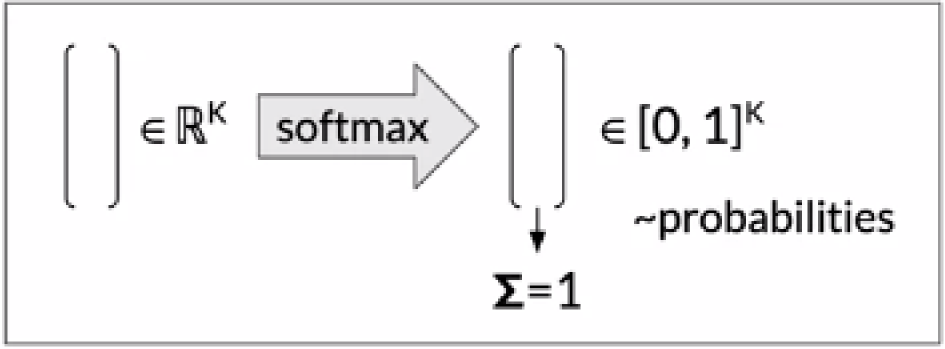
Probabilities of
being center word

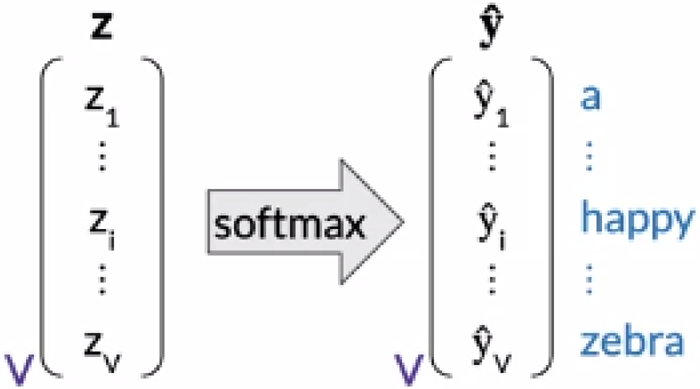
Softmax



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

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

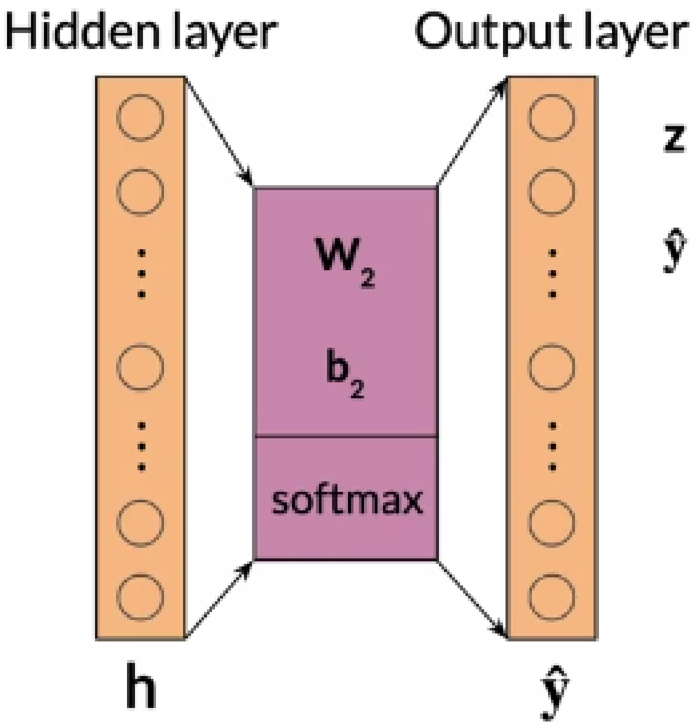




Probabilities of being center word

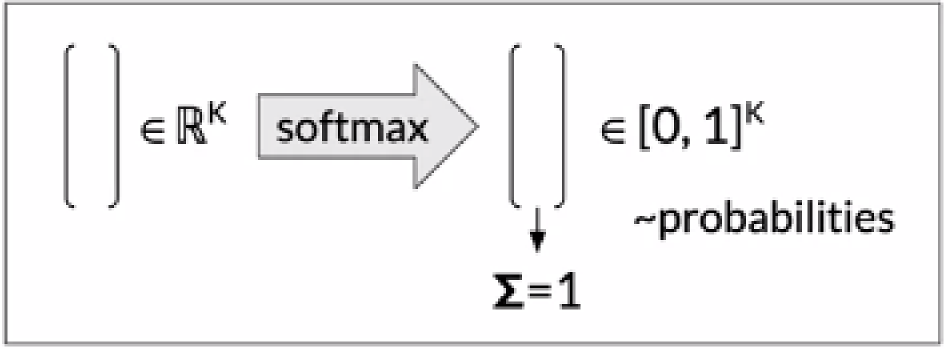
$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}}$$

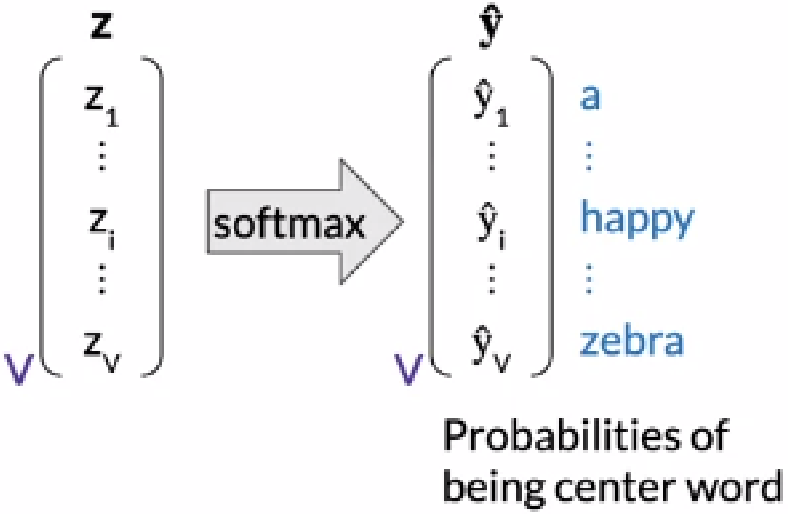
Softmax



$\mathbf{z} = \mathbf{W_2 h} + \mathbf{b_2}$

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





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

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

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

$$\mathbf{z} = \begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix} \xrightarrow{\text{exp}} \exp(\mathbf{z}) = \begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix}$$

Softmax: example

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

$$\underset{\mathbf{z}}{\begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix}} \xrightarrow{\text{exp}} \underset{\exp(\mathbf{z})}{\begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix}}$$

$$\Sigma = 97899$$

Softmax: example

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

$$\underset{\mathbf{z}}{\begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix}} \xrightarrow{\text{exp}} \underset{\exp(\mathbf{z})}{\begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix}}$$

$$\boxed{\Sigma=97899}$$

Softmax: example

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

$$\underset{\mathbf{z}}{\begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix}} \xrightarrow{\text{exp}} \underset{\exp(\mathbf{z})}{\begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix}} \xrightarrow{/\,\boldsymbol{\Sigma}} \underset{\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})}{\begin{pmatrix} \\ \\ \\ \\ \end{pmatrix}}$$

$$\boxed{\Sigma = 97899}$$

Softmax: example

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

$$\mathbf{z} = \begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix} \xrightarrow{\text{exp}} \exp(\mathbf{z}) = \begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix} \xrightarrow{/\,\boldsymbol{\Sigma}} \hat{\mathbf{y}} = \text{softmax}(\mathbf{z}) = \begin{pmatrix} 0.083 \\ \\ \\ \\ \\ \end{pmatrix}$$

$$\boxed{\Sigma = 97899}$$

Softmax: example

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

$$\mathbf{z} = \begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix} \xrightarrow{\text{exp}} \exp(\mathbf{z}) = \begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix} \xrightarrow{/\,\boldsymbol{\Sigma}} \hat{\mathbf{y}} = \text{softmax}(\mathbf{z}) = \begin{pmatrix} 0.083 \\ \\ \\ \\ \\ \end{pmatrix}$$

$$\boxed{\Sigma = 97899}$$

Softmax: example

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

$$\underset{\mathbf{z}}{\begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix}} \xrightarrow{\text{exp}} \underset{\text{exp}(\mathbf{z})}{\begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix}} \xrightarrow{/\,\boldsymbol{\Sigma}} \underset{\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})}{\begin{pmatrix} 0.083 \\ 0.03 \\ 0.612 \\ 0.225 \\ 0.05 \end{pmatrix}}$$

$$\boxed{\Sigma = 97899}$$

Softmax: example

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

$$\overset{\mathbf{z}}{\begin{pmatrix}9\\8\\11\\10\\8.5\end{pmatrix}} \xrightarrow{\text{exp}} \underset{\Sigma=97899}{\overset{\exp(\mathbf{z})}{\begin{pmatrix}8103\\2981\\59874\\22026\\4915\end{pmatrix}}} \xrightarrow{/\,\Sigma} \underset{\Sigma=1}{\overset{\hat{\mathbf{y}}\,=\,\mathrm{softmax}(\mathbf{z})}{\begin{pmatrix}0.083\\0.03\\0.612\\0.225\\0.05\end{pmatrix}}}$$

Softmax: example

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

$$\begin{array}{cccccc}
\mathbf{z} & & \exp(\mathbf{z}) & & \hat{\mathbf{y}} = \text{softmax}(\mathbf{z}) & \\
\begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix} & \xrightarrow{\exp} & \begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix} & \xrightarrow{/\,\Sigma} & \begin{pmatrix} 0.083 \\ 0.03 \\ 0.612 \\ 0.225 \\ 0.05 \end{pmatrix} & \begin{matrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{matrix} \\
 & & \downarrow & & \downarrow & \\
 & & \Sigma=97899 & & \Sigma=1 &
\end{array}$$

Softmax: example

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

$$\underset{\mathbf{z}}{\begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix}} \xrightarrow{\text{exp}} \underset{\text{exp}(\mathbf{z})}{\begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix}} \xrightarrow{/\,\Sigma} \underset{\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})}{\begin{pmatrix} 0.083 \\ 0.03 \\ 0.612 \\ 0.225 \\ 0.05 \end{pmatrix}} \begin{matrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{matrix}$$

$\boxed{\Sigma=97899}$

$\Sigma=1$

Softmax: example

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

$\mathbf{z}$		$\exp(\mathbf{z})$		$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$	
9		8103		0.083	am
8		2981		**0.03**	**because**
11	exp	59874	/ Σ	0.612	happy
10		22026		0.225	I
8.5		4915		0.05	learning
		$\boxed{\Sigma=97899}$		$\Sigma=1$	

Softmax: example

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

$$\mathbf{z} = \begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix} \xrightarrow{\text{exp}} \exp(\mathbf{z}) = \begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix} \xrightarrow{/\Sigma} \hat{\mathbf{y}} = \text{softmax}(\mathbf{z}) = \begin{pmatrix} 0.083 \\ 0.03 \\ 0.612 \\ 0.225 \\ 0.05 \end{pmatrix} \begin{matrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{matrix}$$

$$\boxed{\Sigma = 97899} \qquad \Sigma = 1$$

Softmax: example

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

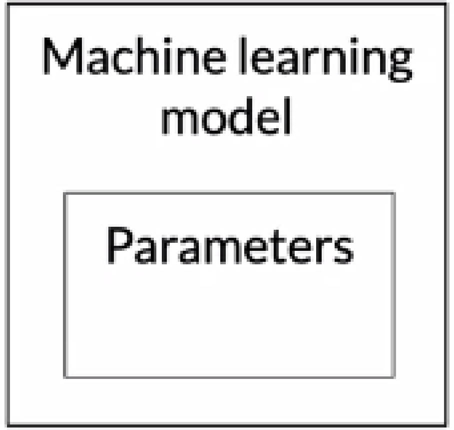
$$\mathbf{z} = \begin{pmatrix} 9 \\ 8 \\ 11 \\ 10 \\ 8.5 \end{pmatrix} \xrightarrow{\text{exp}} \exp(\mathbf{z}) = \begin{pmatrix} 8103 \\ 2981 \\ 59874 \\ 22026 \\ 4915 \end{pmatrix} \xrightarrow{/\,\Sigma} \hat{\mathbf{y}} = \text{softmax}(\mathbf{z}) = \begin{pmatrix} 0.083 \\ 0.03 \\ 0.612 \\ 0.225 \\ 0.05 \end{pmatrix} \begin{matrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{matrix}$$

← Predicted center word

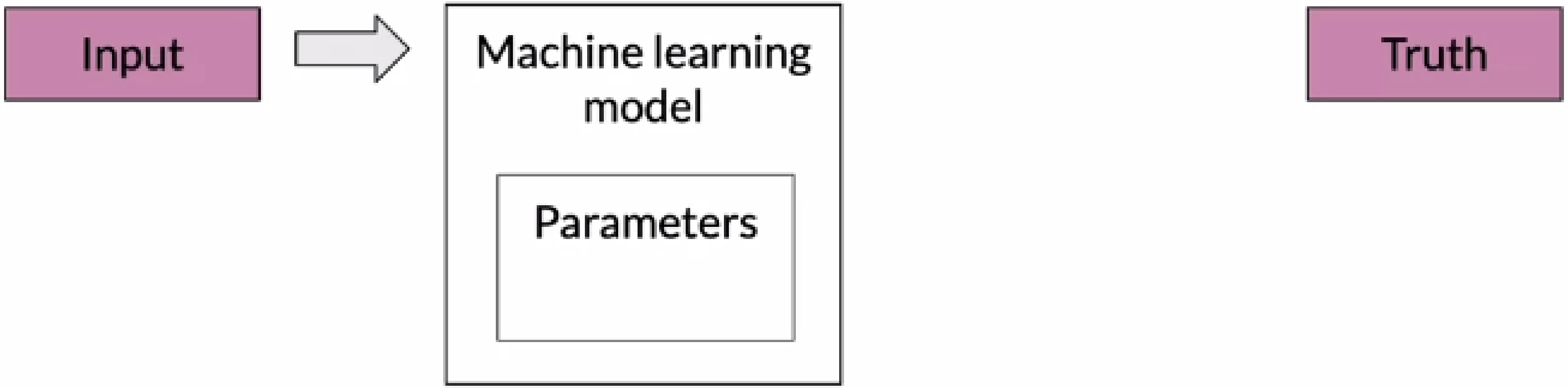
$\boxed{\Sigma = 97899}$

$\Sigma = 1$

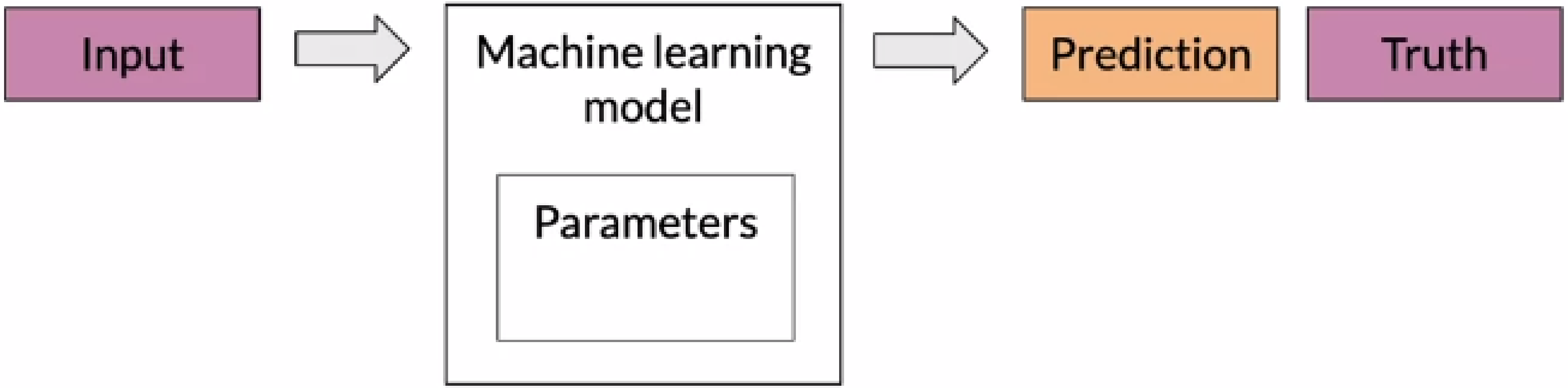
Loss



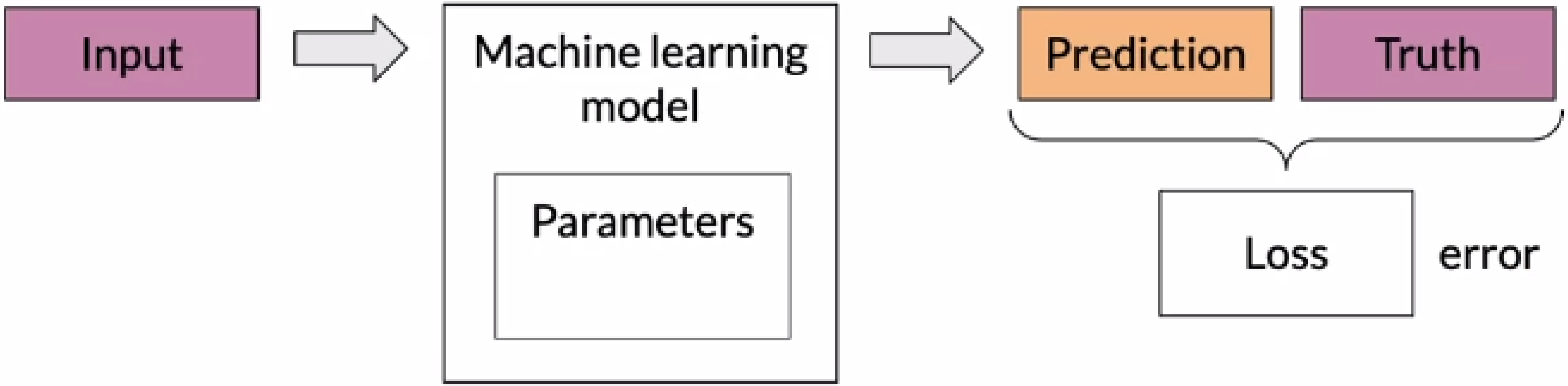
Loss



Loss

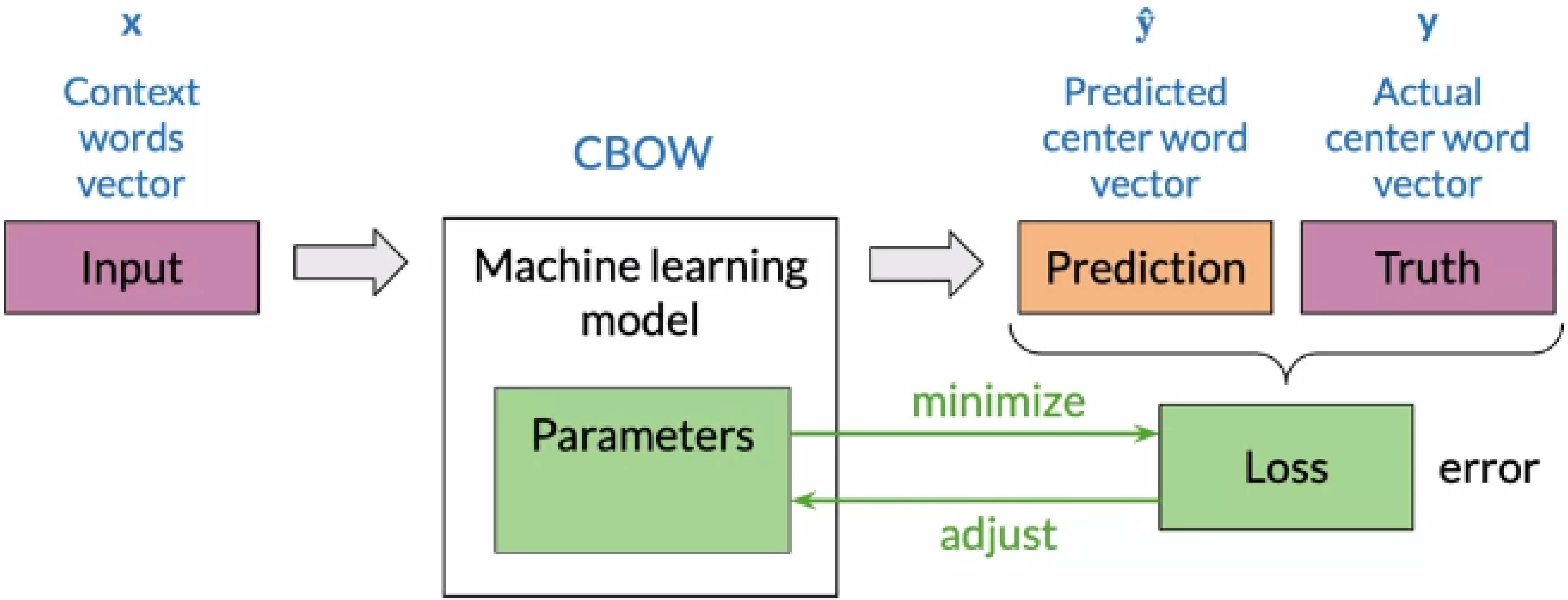

Input
Machine learning model
Parameters
Prediction
Truth

Loss



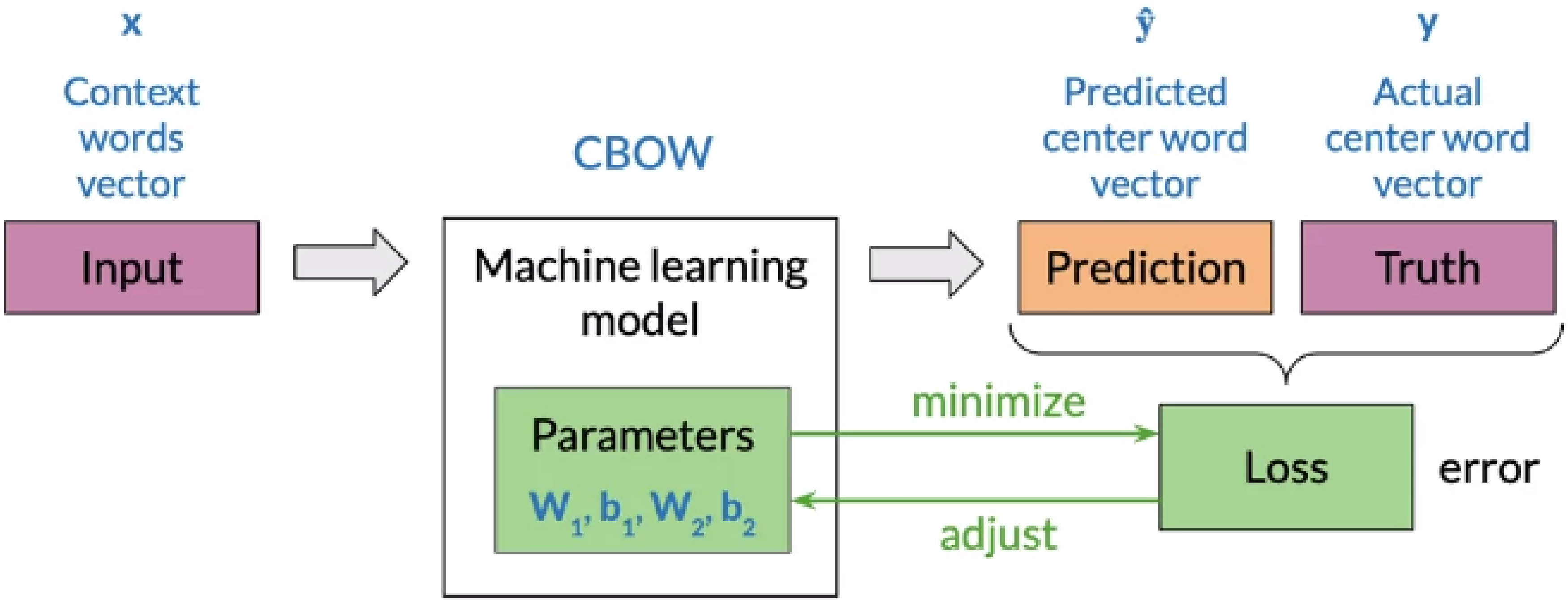
Input
Machine learning model
Parameters
Prediction
Truth
Loss
error

Loss



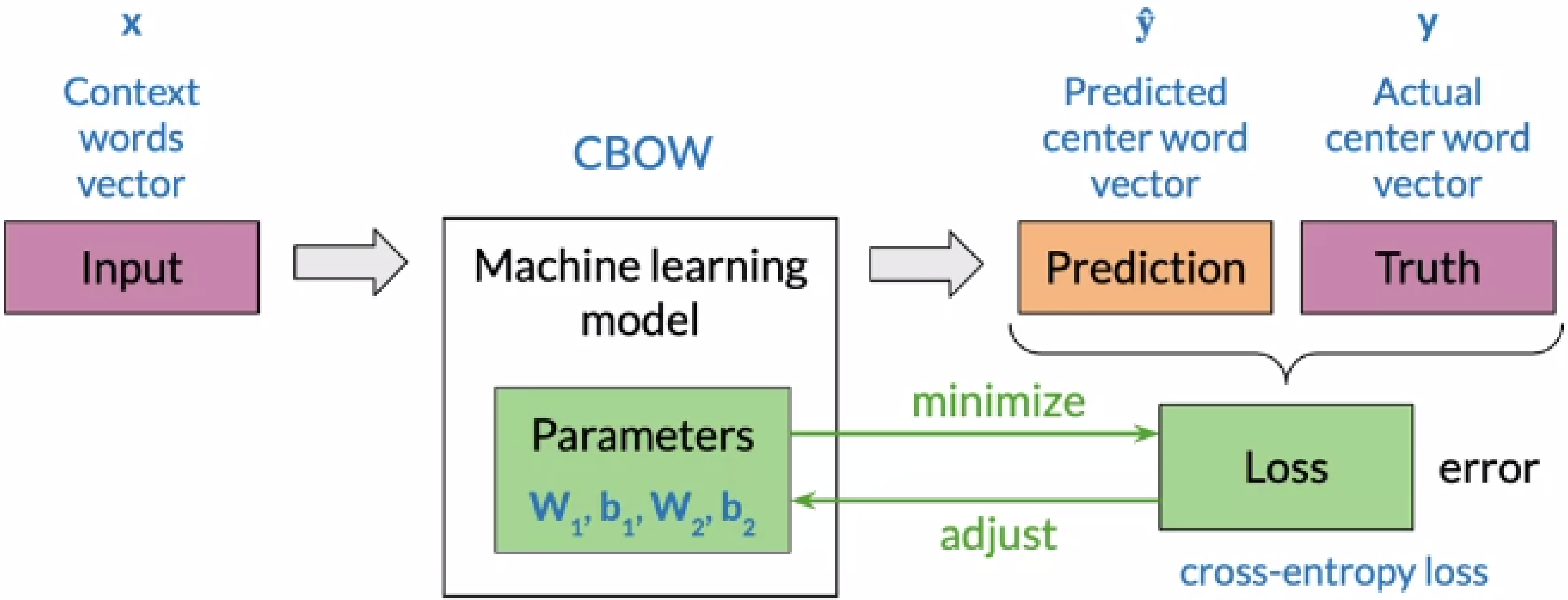
x
Context words vector
Input
CBOW
Machine learning model
Parameters
ŷ
Predicted center word vector
Prediction
y
Actual center word vector
Truth
minimize
adjust
Loss
error

Loss



x
Context words vector
Input
CBOW
Machine learning model
Parameters
W_1, b_1, W_2, b_2
ŷ
Predicted center word vector
Prediction
y
Actual center word vector
Truth
minimize
adjust
Loss
error

Loss



x
Context words vector
Input
CBOW
Machine learning model
Parameters
W_1, b_1, W_2, b_2
ŷ
Predicted center word vector
Prediction
y
Actual center word vector
Truth
minimize
adjust
Loss
error
cross-entropy loss

Cross-entropy loss

Actual

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_V \end{pmatrix}$$

Predicted

$$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_V \end{pmatrix}$$

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

$$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_V \end{pmatrix}$$

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

Cross-entropy loss

Actual

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_V \end{pmatrix}$$

Predicted

$$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_V \end{pmatrix}$$

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

I am happy because I am learning

$$\begin{array}{cl} & \mathbf{y} \\ \left(\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{array}\right) & \begin{array}{l} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} \end{array}$$

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

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

Cross-entropy loss

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

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

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

Cross-entropy loss

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

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

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

Cross-entropy loss

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

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

I am happy because I am learning

$$\overset{\mathbf{y}}{\begin{pmatrix}0\\0\\1\\0\\0\end{pmatrix}} \begin{matrix}\text{am}\\ \text{because}\\ \text{happy}\\ \text{I}\\ \text{learning}\end{matrix} \quad \overset{\hat{\mathbf{y}}}{\begin{pmatrix}0.083\\0.03\\0.611\\0.225\\0.05\end{pmatrix}} \xrightarrow{\log} \overset{\log(\hat{\mathbf{y}})}{\begin{pmatrix}-2.49\\-3.49\\-0.49\\-1.49\\-2.49\end{pmatrix}} \xrightarrow{\odot\,\mathbf{y}} \overset{\mathbf{y}\odot\log(\hat{\mathbf{y}})}{\begin{pmatrix}0\\0\\-0.49\\0\\0\end{pmatrix}}$$

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

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

$\xrightarrow{-\Sigma}$ $J = 0.49$

Cross-entropy loss

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

$$\mathbf{y}$$

$$\begin{array}{cc} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} & \begin{array}{c} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} \end{array}$$

Cross-entropy loss

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

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

Cross-entropy loss

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

$\mathbf{y}$		$\hat{\mathbf{y}}$	log	$\log(\hat{\mathbf{y}})$
0	am	0.96		-0.04
0	because	0.01		-4.61
1	happy	0.01		-4.61
0	I	0.01		-4.61
0	learning	0.01		-4.61

Cross-entropy loss

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

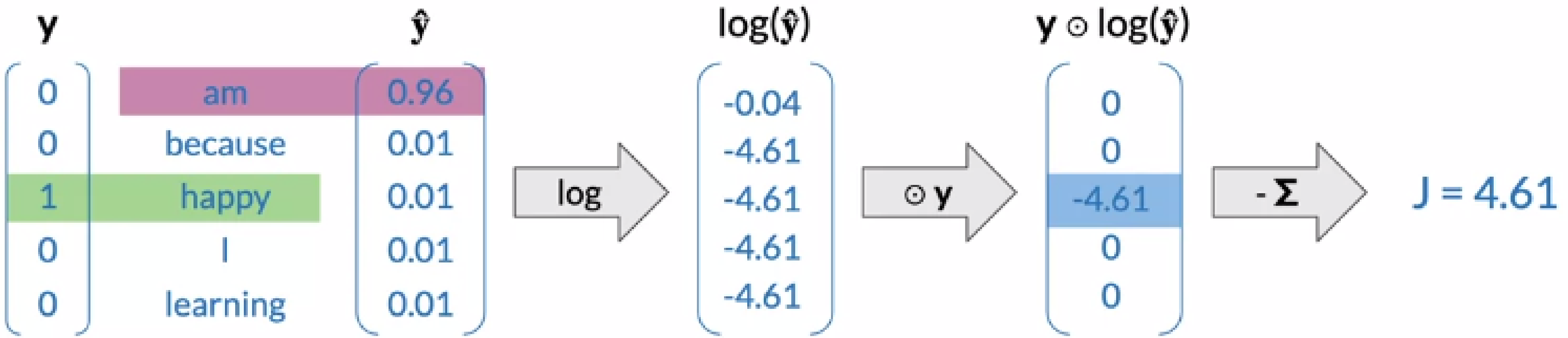
$\mathbf{y}$		$\hat{\mathbf{y}}$	$\log(\hat{\mathbf{y}})$	$\mathbf{y} \odot \log(\hat{\mathbf{y}})$
0	am	0.96	-0.04	0
0	because	0.01	-4.61	0
1	happy	0.01	-4.61	-4.61
0	I	0.01	-4.61	0
0	learning	0.01	-4.61	0

log

$\odot$ **y**

Cross-entropy loss

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



Cross-entropy loss

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

$$\begin{array}{ccc} \mathbf{y} & & \hat{\mathbf{y}} \\ \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} & \begin{array}{c} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} & \begin{pmatrix} 0.96 \\ 0.01 \\ 0.01 \\ 0.01 \\ 0.01 \end{pmatrix} \end{array} \xrightarrow{\log} \begin{array}{c} \log(\hat{\mathbf{y}}) \\ \begin{pmatrix} -0.04 \\ -4.61 \\ -4.61 \\ -4.61 \\ -4.61 \end{pmatrix} \end{array} \xrightarrow{\odot\, \mathbf{y}} \begin{array}{c} \mathbf{y} \odot \log(\hat{\mathbf{y}}) \\ \begin{pmatrix} 0 \\ 0 \\ -4.61 \\ 0 \\ 0 \end{pmatrix} \end{array} \xrightarrow{-\Sigma} \begin{array}{c} J = 4.61 \\ > \\ J^{(\text{correct})} = 0.49 \end{array}$$

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

$\rightarrow J = 4.61$

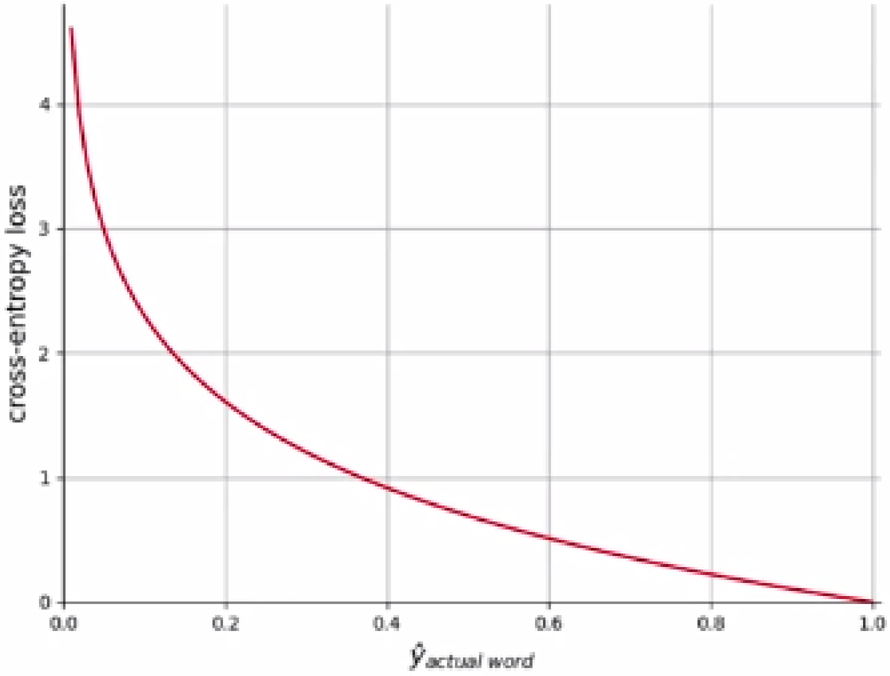
Cross-entropy loss

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

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

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

$\rightarrow J = 4.61$



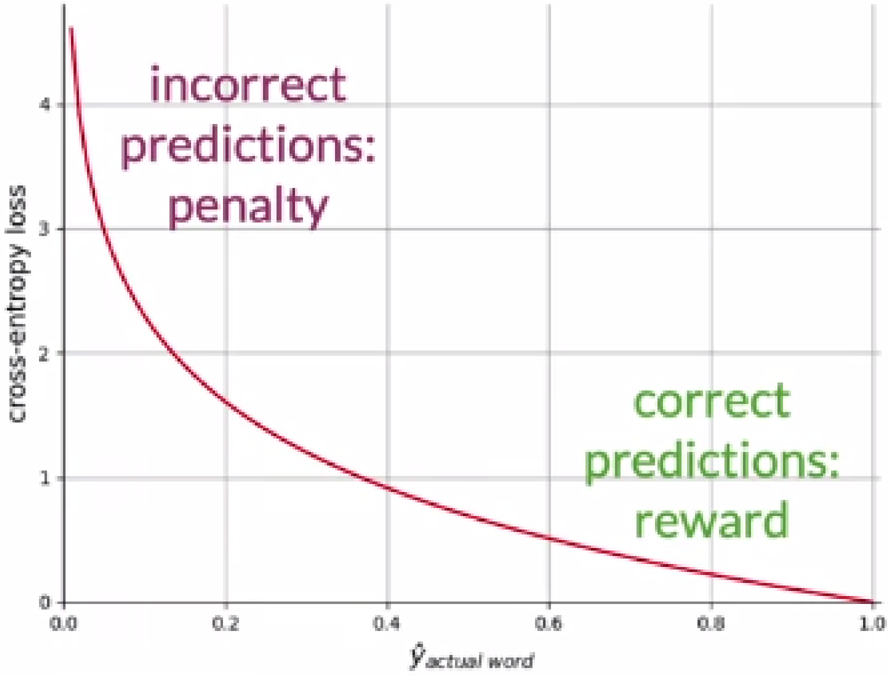
Cross-entropy loss

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

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

y		$\hat{\mathbf{y}}$
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

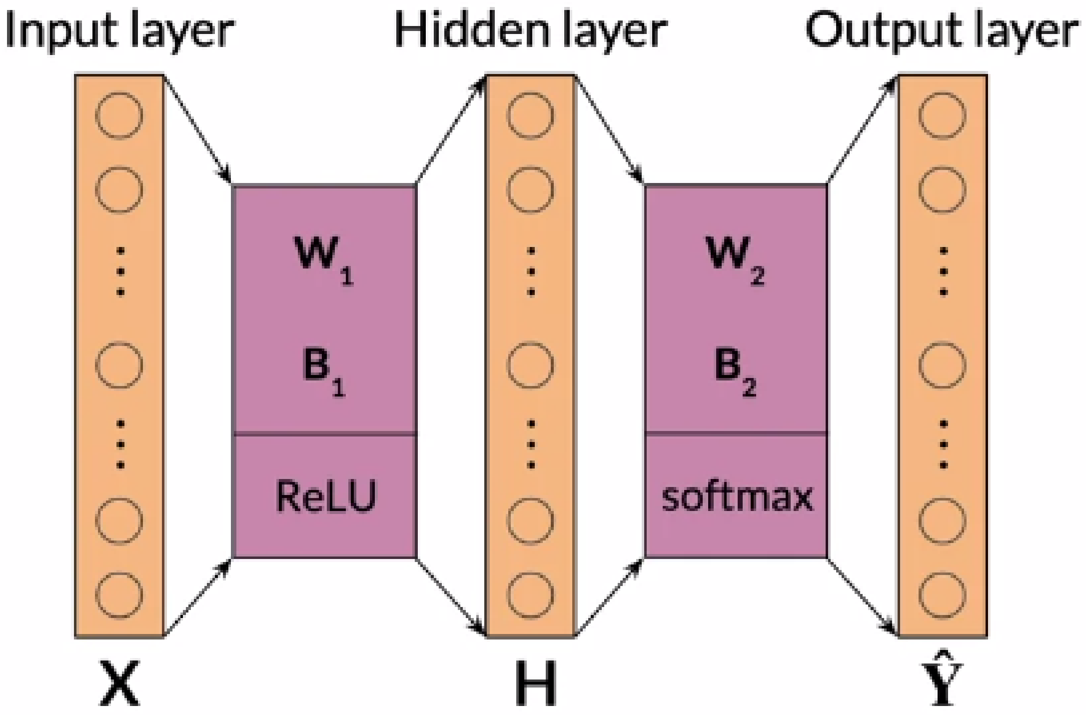
- Forward propagation

- Cost

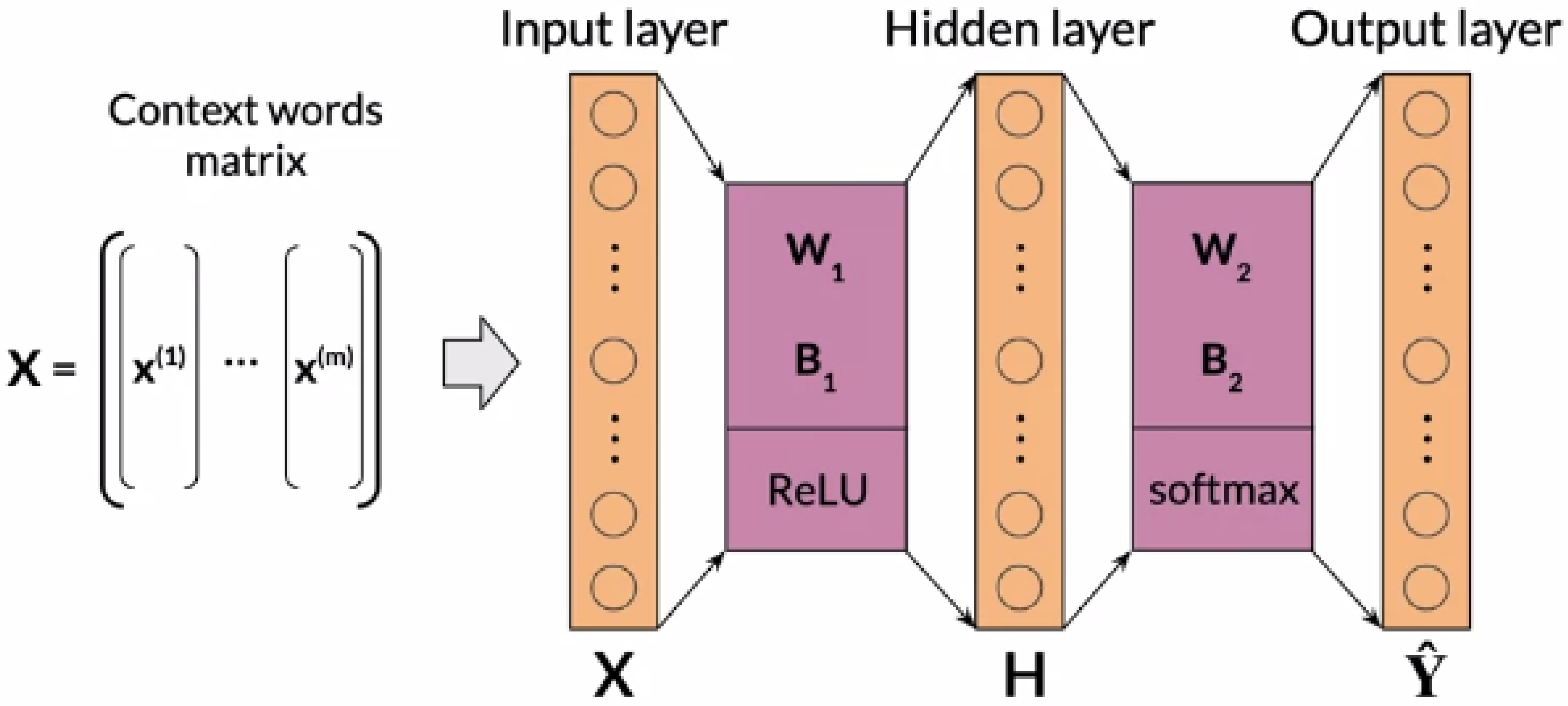
Training process

- Forward propagation
- Cost
- Backpropagation and gradient descent

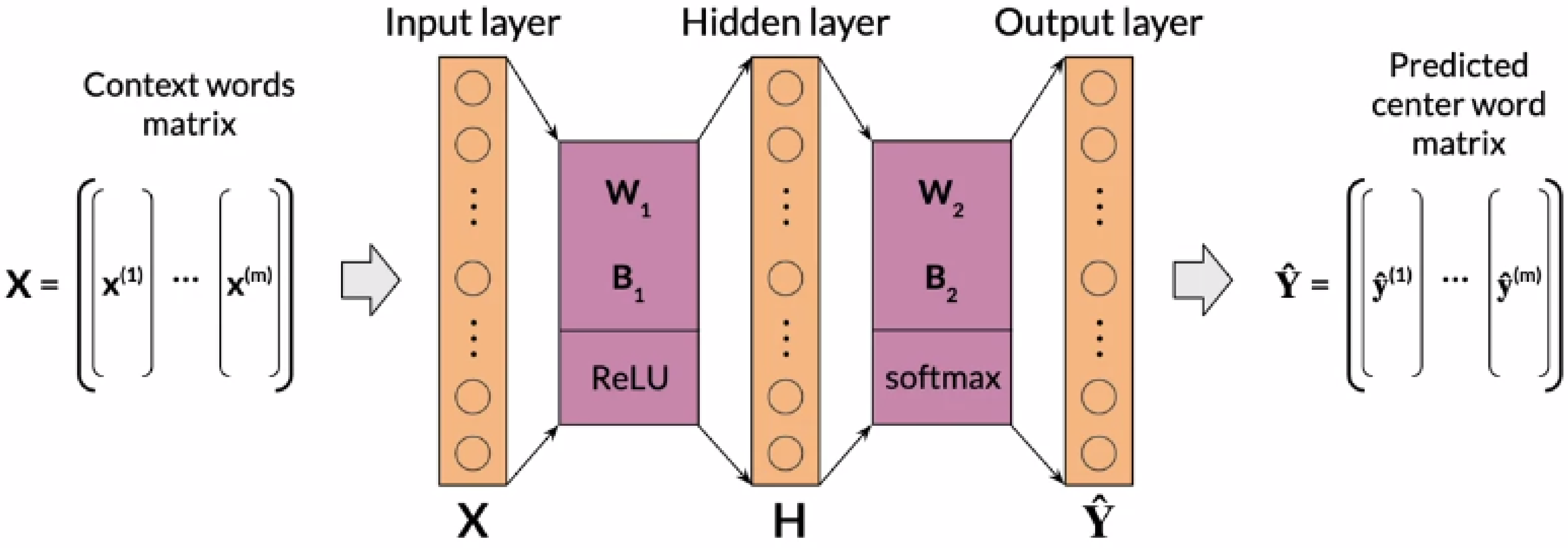
Forward propagation



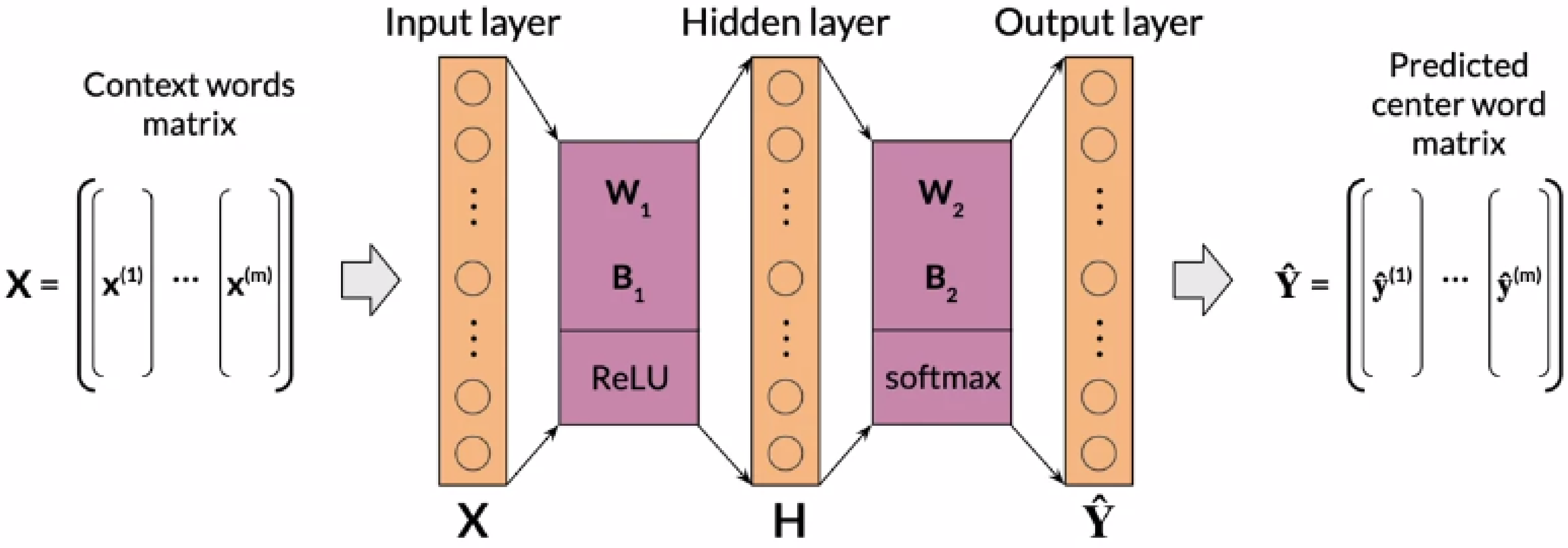
Forward propagation



Forward propagation

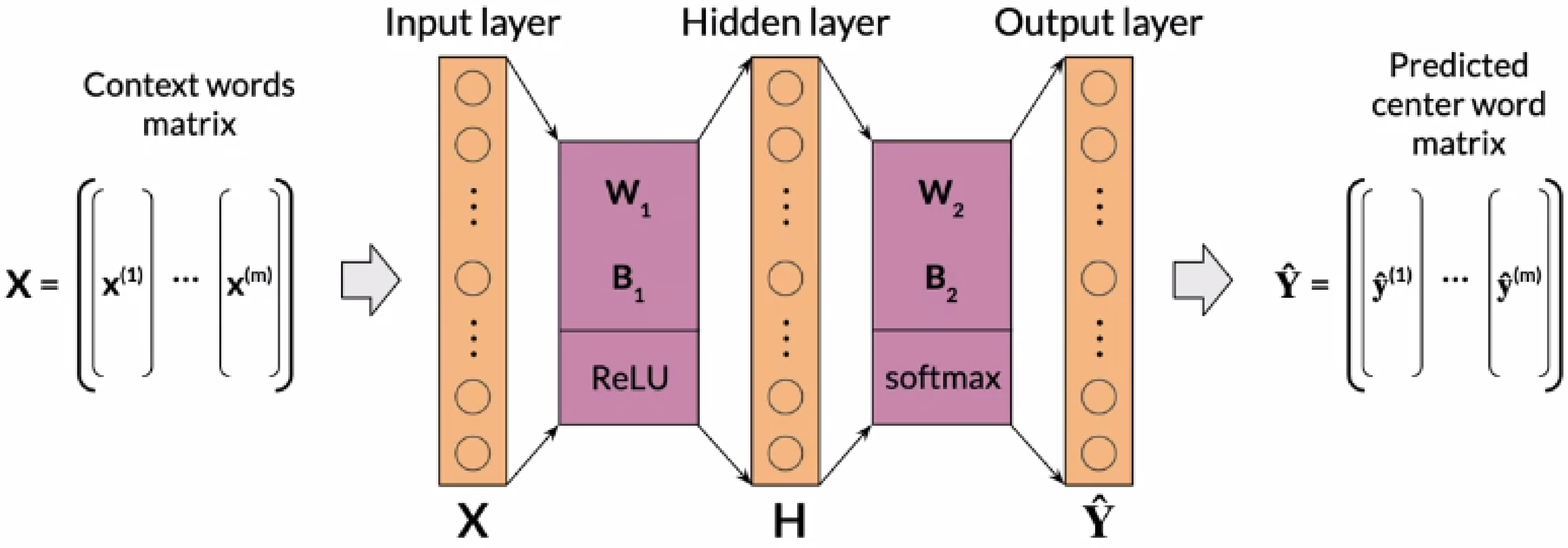


Forward propagation



Forward propagation

$$Z_1 = W_1X + B_1 \qquad Z_2 = W_2H + B_2$$
$$H = \mathrm{ReLU}(Z_1) \qquad \hat{Y} = \mathrm{softmax}(Z_2)$$



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

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

Actual center word matrix

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

Cost

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

Cost: mean of losses

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

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

Predicted center word matrix

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

Actual center word matrix

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

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

$$\frac{\partial J_{batch}}{\partial \mathbf{W_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W_2^\top}(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W_2}}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_1}}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_2}}$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W_2}^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W_2}} = \frac{1}{m}(\hat{\mathbf{Y}} - \mathbf{Y})\mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_1}}$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_2}}$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W_2}^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W_2}} = \frac{1}{m}(\hat{\mathbf{Y}} - \mathbf{Y})\mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W_2}^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{1}_m^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_2}}$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W_2}^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W_2}} = \frac{1}{m}(\hat{\mathbf{Y}} - \mathbf{Y})\mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W_2}^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{1}_m^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_2}}$$

$$\mathbf{1}_m = \underbrace{\begin{pmatrix} 1, \dots, 1 \end{pmatrix}}_{m}$$

$$\mathbf{A}.\mathbf{1}_m^\top = \begin{pmatrix} \boxed{\qquad\qquad} \\ \\ \\ \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} \Sigma \\ \\ \\ \end{pmatrix}$$

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W}_2^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W_2}} = \frac{1}{m}(\hat{\mathbf{Y}} - \mathbf{Y})\mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W}_2^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{1}_m^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_2}}$$

$$\mathbf{1}_m = \underbrace{\begin{pmatrix} 1, \ldots, 1 \end{pmatrix}}_{m}$$

$$\mathbf{A.1}_m^\top = \begin{pmatrix} \boxed{} \\ \\ \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} \Sigma \\ \\ \end{pmatrix}$$

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

Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W}_2^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{X}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{W_2}} = \frac{1}{m}(\hat{\mathbf{Y}} - \mathbf{Y})\mathbf{H}^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_1}} = \frac{1}{m}\mathrm{ReLU}\left(\mathbf{W}_2^\top(\hat{\mathbf{Y}} - \mathbf{Y})\right)\mathbf{1}_m^\top$$

$$\frac{\partial J_{batch}}{\partial \mathbf{b_2}} = \frac{1}{m}(\hat{\mathbf{Y}} - \mathbf{Y})\mathbf{1}_m^\top$$

$$\mathbf{1}_m = \underbrace{\begin{pmatrix} 1, \ldots, 1 \end{pmatrix}}_{m}$$

$$\mathbf{A}.\mathbf{1}_m^\top = \begin{pmatrix} \boxed{} \\ \\ \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} \Sigma \\ \\ \end{pmatrix}$$

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

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

$$\mathbf{W_1} := \mathbf{W_1} - \alpha \frac{\partial J_{batch}}{\partial \mathbf{W_1}}$$

$$\mathbf{W_2} := \mathbf{W_2} - \alpha \frac{\partial J_{batch}}{\partial \mathbf{W_2}}$$

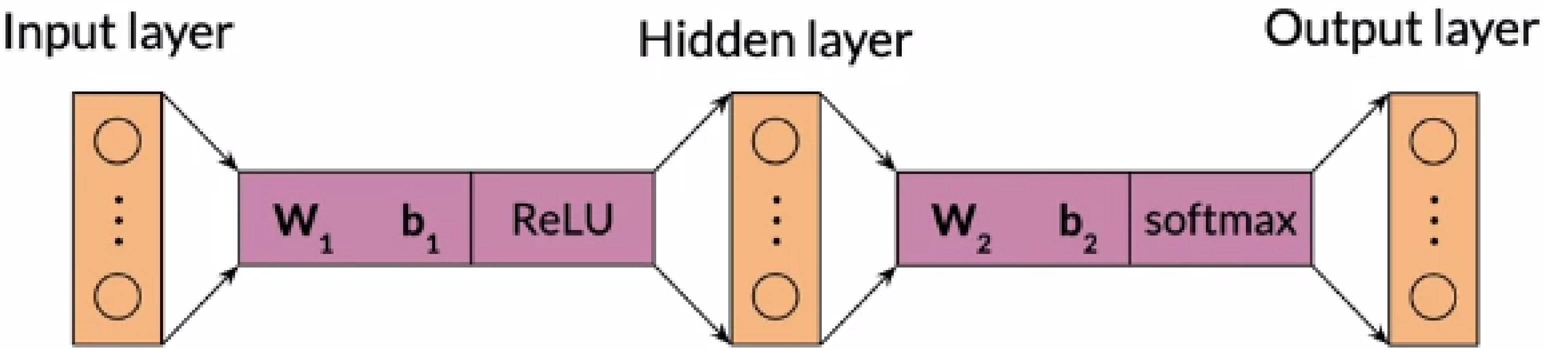
$$\mathbf{b_1} := \mathbf{b_1} - \alpha \frac{\partial J_{batch}}{\partial \mathbf{b_1}}$$

$$\mathbf{b_2} := \mathbf{b_2} - \alpha \frac{\partial J_{batch}}{\partial \mathbf{b_2}}$$

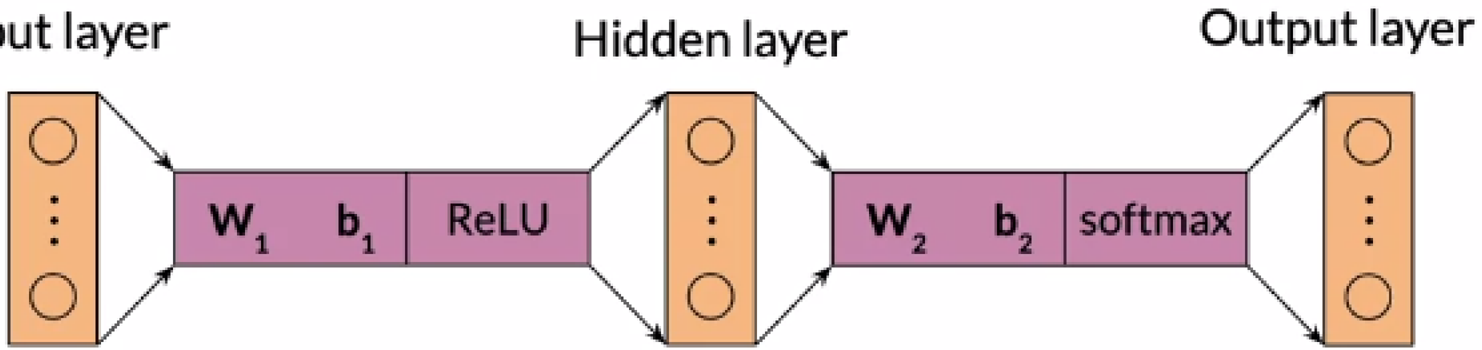
deeplearning.ai

Extracting Word Embedding Vectors

Extracting word embedding vectors: option 1

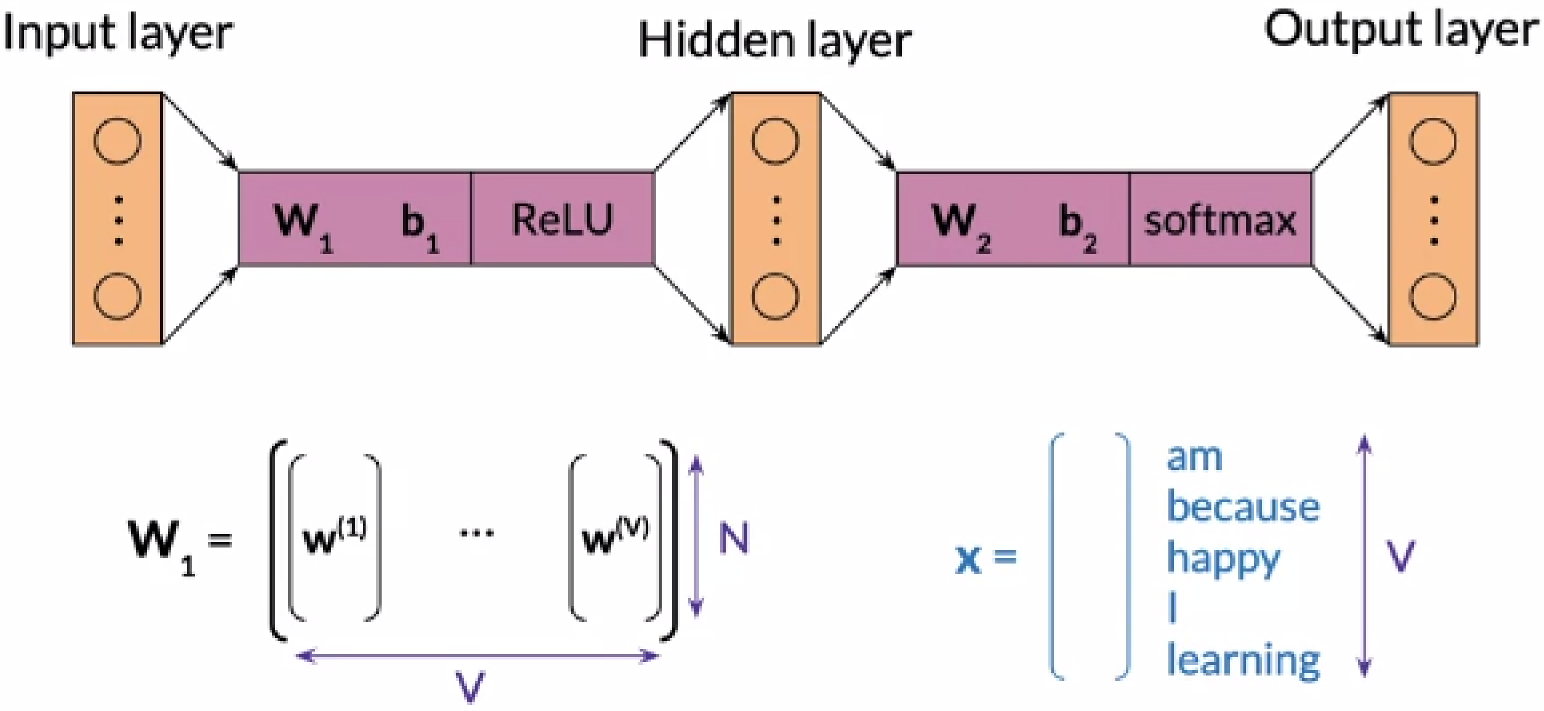


Extracting word embedding vectors: option 1

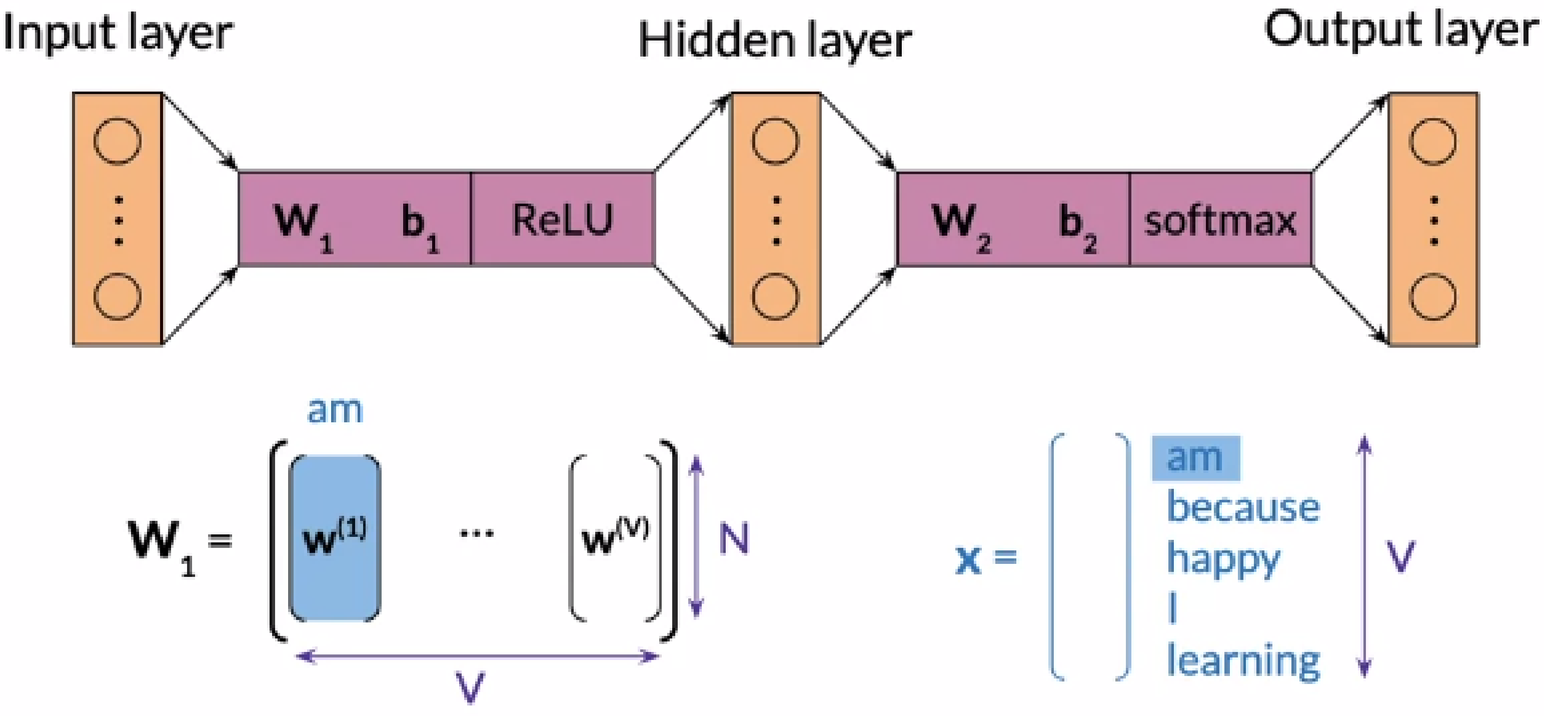


$$\mathbf{W}_1 = \underbrace{\begin{bmatrix} \mathbf{w}^{(1)} & \cdots & \mathbf{w}^{(V)} \end{bmatrix}}_{V} \Big\updownarrow N$$

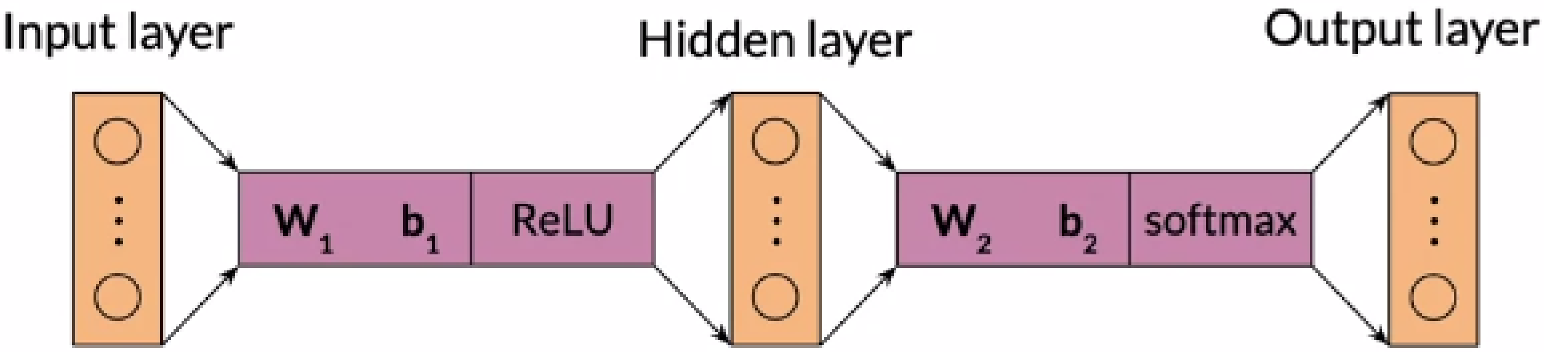
Extracting word embedding vectors: option 1



Extracting word embedding vectors: option 1

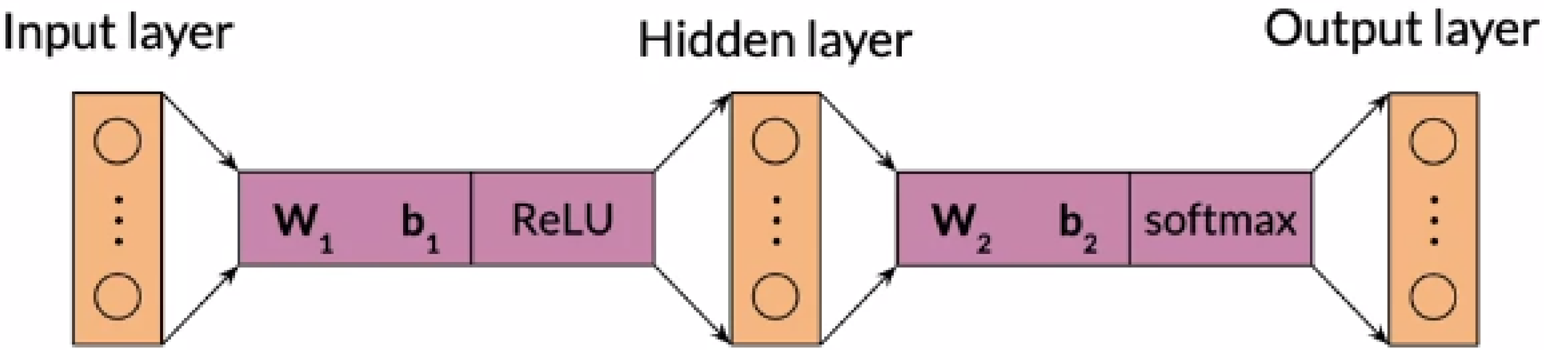


Extracting word embedding vectors: option 2



$$\mathbf{W}_2 = \begin{array}{cc} \begin{bmatrix} \mathbf{w}^{(1)} \\ \dots \\ \mathbf{w}^{(V)} \end{bmatrix} & V \\ N & \end{array}$$

Extracting word embedding vectors: option 2



$$\mathbf{W}_2 = \underbrace{\begin{bmatrix} \mathbf{w}^{(1)} \\ \cdots \\ \mathbf{w}^{(V)} \end{bmatrix}}_{N} \; V$$

$$\mathbf{x} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} \begin{matrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{matrix} \; V$$

Extracting word embedding vectors: option 3

$$\mathbf{W_1} = \begin{pmatrix} \begin{pmatrix} \mathbf{w_1}^{(1)} \end{pmatrix} & \cdots & \begin{pmatrix} \mathbf{w_1}^{(V)} \end{pmatrix} \end{pmatrix} \qquad \mathbf{W_2} = \begin{pmatrix} \begin{pmatrix} \mathbf{w_2}^{(1)} \end{pmatrix} \\ \cdots \\ \begin{pmatrix} \mathbf{w_2}^{(V)} \end{pmatrix} \end{pmatrix}$$

Extracting word embedding vectors: option 3

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

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

Extracting word embedding vectors: option 3

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{w}_1^{(1)} & \cdots & \mathbf{w}_1^{(V)} \end{bmatrix} \qquad \mathbf{W}_2 = \begin{bmatrix} \mathbf{w}_2^{(1)} \\ \cdots \\ \mathbf{w}_2^{(V)} \end{bmatrix}$$

$$\mathbf{W}_3 = 0.5\,(\mathbf{W}_1 + \mathbf{W}_2^{T}) = \underbrace{\begin{bmatrix} \mathbf{w}_3^{(1)} & \cdots & \mathbf{w}_3^{(V)} \end{bmatrix}}_{V} \; \Big\updownarrow N$$

$$\mathbf{x} = \begin{bmatrix} \; \\ \; \\ \; \\ \; \\ \; \end{bmatrix} \begin{matrix} \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{matrix} \; \Big\updownarrow V$$

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

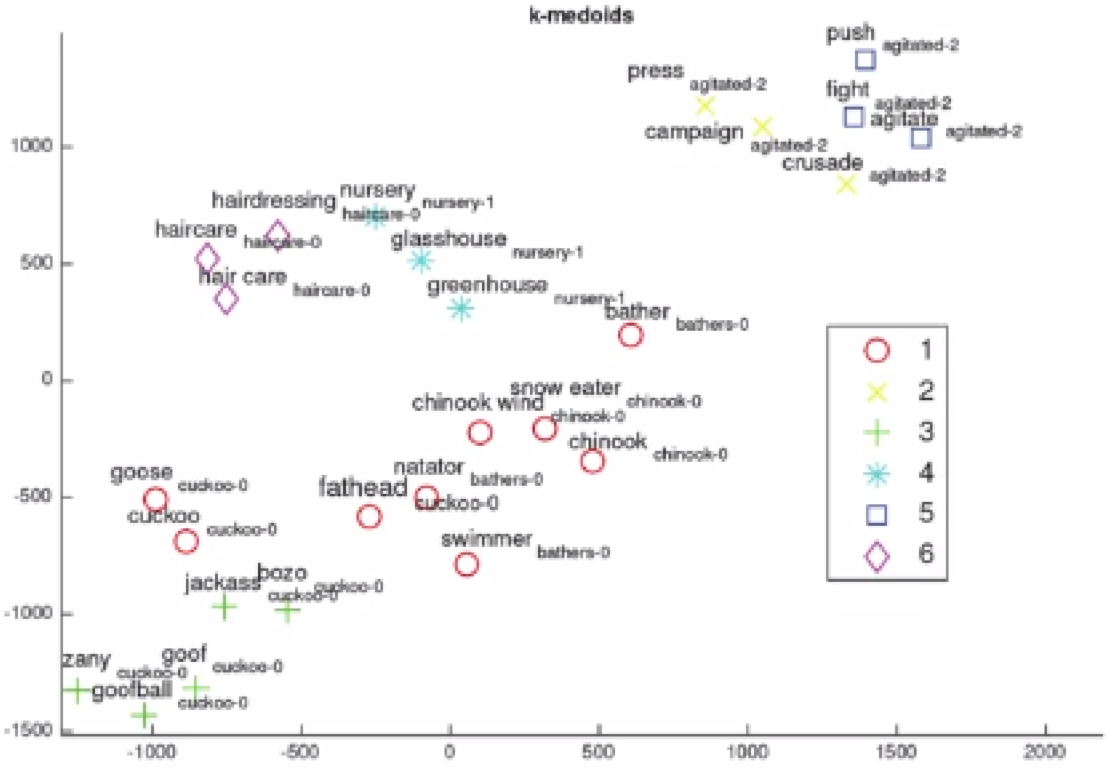
- Analogies
- Clustering

Intrinsic evaluation

Test relationships between words

- Analogies
- Clustering

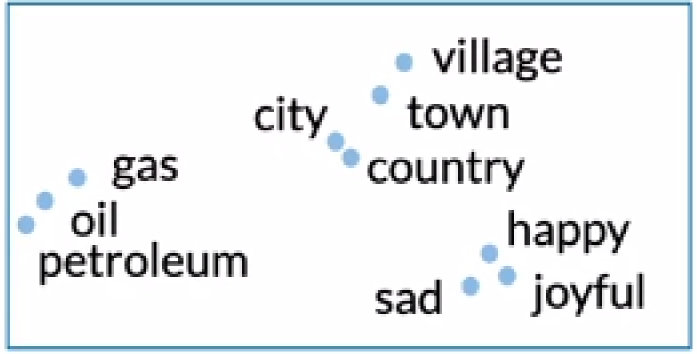
Source: Michael Zhai, Johnny Tan, and Jinho D. Choi. 2016. [Intrinsic and extrinsic evaluations of word embeddings]



Intrinsic evaluation

Test relationships between words

- Analogies
- Clustering
- Visualization



Evaluating Word Embeddings

Extrinsic Evaluation

deeplearning.ai

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

Named entity

Andrew works at deeplearning.ai

Test word embeddings on external task

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

Extrinsic evaluation

Named entity

Andrew works at deeplearning.ai

person

Test word embeddings on external task

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

Extrinsic evaluation

Named entity

Andrew works at deeplearning.ai

person

Test word embeddings on external task

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

Extrinsic evaluation

Named entity

Andrew works at deeplearning.ai

person *organization*

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

\+ 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)
```