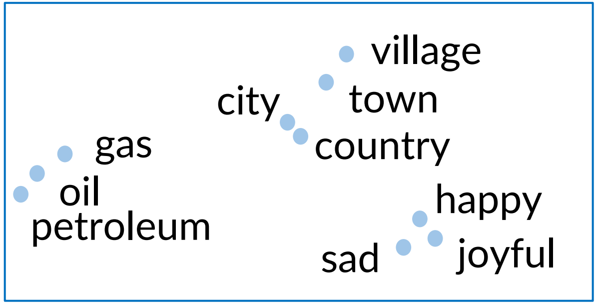
deeplearning.ai

Overview

Some basic applications of word embeddings

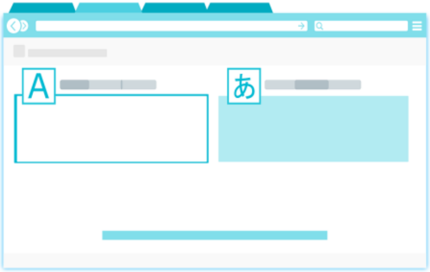


Semantic analogies and similarity

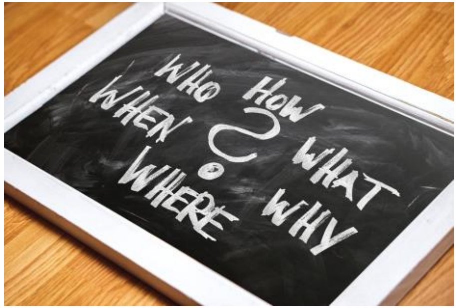
Sentiment analysis

Classification of customer feedback

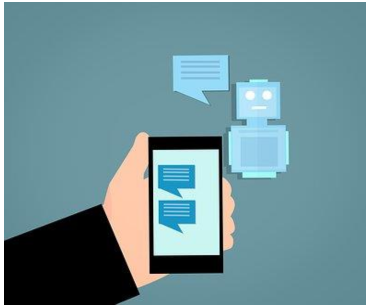
Advanced applications of word embeddings



Machine translation



Information extraction



Question answering

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

deeplearning.ai

Basic Word Representations

Outline

- Integers
- One-hot vectors
- Word embeddings

Integers

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

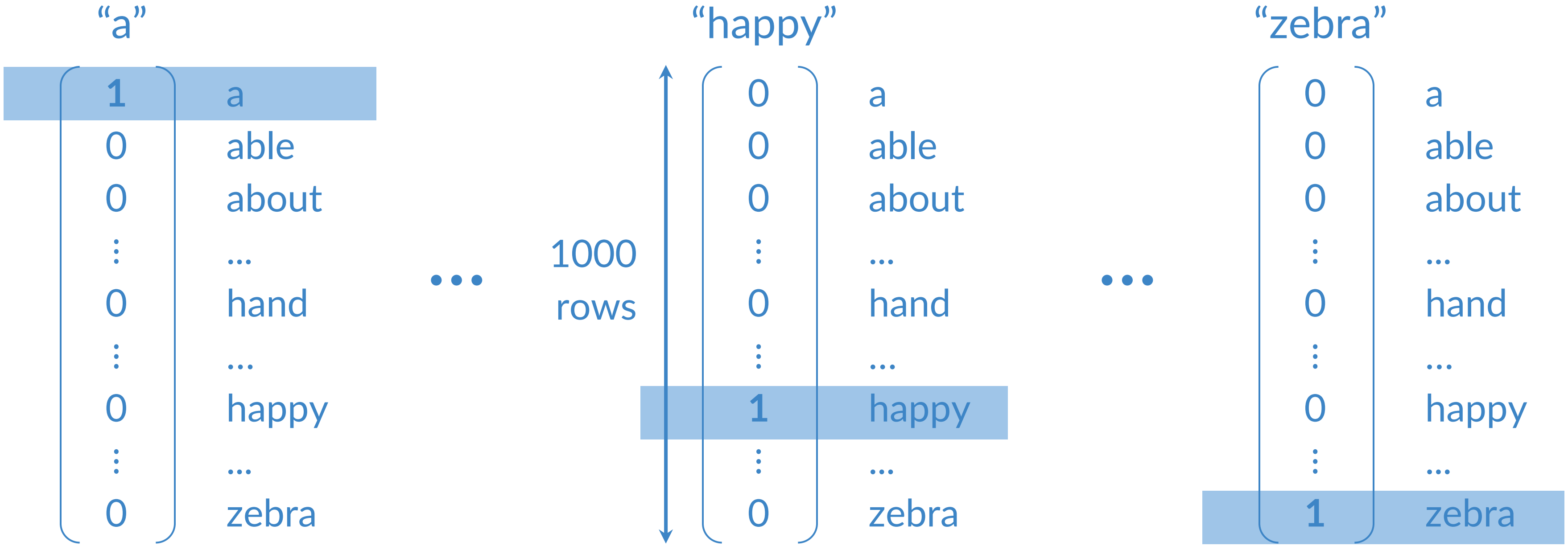
Integers

+ Simple

- Ordering: little semantic sense

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

One-hot vectors



“a”

1	a
0	able
0	about
⋮	…
0	hand
⋮	…
0	happy
⋮	…
0	zebra

• • •

“happy”

1000 rows

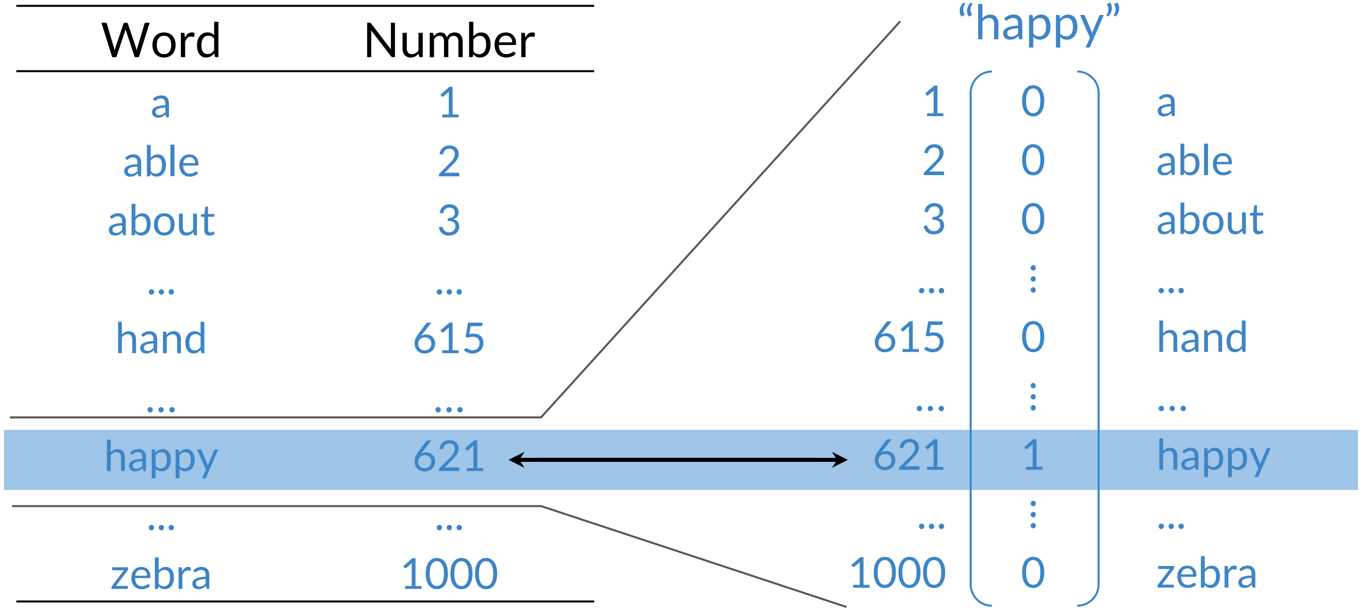
0	a
0	able
0	about
⋮	…
0	hand
⋮	…
1	happy
⋮	…
0	zebra

• • •

“zebra”

0	a
0	able
0	about
⋮	…
0	hand
⋮	…
0	happy
⋮	…
1	zebra

One-hot vectors



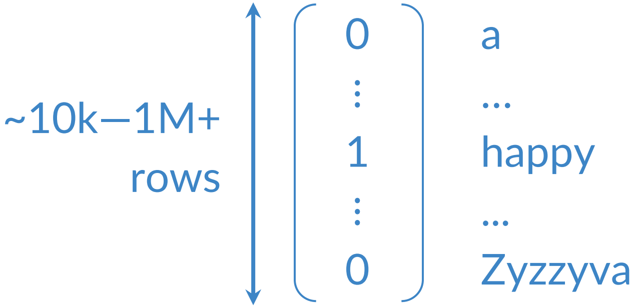
Word	Number
a	1
able	2
about	3
…	…
hand	615
…	…
happy	621
…	…
zebra	1000

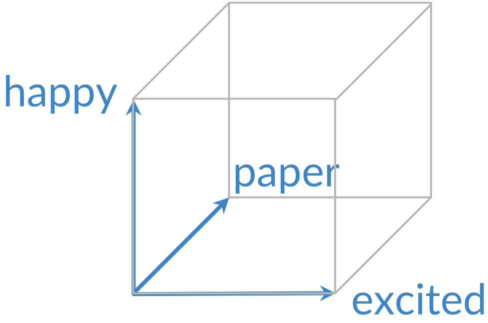
“happy”

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

One-hot vectors

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



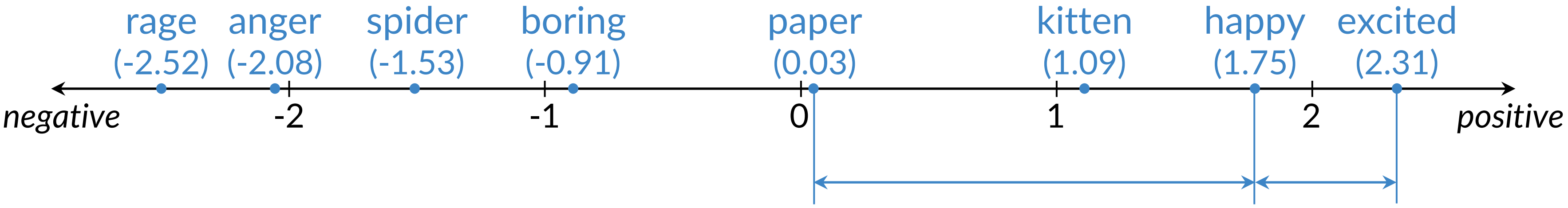


d(paper, excited)
= d(paper, happy)
= d(excited, happy)

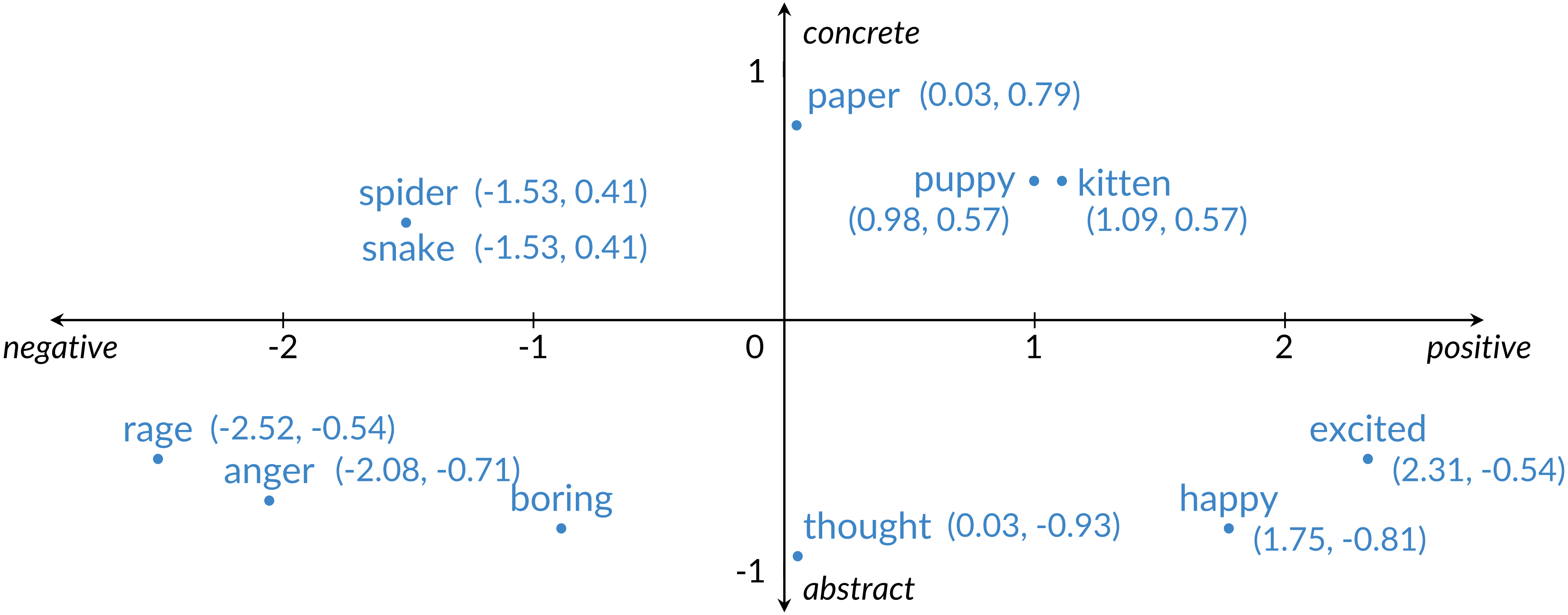
deeplearning.ai

Word Embeddings

Meaning as vectors



Meaning as vectors



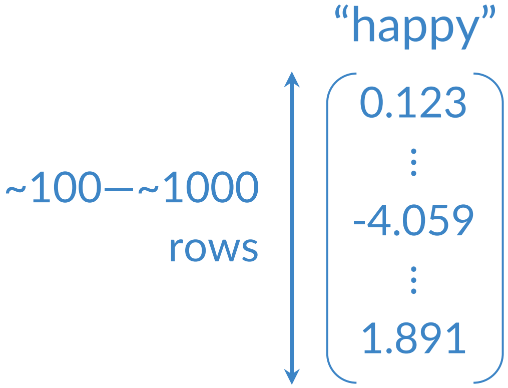
Word embedding vectors

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

forest ≈ tree   forest ≉ ticket

  - e.g. analogies

Paris:France :: Rome:?

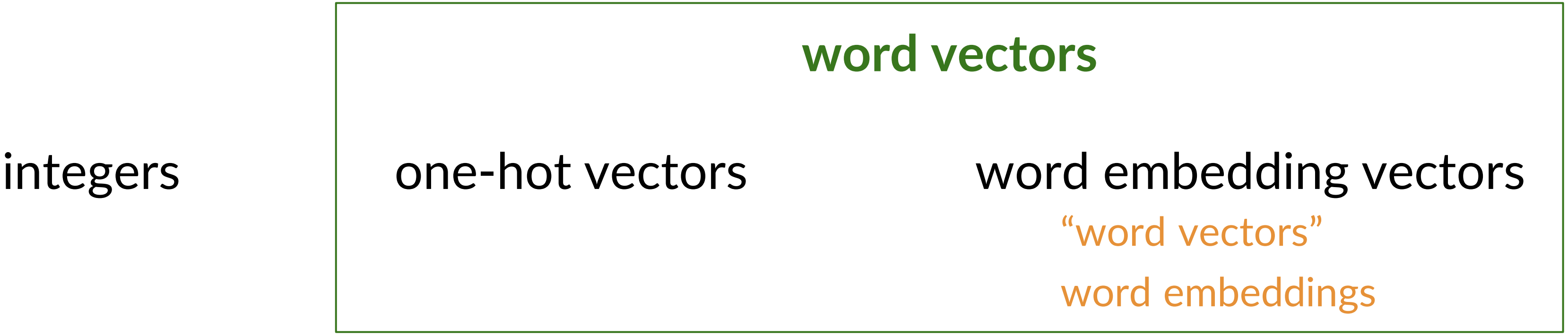


“happy”

~100—~1000 rows

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

Terminology



word vectors

integers

one-hot vectors

word embedding vectors

"word vectors"

word embeddings

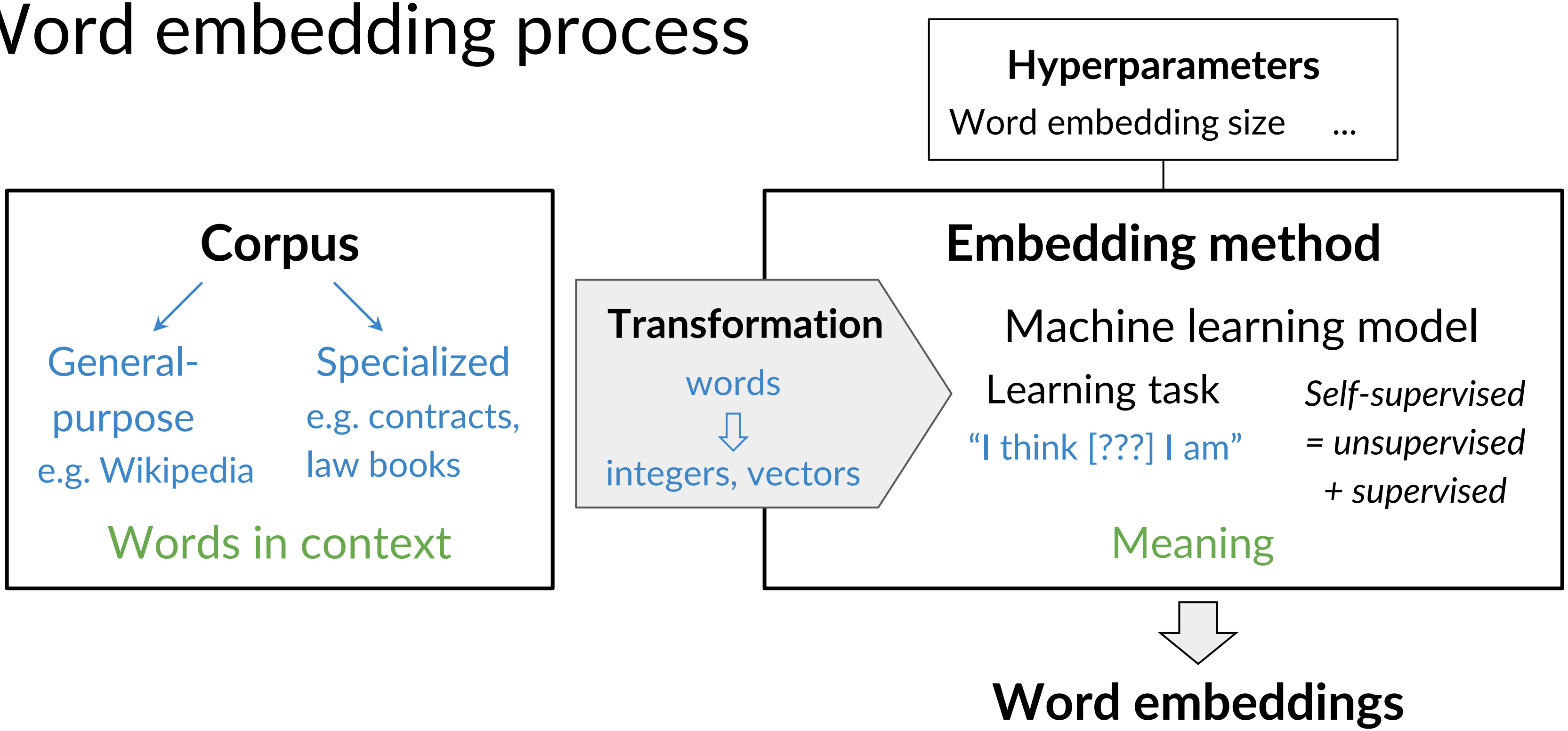
Summary

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

deeplearning.ai

How to Create Word Embeddings

Word embedding process



Hyperparameters

Word embedding size ...

Corpus

General-purpose

e.g. Wikipedia

Specialized

e.g. contracts, law books

Words in context

Transformation

words

⇩

integers, vectors

Embedding method

Machine learning model

Learning task

"I think [???] I am"

Self-supervised

= unsupervised

+ supervised

Meaning

Word embeddings

deeplearning.ai

Word Embedding Methods

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

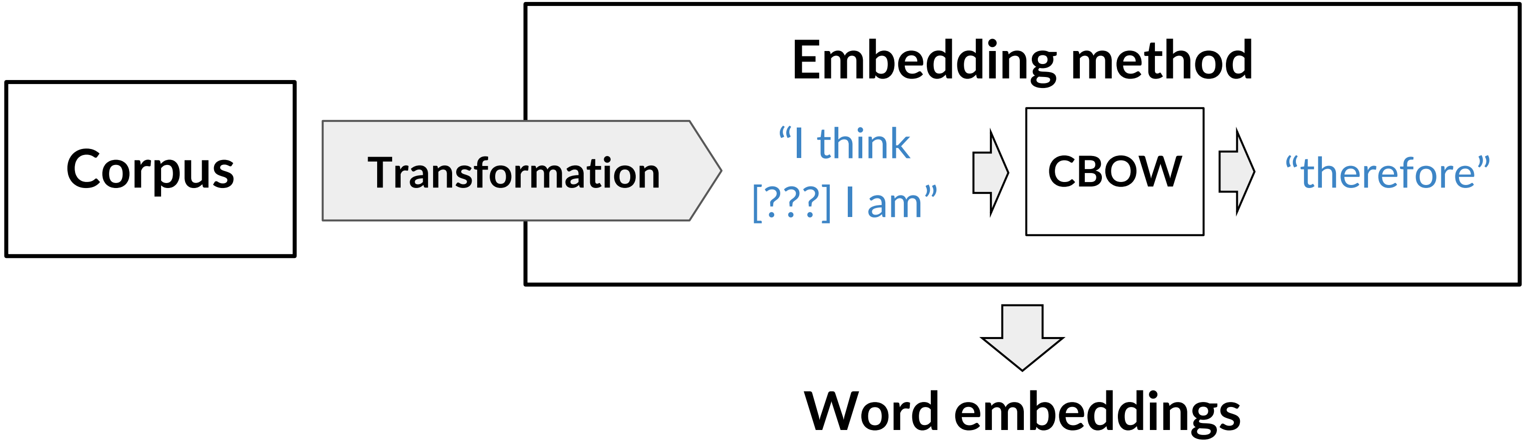
- BERT (Google, 2018)
- ELMo (Allen Institute for AI, 2018)
- GPT-2 (OpenAI, 2018)

Tunable pre-trained models available

deeplearning.ai

Continuous Bag-of-Words Model

Continuous bag-of-words word embedding process



Corpus

Transformation

Embedding method

"I think [???] I am"

CBOW

"therefore"

Word embeddings

Center word prediction: rationale

The little ____?____ is barking

dog

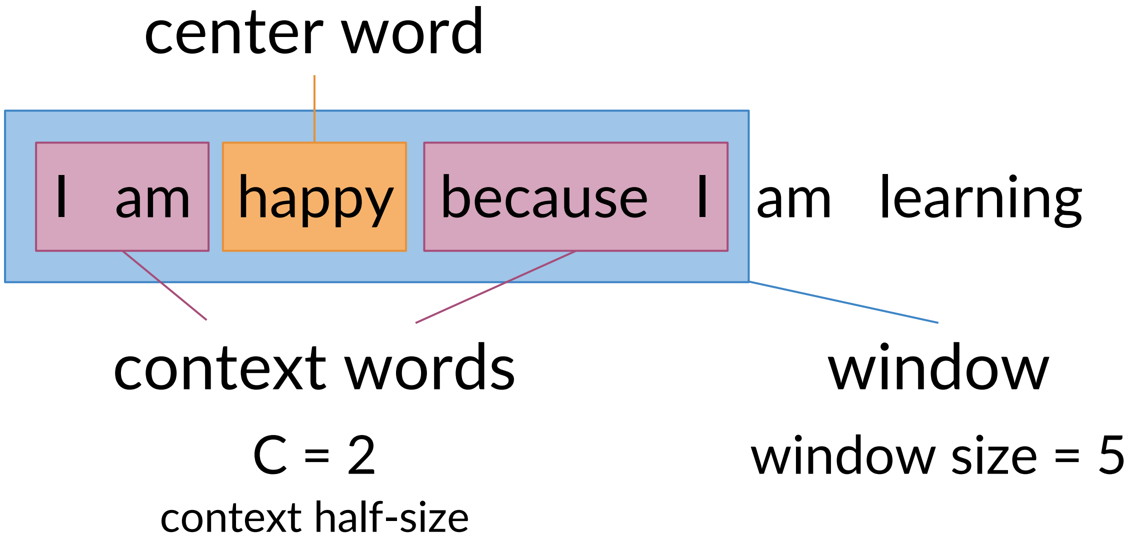
puppy

hound

terrier

…

Creating a training example



center word

I am happy because I am learning

context words

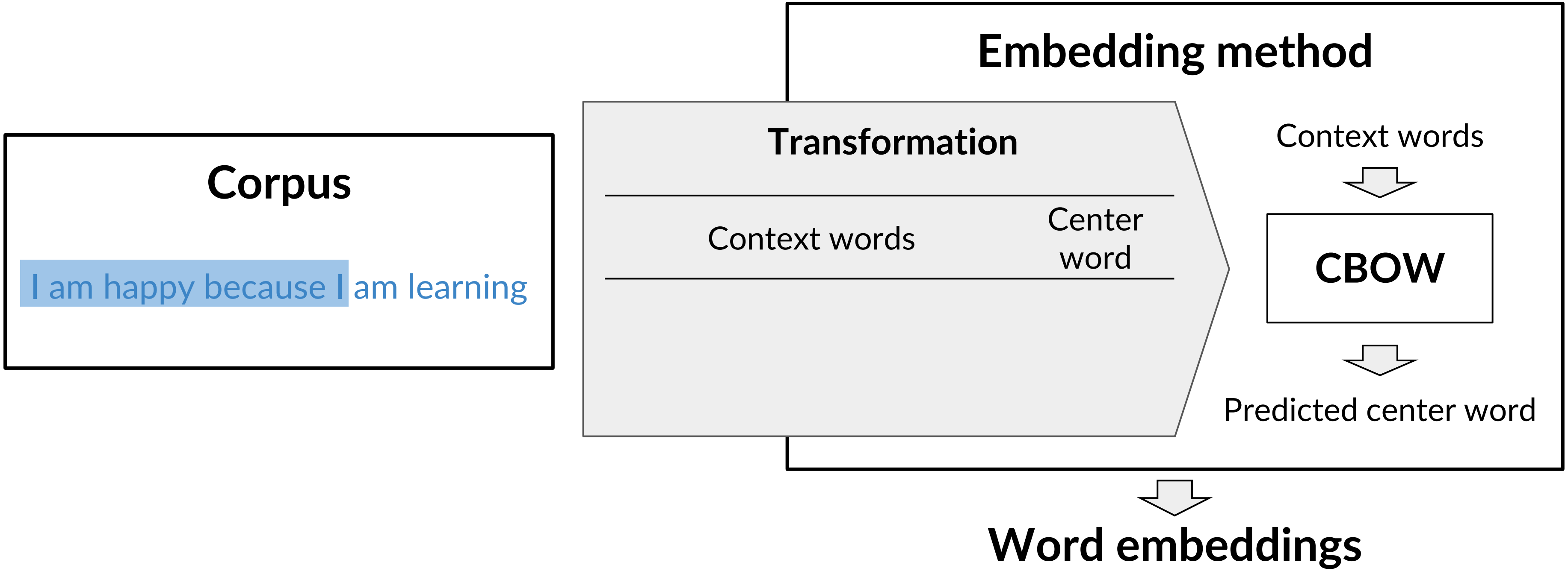
C = 2

context half-size

window

window size = 5

From corpus to training



Corpus

I am happy because I am learning

Embedding method

Transformation

Context words	Center word

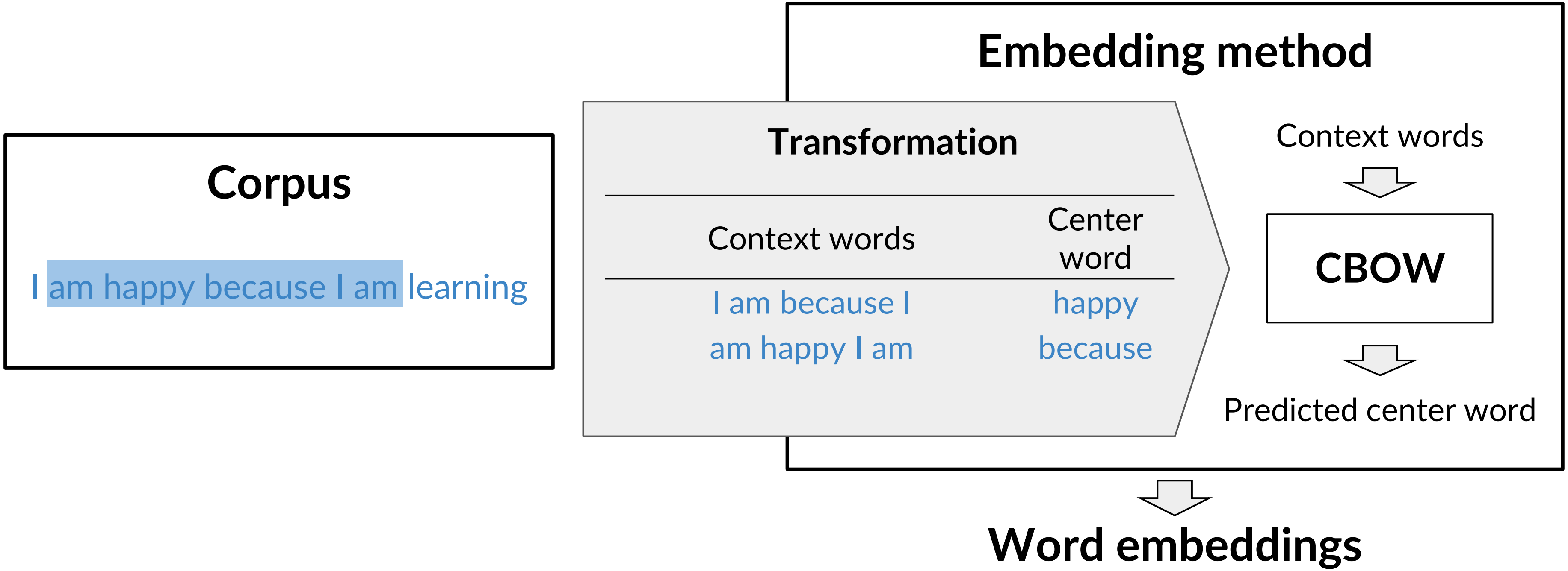
Context words

CBOW

Predicted center word

Word embeddings

From corpus to training



Corpus

I am happy because I am learning

Transformation

Context words	Center word
I am because I	happy
am happy I am	because

Embedding method

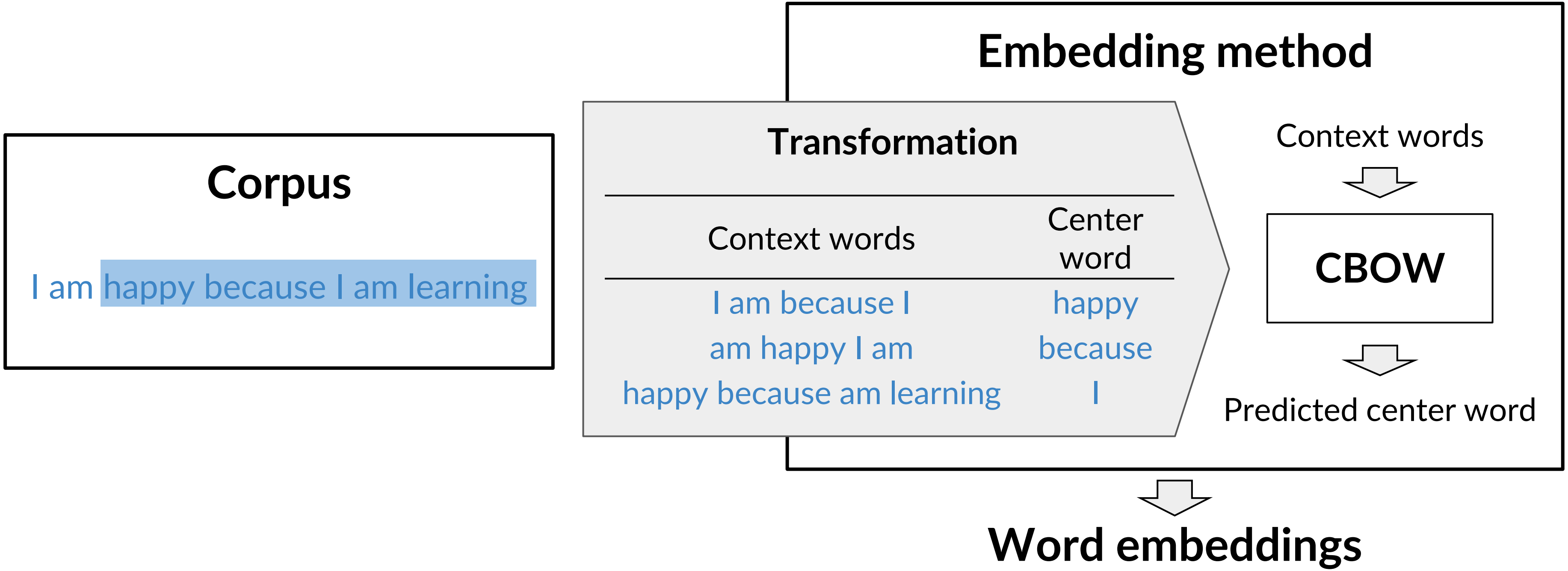
Context words

CBOW

Predicted center word

Word embeddings

From corpus to training



Corpus

I am happy because I am learning

Transformation

Context words	Center word
I am because I	happy
am happy I am	because
happy because am learning	I

Embedding method

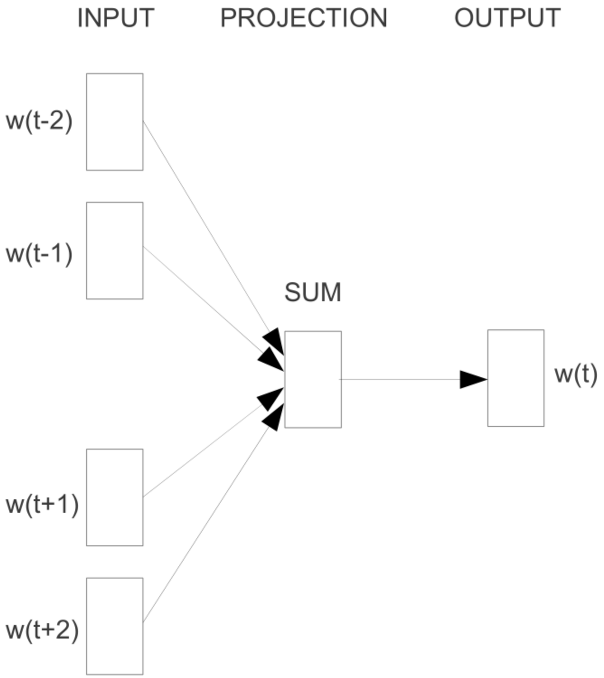
Context words

CBOW

Predicted center word

Word embeddings

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

- 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   , ! . ? → .   “ ‘ « » ’ ” → ∅   ... !! ??? → .
- 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 ∇ $ € § ¶ ** → ∅
- Special words 😊 #nlp → :happy: #nlp

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)
```

```
→ 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

```python
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)
       ]
```

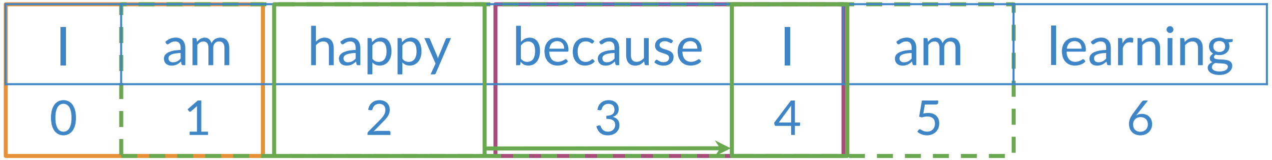
```
→ ['who', '♡', 'word', 'embeddings', 'in', '.', 'i', 'do', '.']
```

deeplearning.ai

Sliding Window of Words in Python

Sliding window of words in Python

```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):
    ...
        yield context_words, center_word
```

```python
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

Vocabulary am, because, happy, I, learning

One-hot vector

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

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]

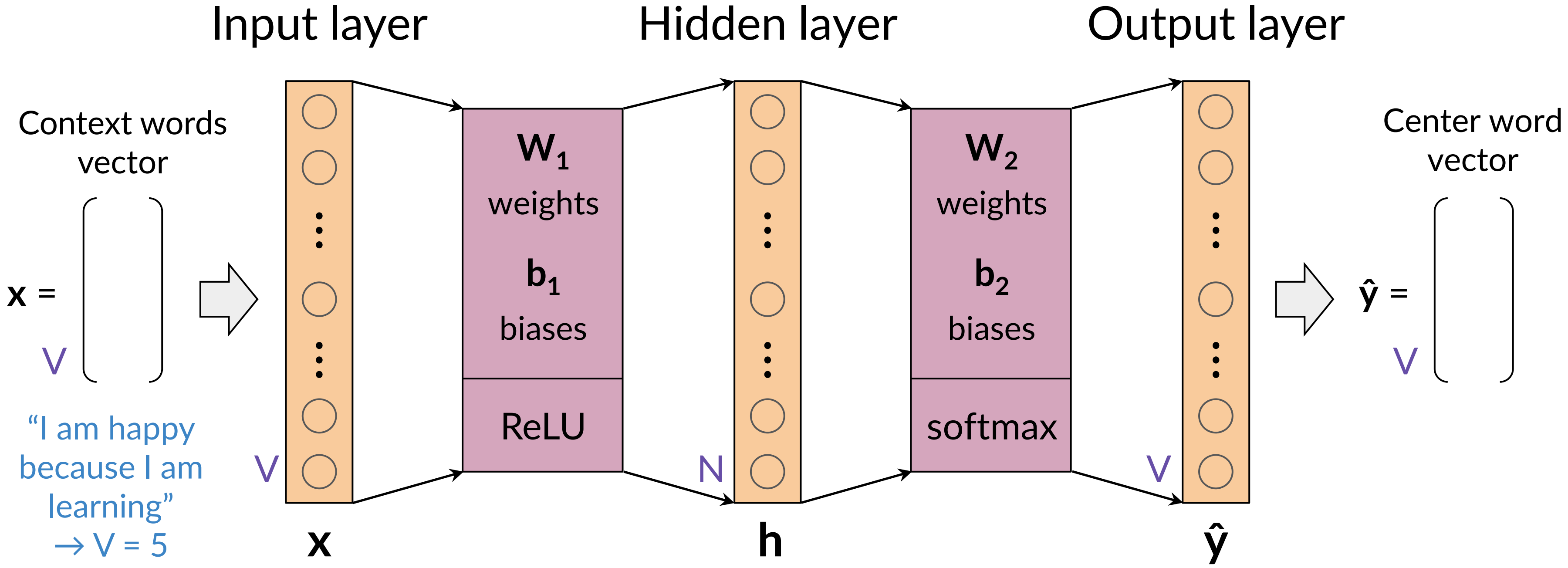
deeplearning.ai

Architecture of the CBOW Model

Architecture of the CBOW model

Hyperparameters

N: Word embedding size ...



Input layer

Hidden layer

Output layer

Context words vector

x = V

"I am happy because I am learning"

→ V = 5

V

x

W_1 weights

b_1 biases

ReLU

N

h

W_2 weights

b_2 biases

softmax

V

ŷ

Center word vector

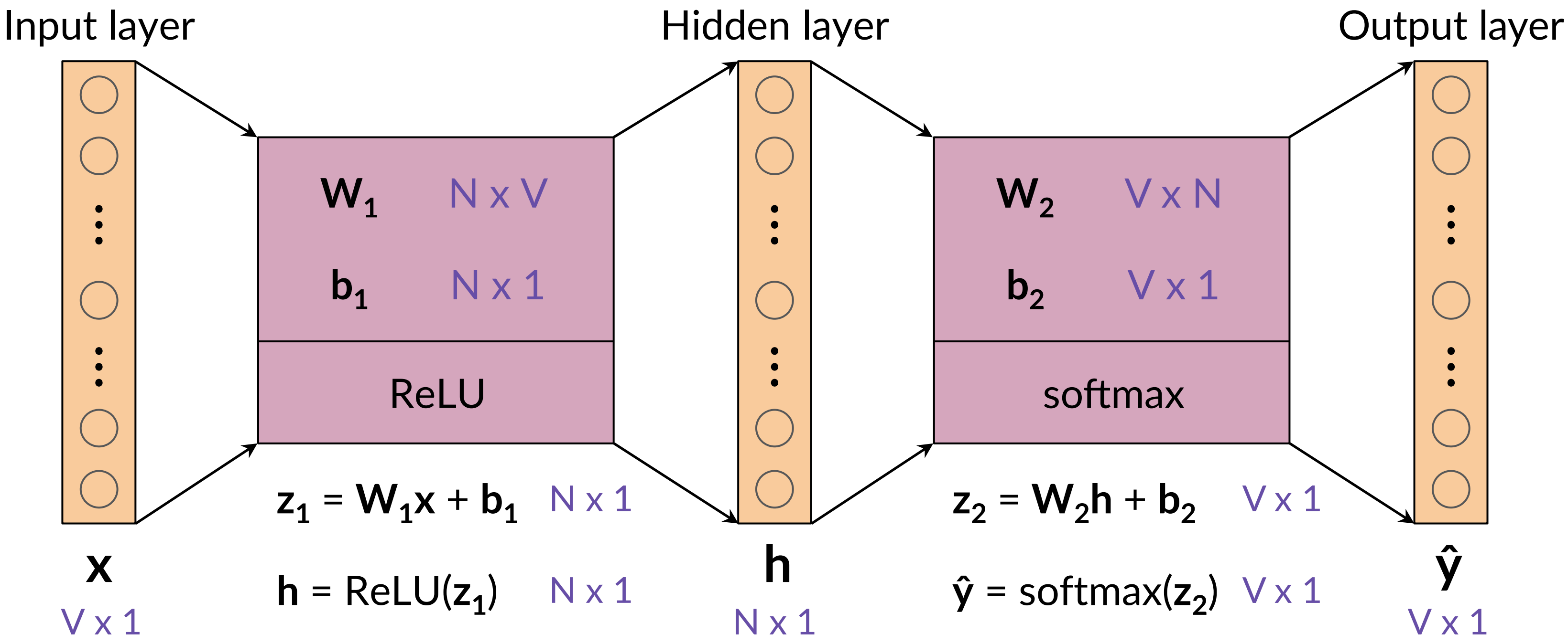
ŷ = V

deeplearning.ai

Architecture of the CBOW Model:

Dimensions

Dimensions (single input)



Input layer

Hidden layer

Output layer

$\mathbf{W_1}$ N x V

$\mathbf{b_1}$ N x 1

ReLU

$\mathbf{W_2}$ V x N

$\mathbf{b_2}$ V x 1

softmax

$\mathbf{z_1} = \mathbf{W_1 x} + \mathbf{b_1}$ N x 1

$\mathbf{h} = \text{ReLU}(\mathbf{z_1})$ N x 1

$\mathbf{z_2} = \mathbf{W_2 h} + \mathbf{b_2}$ V x 1

$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z_2})$ V x 1

$\mathbf{x}$ V x 1

$\mathbf{h}$ N x 1

$\hat{\mathbf{y}}$ V x 1

Dimensions (single input)

Column vectors

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

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

Row vectors

$$\mathbf{z_1} = \mathbf{xW_1}^T + \mathbf{b_1}$$

$$\mathbf{b_1} = \begin{bmatrix} \text{1 x N} \end{bmatrix} \quad \mathbf{W_1} = \begin{pmatrix} \text{N x V} \end{pmatrix} \quad \mathbf{b_1} = \begin{bmatrix} \text{1 x N} \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} \text{1 x V} \end{bmatrix}$$

deeplearning.ai

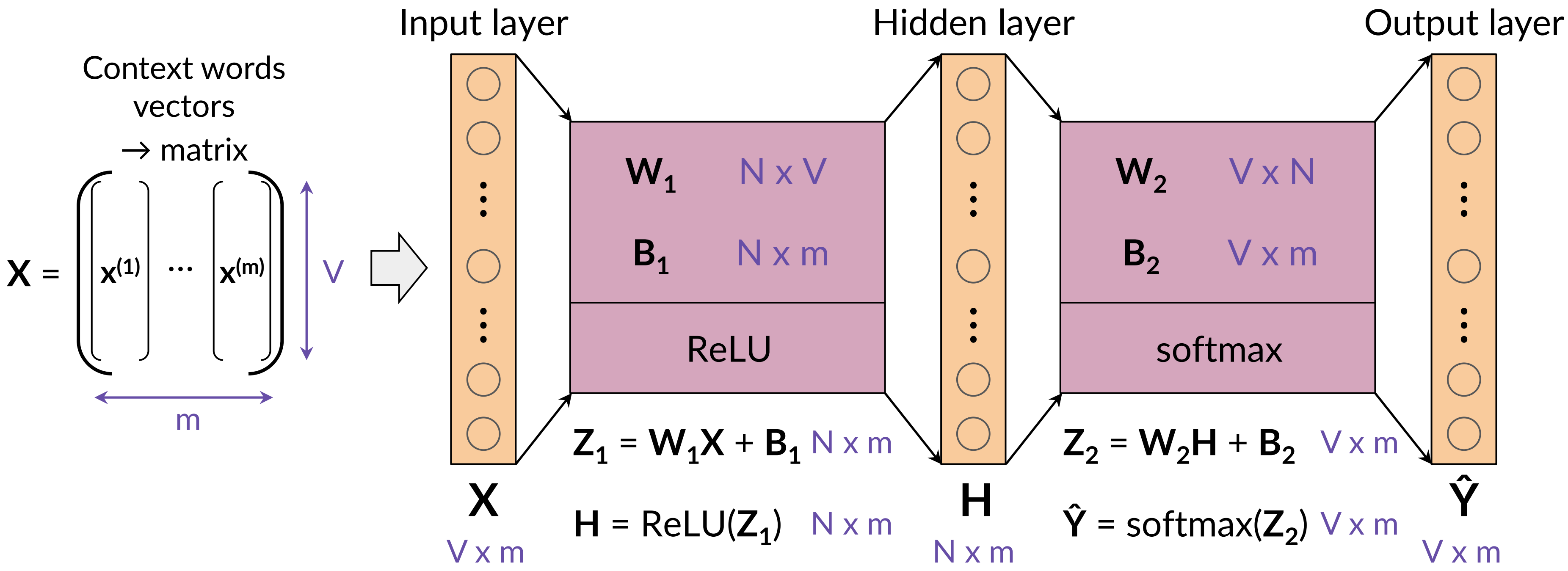
Architecture of the CBOW Model:

Dimensions 2

Dimensions (batch input)

$$\begin{bmatrix} b_1 \end{bmatrix} \rightarrow \mathbf{B}_1 = \underbrace{\left(\begin{bmatrix} b_1 \end{bmatrix} \cdots \begin{bmatrix} b_1 \end{bmatrix}\right)}_{m} \updownarrow N$$

broadcasting



Context words vectors
→ matrix

$$\mathbf{X} = \underbrace{\begin{pmatrix} \mathbf{x}^{(1)} & \cdots & \mathbf{x}^{(m)} \end{pmatrix}}_{m} \updownarrow V$$

Input layer

X

V x m

W_1 N x V

B_1 N x m

ReLU

$\mathbf{Z_1} = \mathbf{W_1 X} + \mathbf{B_1}$ N x m

$\mathbf{H} = \mathrm{ReLU}(\mathbf{Z_1})$ N x m

Hidden layer

H

N x m

W_2 V x N

B_2 V x m

softmax

$\mathbf{Z_2} = \mathbf{W_2 H} + \mathbf{B_2}$ V x m

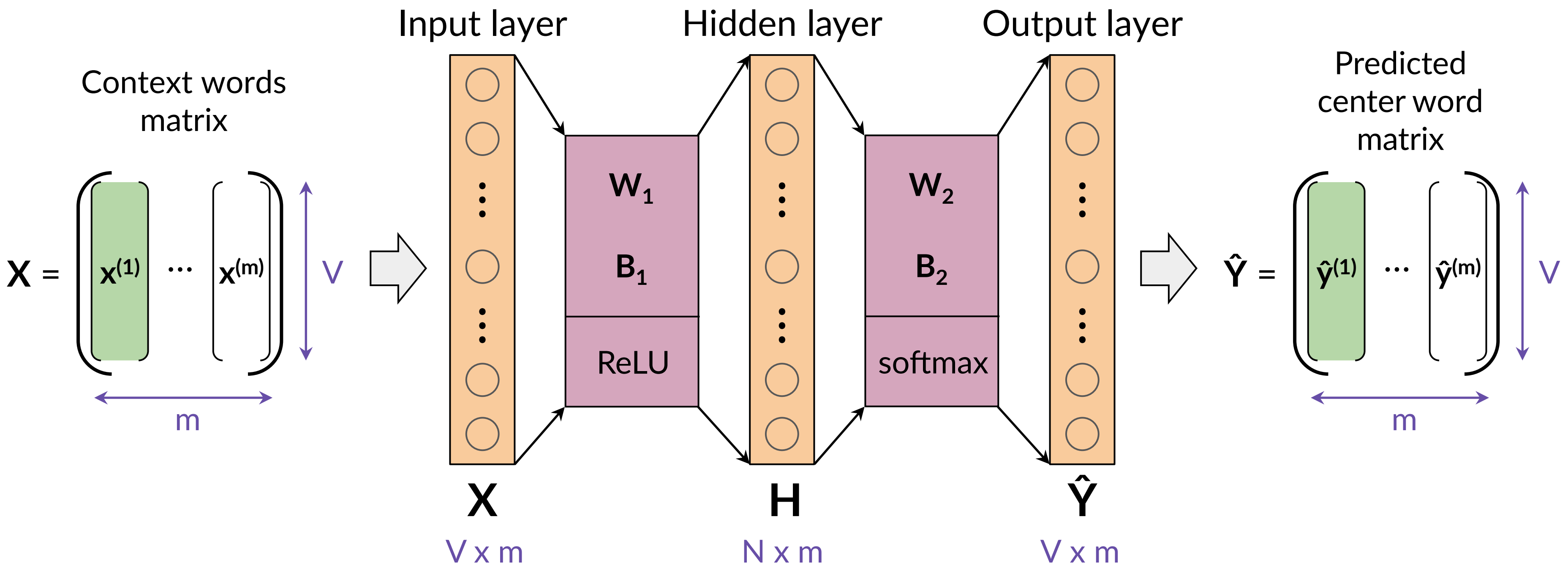
$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathbf{Z_2})$ V x m

Output layer

$\hat{Y}$

V x m

Dimensions (batch input)

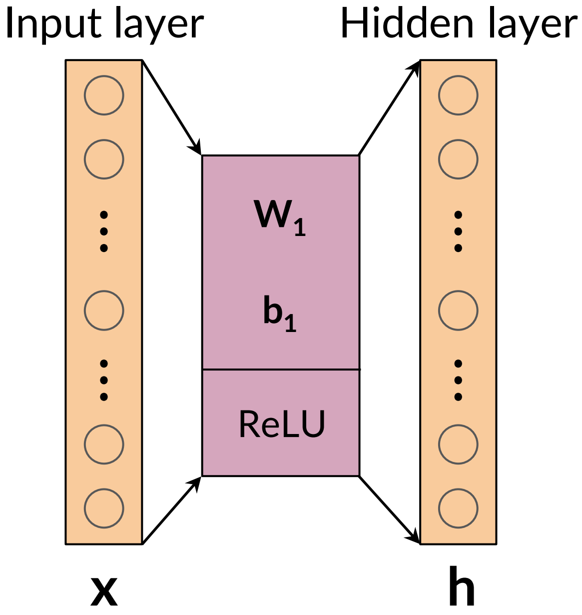


deeplearning.ai

Architecture of the CBOW Model

Activation Functions

Rectified Linear Unit (ReLU)



Input layer

Hidden layer

$\mathbf{W_1}$

$\mathbf{b_1}$

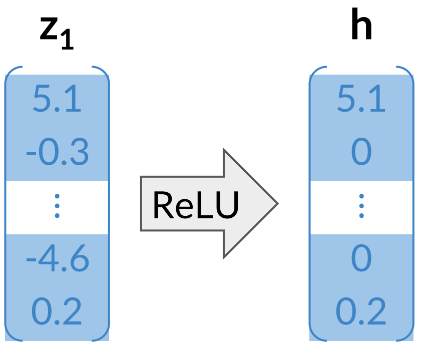
ReLU

$\mathbf{x}$

$\mathbf{h}$

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

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



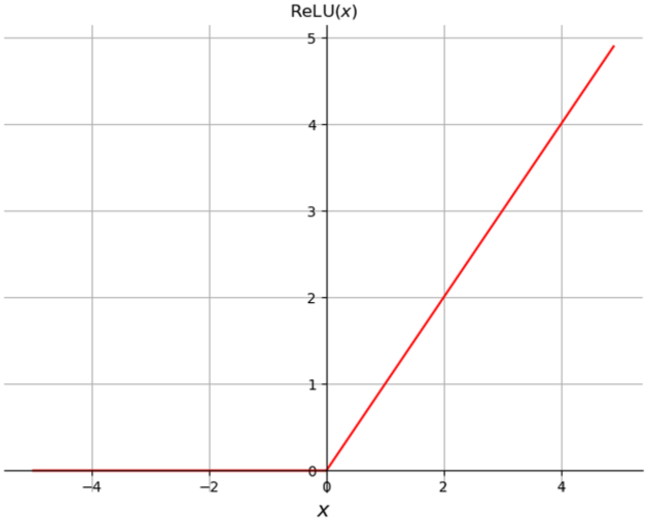
$\mathbf{z_1}$

5.1	5.1
-0.3	0
⋮	⋮
-4.6	0
0.2	0.2

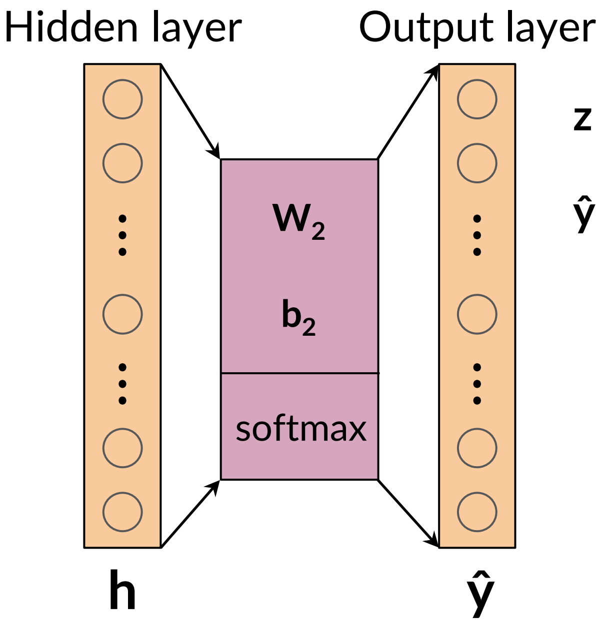
ReLU

$\mathbf{h}$

ReLU(x) = max(0, x)



Softmax



Hidden layer

Output layer

$\mathbf{W_2}$

$\mathbf{b_2}$

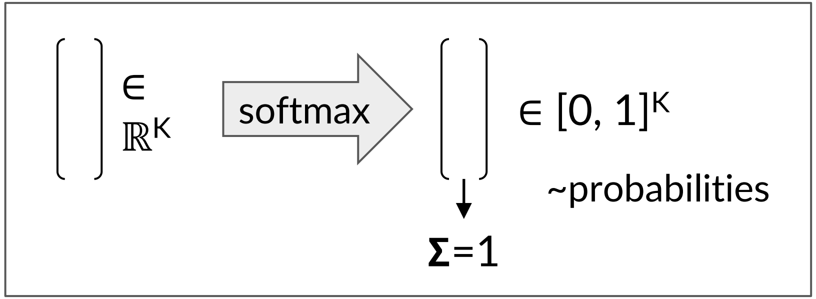
softmax

$\mathbf{h}$

$\hat{\mathbf{y}}$

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

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



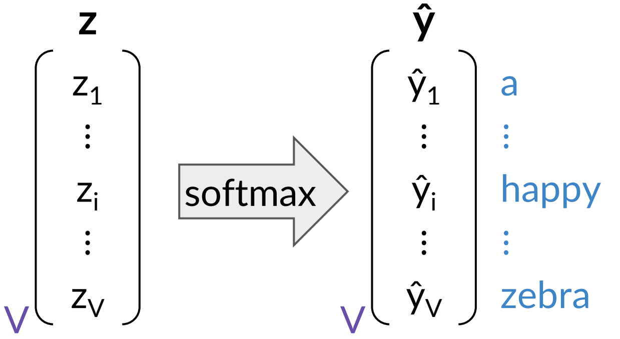
$\in \mathbb{R}^K$

softmax

$\in [0, 1]^K$

~probabilities

$\Sigma=1$



$\mathbf{z}$

z_1, ⋮, z_i, ⋮, z_V

V

softmax

$\hat{\mathbf{y}}$

$\hat{y}_1$, ⋮, $\hat{y}_i$, ⋮, $\hat{y}_V$

V

a

⋮

happy

⋮

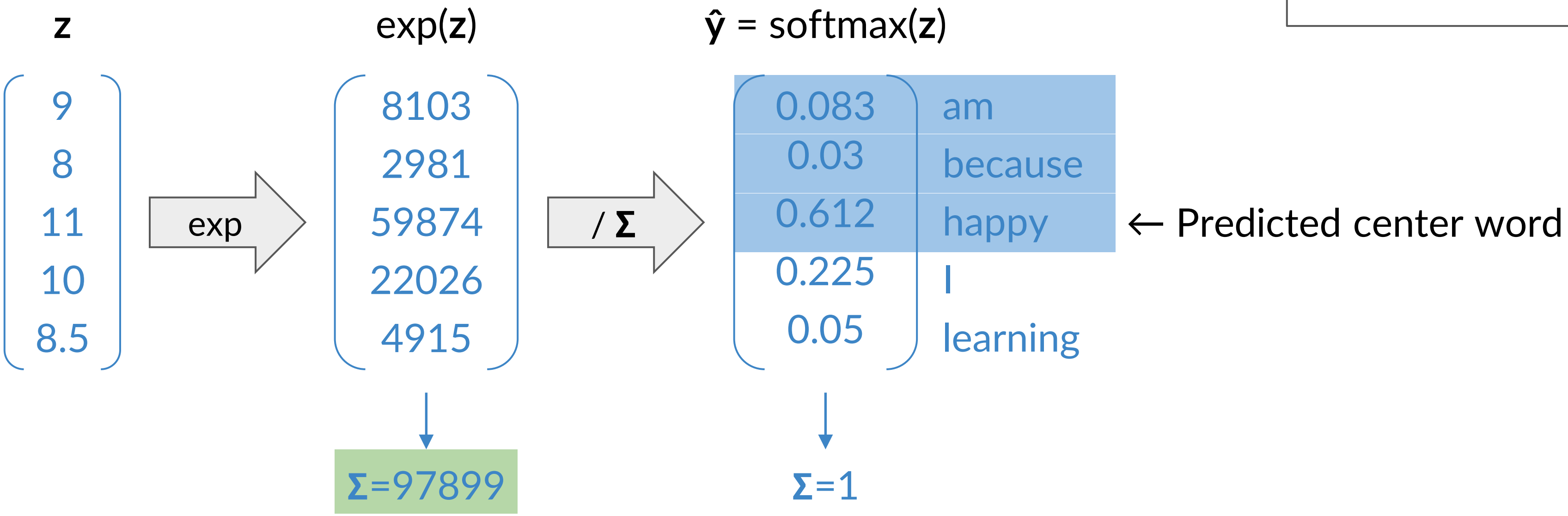
zebra

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

Probabilities of
being center word

Softmax: example

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



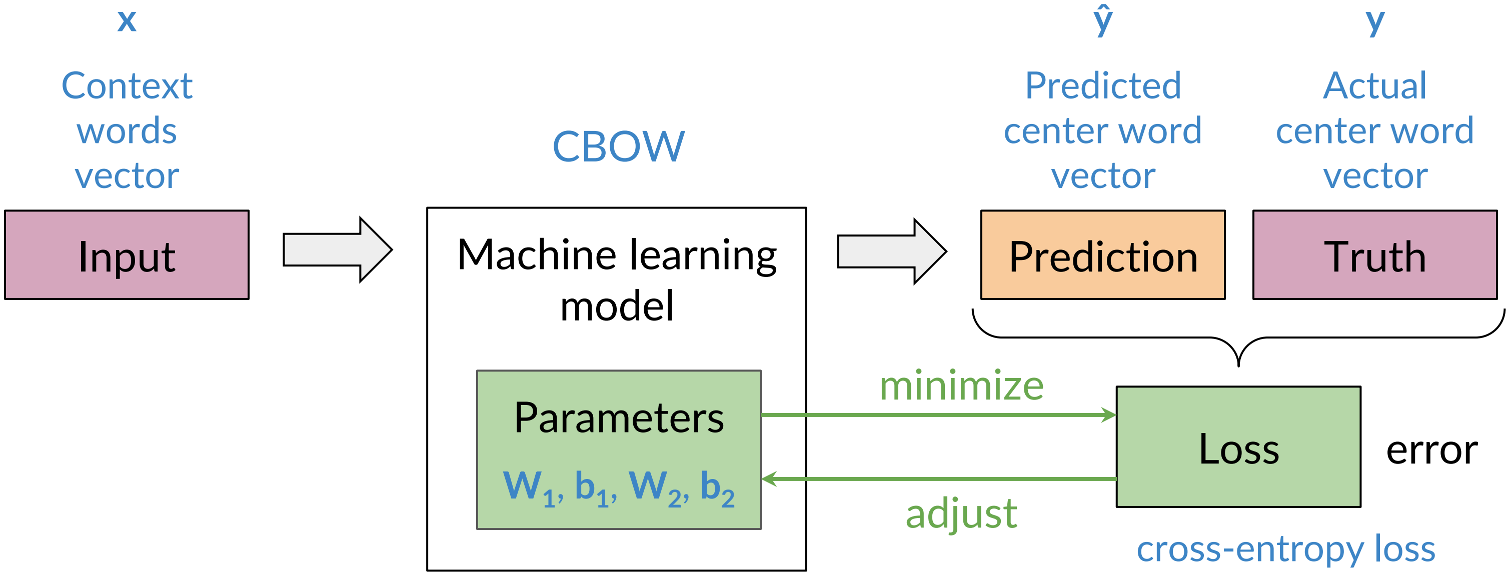
z	exp(z)	ŷ = softmax(z)	
9	8103	0.083	am
8	2981	0.03	because
11	59874	0.612	happy
10	22026	0.225	I
8.5	4915	0.05	learning

deeplearning.ai

Training a CBOW Model

Cost Function

Loss



$\mathbf{x}$

Context words vector

Input

CBOW

Machine learning model

Parameters

W_1, b_1, W_2, b_2

$\hat{\mathbf{y}}$

Predicted center word vector

Prediction

$\mathbf{y}$

Actual center word vector

Truth

minimize

adjust

Loss

error

cross-entropy loss

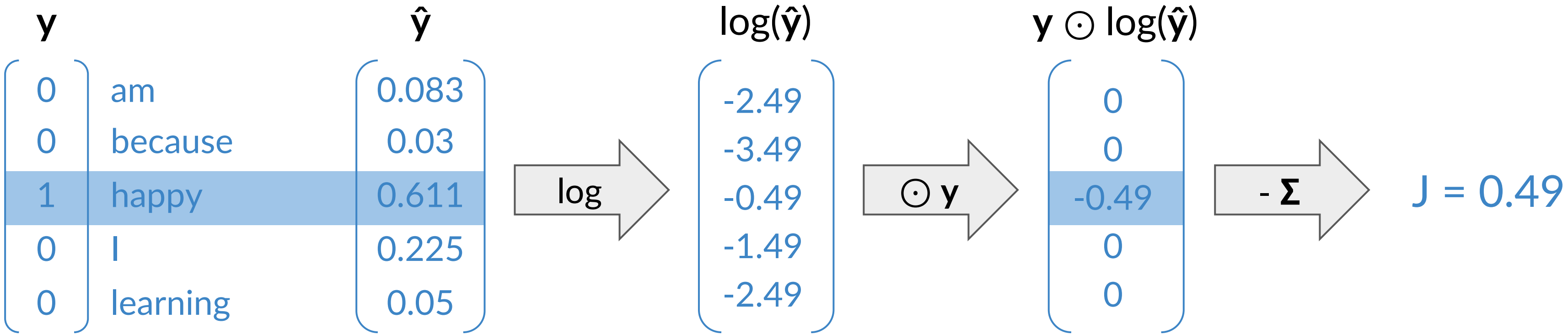
Cross-entropy loss

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

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

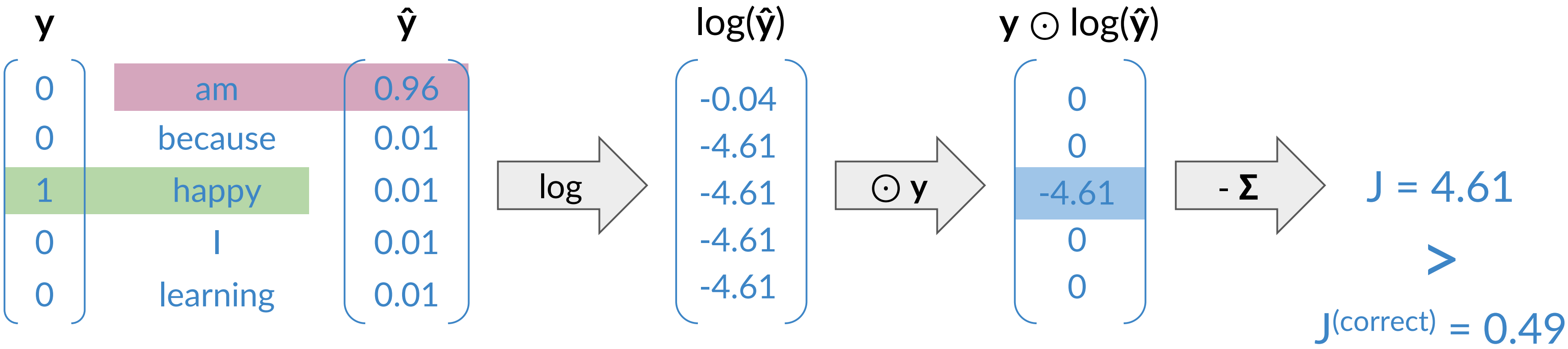


y		ŷ	log(ŷ)	y ⊙ log(ŷ)
0	am	0.083	-2.49	0
0	because	0.03	-3.49	0
1	happy	0.611	-0.49	-0.49
0	I	0.225	-1.49	0
0	learning	0.05	-2.49	0

J = 0.49

Cross-entropy loss

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



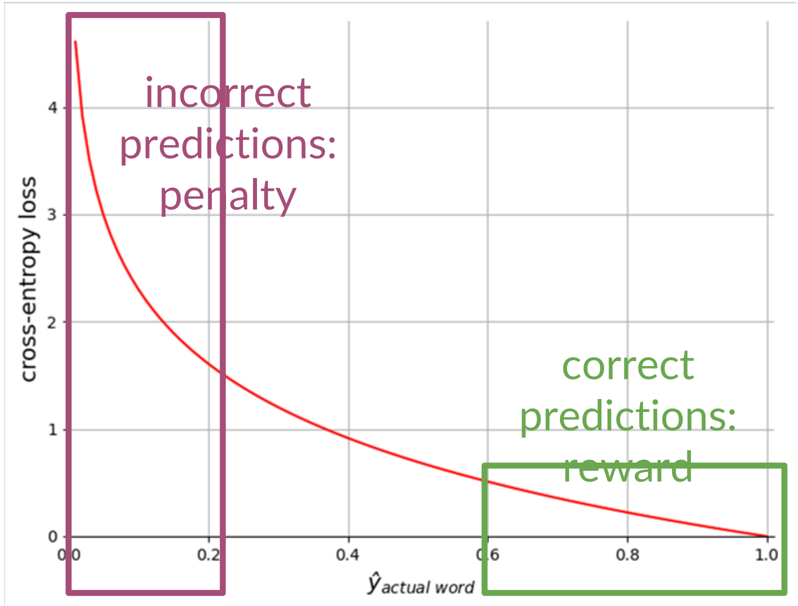
Cross-entropy loss

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

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

y		ŷ
0	am	0.96
0	because	0.01
1	happy	0.01
0	I	0.01
0	learning	0.01

→ $J = 4.61$



incorrect predictions: penalty

correct predictions: reward

deeplearning.ai

Training a CBOW Model

Forward Propagation

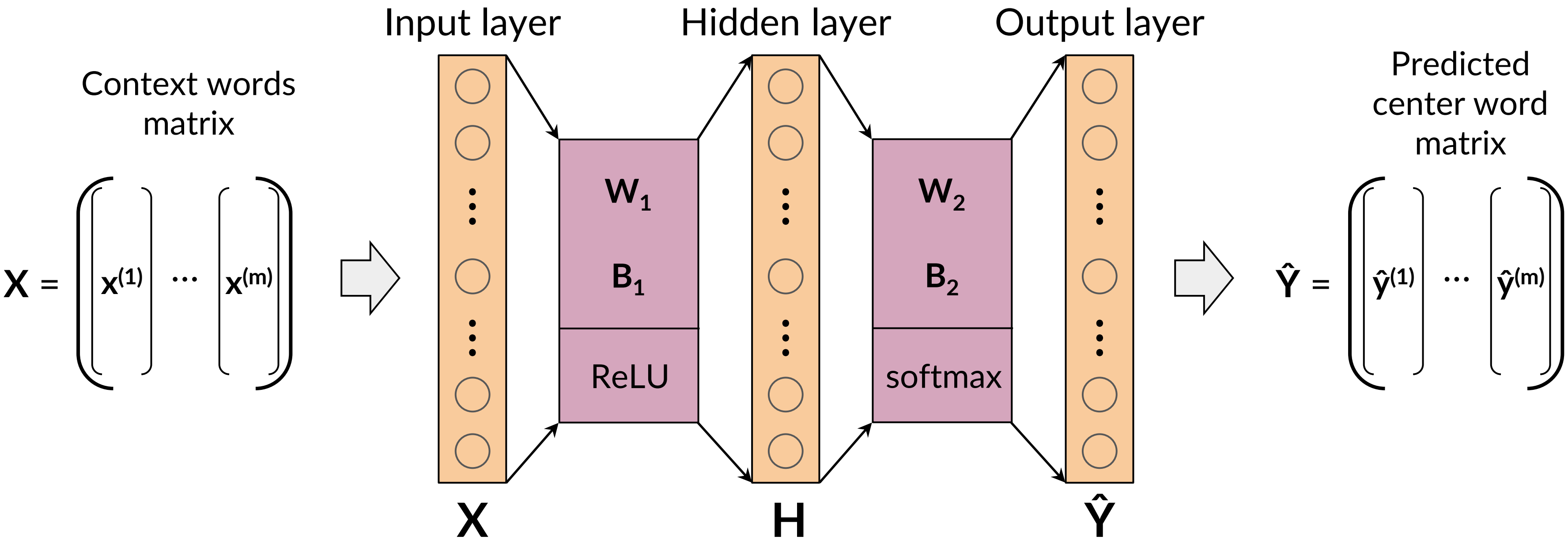
Training process

- Forward propagation
- Cost
- Backpropagation and gradient descent

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: 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

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

Actual center word matrix

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

deeplearning.ai

Training a CBOW Model

Backpropagation and Gradient Descent

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

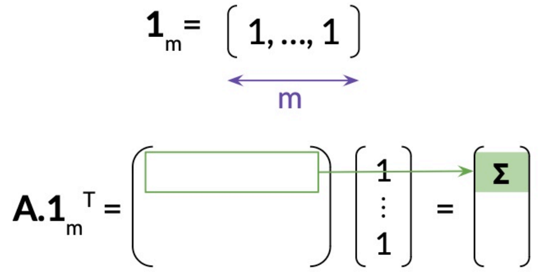
Backpropagation

$$\frac{\partial J_{batch}}{\partial \mathbf{W_1}} = \frac{1}{m}(\mathbf{W_2}^\intercal(\hat{\mathbf{Y}} - \mathbf{Y}) \cdot \mathrm{step}(\mathbf{Z_1}))\mathbf{X}^\intercal$$

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

$$\frac{\partial J_{batch}}{\partial \mathbf{b_1}} = \frac{1}{m}(\mathbf{W_2}^\intercal(\hat{\mathbf{Y}} - \mathbf{Y}) \cdot \mathrm{step}(\mathbf{Z_1}))\mathbf{1}_m^\intercal$$

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



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

Gradient descent

Hyperparameter: learning rate α

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

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

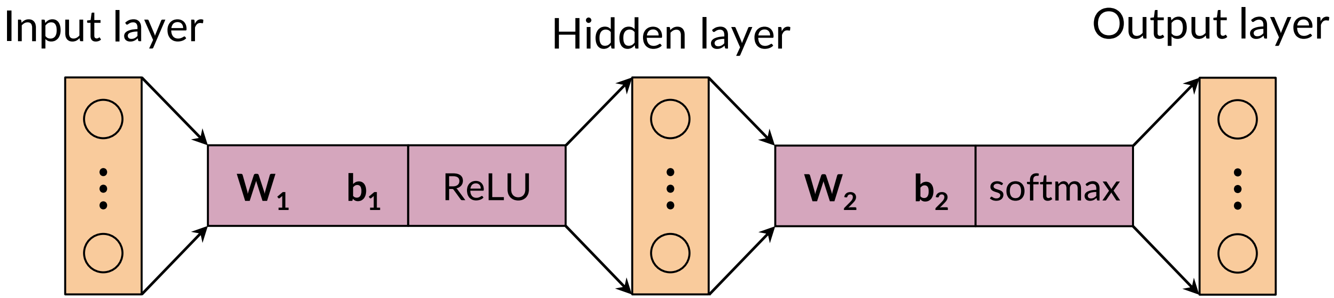
$$\mathbf{b_1} := \mathbf{b_1} - \alpha\frac{\partial J_{batch}}{\partial \mathbf{b_1}}$$

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

deeplearning.ai

Extracting Word Embedding Vectors

Extracting word embedding vectors: option 1



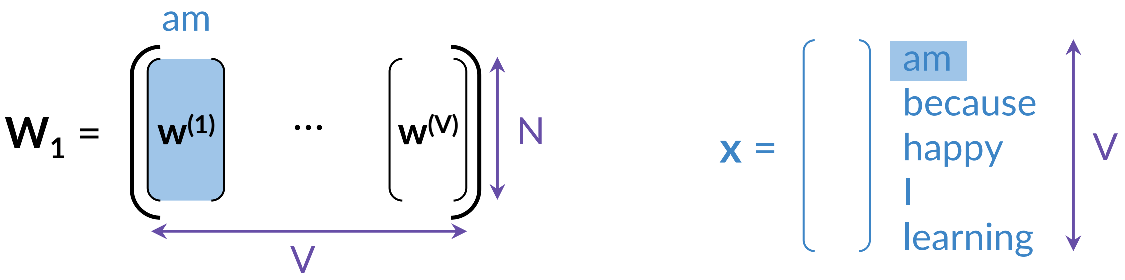
Input layer

Hidden layer

Output layer

W_1 b_1 ReLU

W_2 b_2 softmax



am

$W_1 = \begin{pmatrix} w^{(1)} & \cdots & w^{(V)} \end{pmatrix}$

N

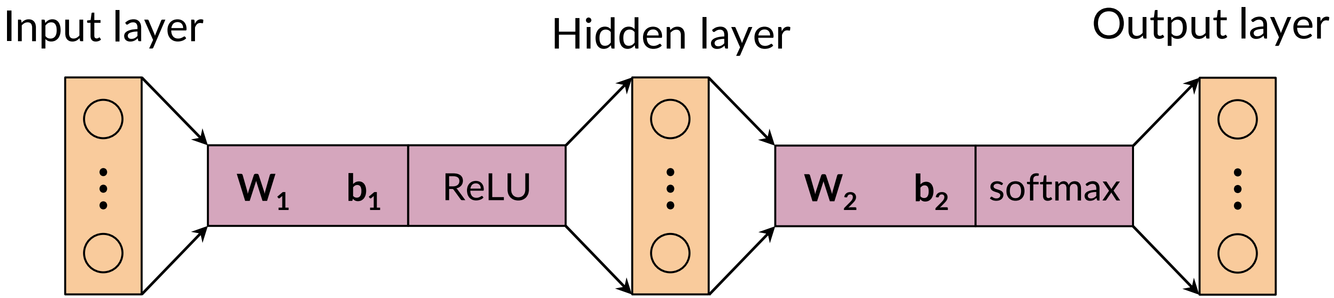
V

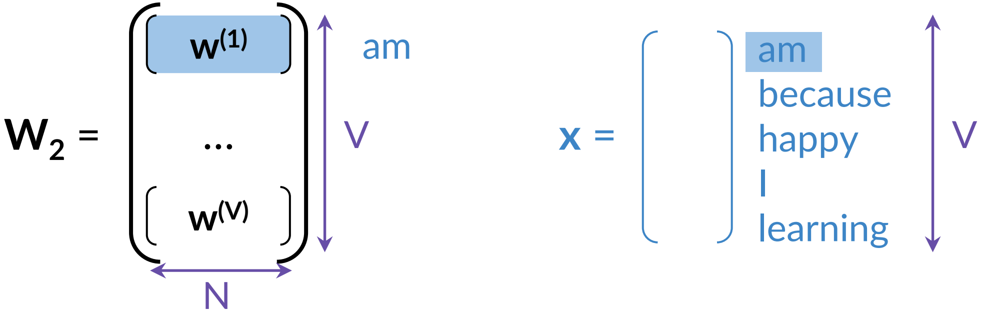
$x = \begin{pmatrix} \\ \end{pmatrix}$

am
because
happy
I
learning

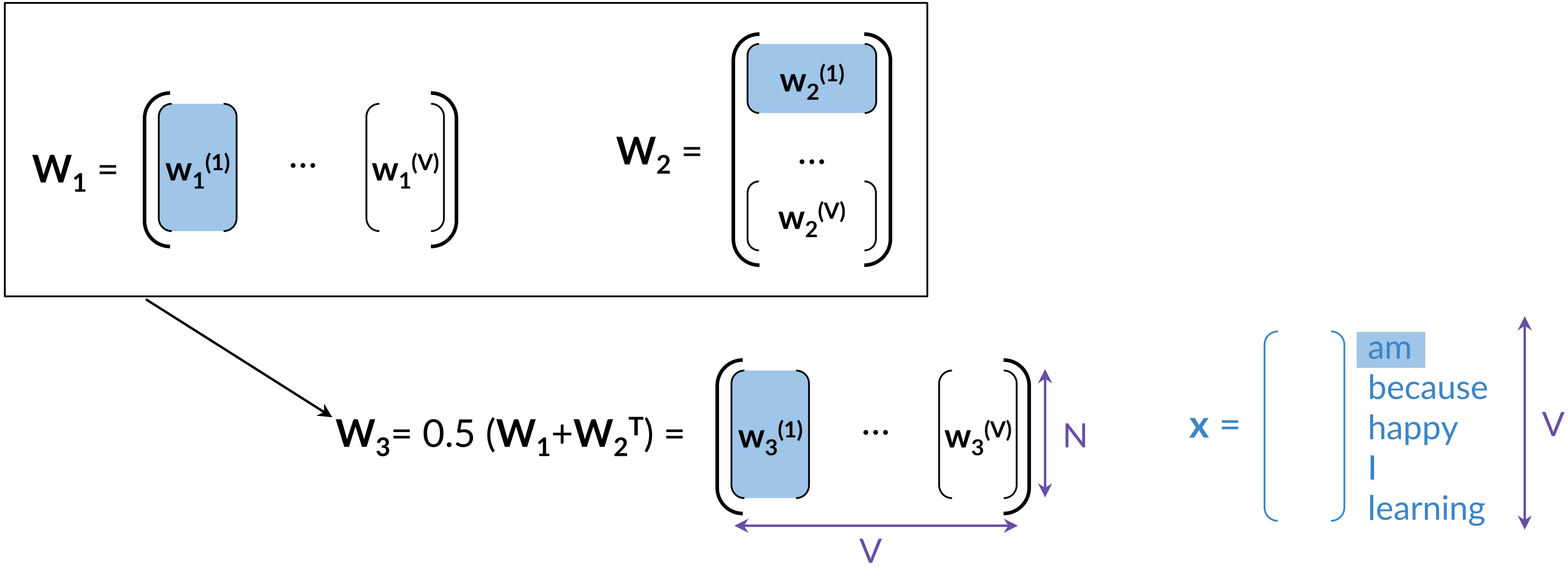
V

Extracting word embedding vectors: option 2





Extracting word embedding vectors: option 3



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

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

$$\mathbf{W_3} = 0.5\ (\mathbf{W_1} + \mathbf{W_2}^{\mathrm{T}}) = \begin{pmatrix} \mathbf{w_3}^{(1)} & \cdots & \mathbf{w_3}^{(V)} \end{pmatrix}$$

$\mathbf{x} =$

deeplearning.ai

Evaluating Word Embeddings

Intrinsic Evaluation

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 <?>

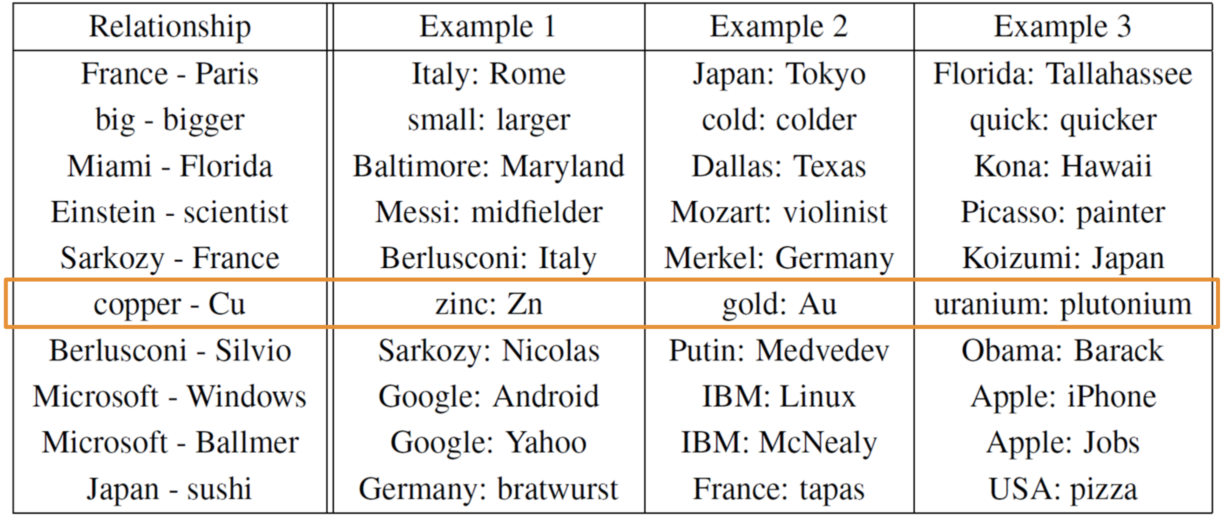
ϟ Ambiguity

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

Intrinsic evaluation

Test relationships between words

- Analogies



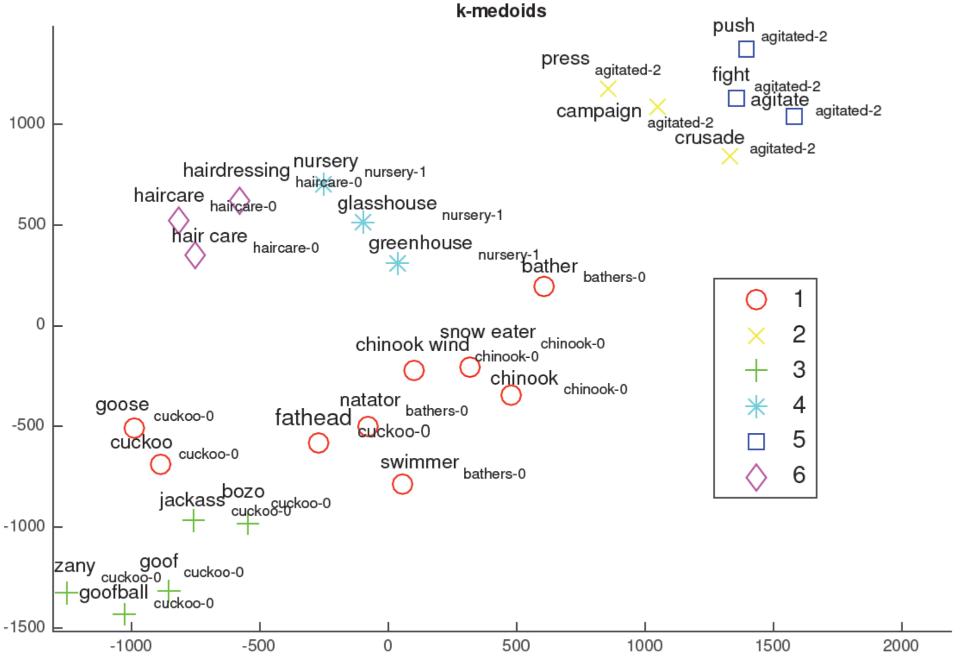
Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - 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
- Clustering

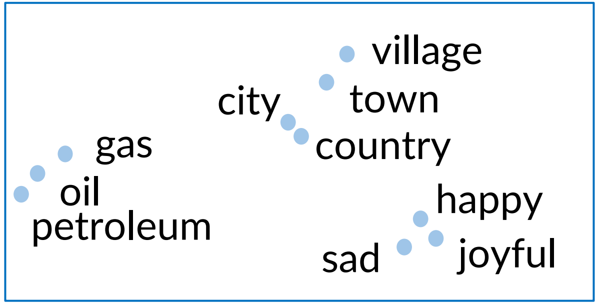
Source: Michael Zhai, Johnny Tan, and Jinho D. Choi. 2016. Intrinsic and extrinsic evaluations of word embeddings



Intrinsic evaluation

Test relationships between words

- Analogies
- Clustering
- Visualization



deeplearning.ai

Evaluating Word Embeddings

Extrinsic Evaluation

Extrinsic evaluation

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

- Time-consuming

- More difficult to troubleshoot

deeplearning.ai

Conclusion

Recap and assignment

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

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)
```