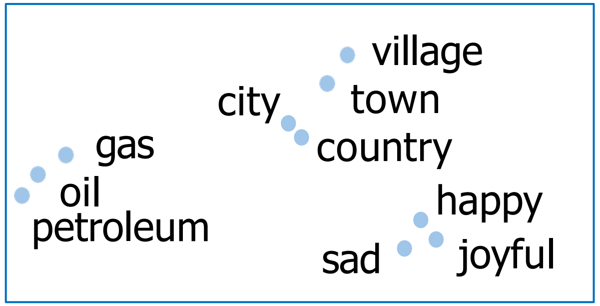
Word embedding with neural network

FPT UNIVERSITY

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

Some basic applications of word embeddings

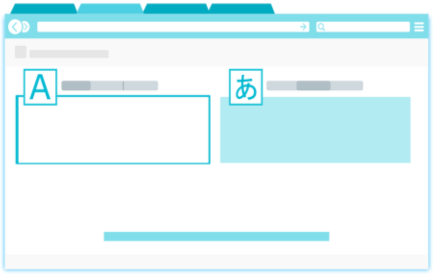


Semantic analogies and similarity

Sentiment analysis

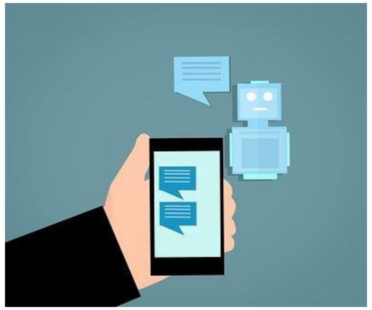
Classification of customer feedback

Advanced applications of word embeddings



Machine translation

Information extraction



Question answering

Basic Word Representations

FPT Education
FPT UNIVERSITY

- Integers
- One-hot vectors
- Word embeddings

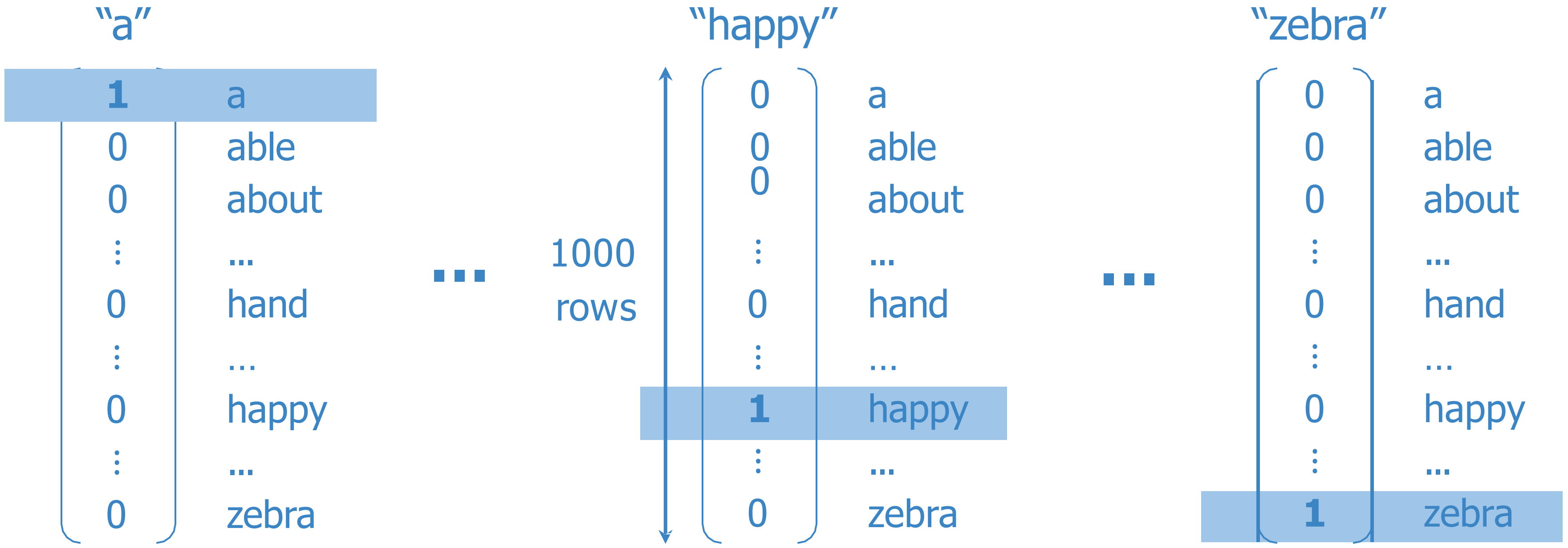
Word	Number
a	1
able	2
about	3
...	...
hand	615
…	…
happy	621
...	...
zebra	1000

hand < happy < zebra

615 ?! 621 ?! 1000

- Integers
  - + Simple
  - - Ordering: little semantic sense

One-hot vectors



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

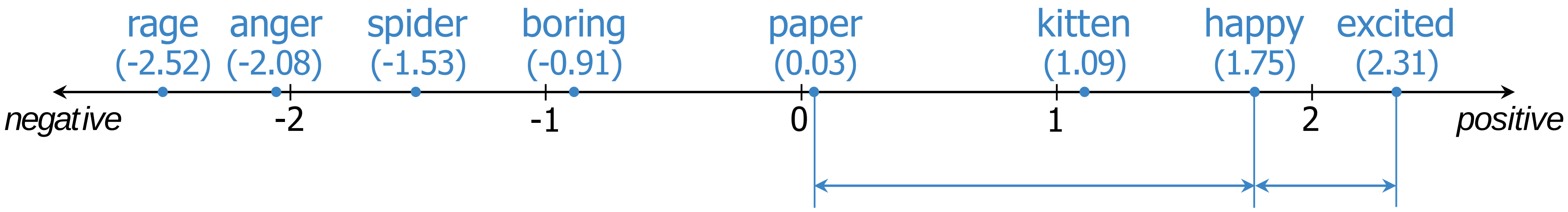
One-hot vectors

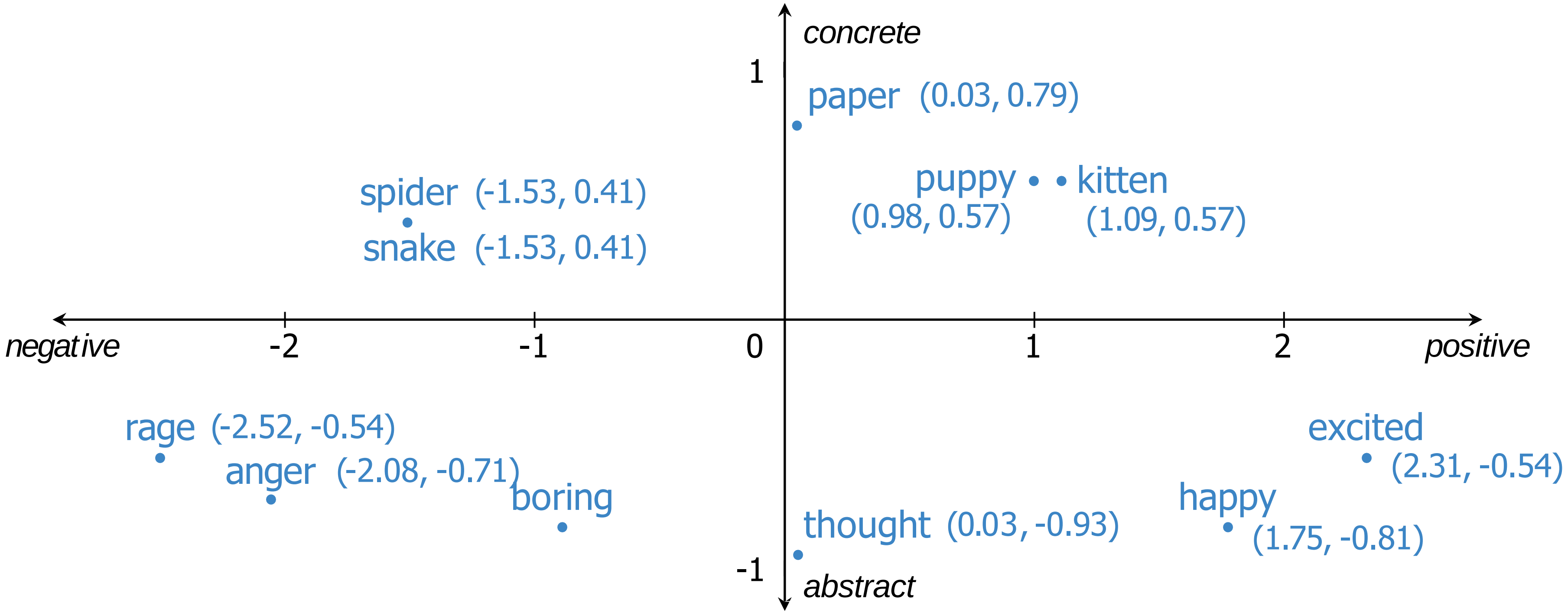
Word	Number
a	1
able	2
about	3
…	…
hand	615
…	…
happy	621
…	…
zebra	1000

"happy"

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

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

integers

word vectors

one-hot vectors

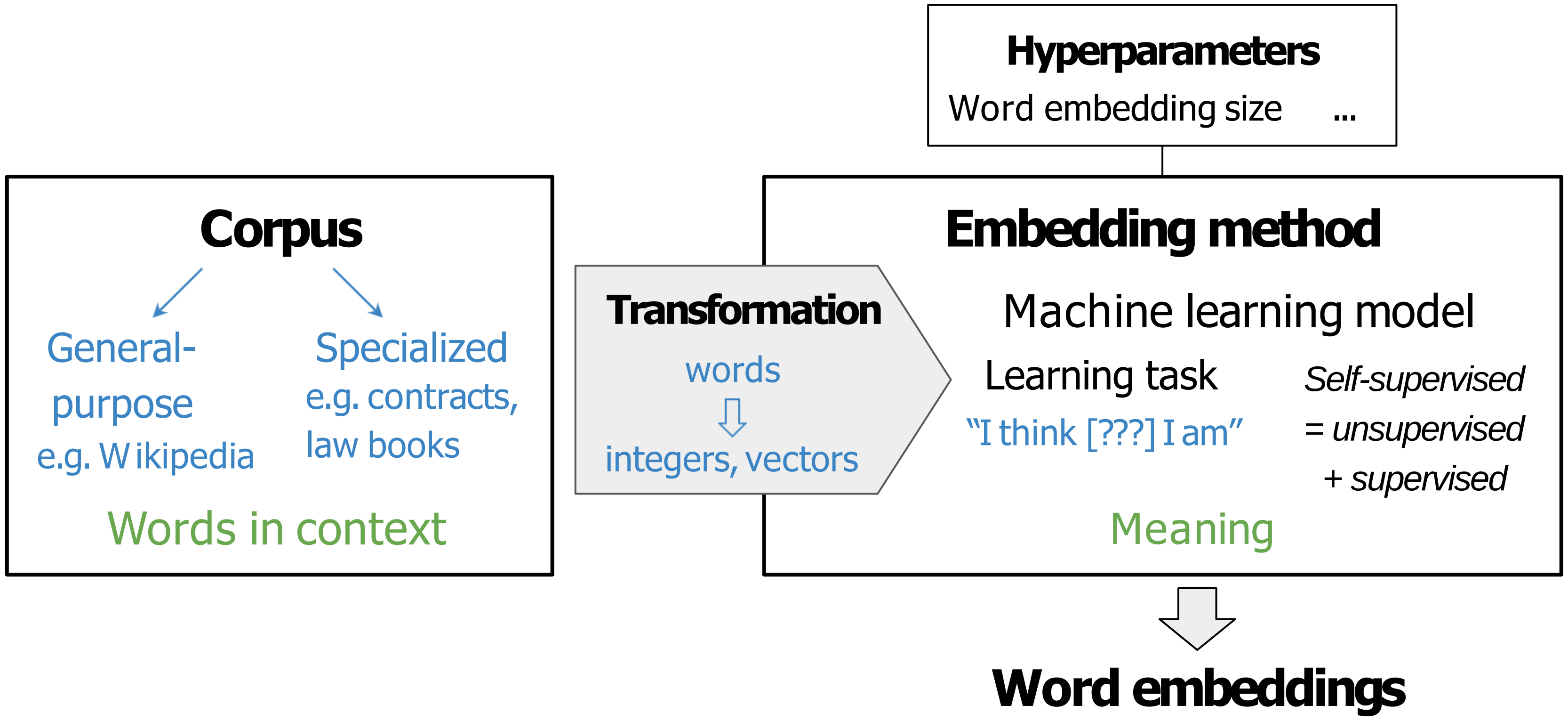
word embedding vectors

"word vectors"

word embeddings

How to Create Word Embeddings

- Word embedding process



Word embedding methods

FPT Education
FPT UNIVERSITY

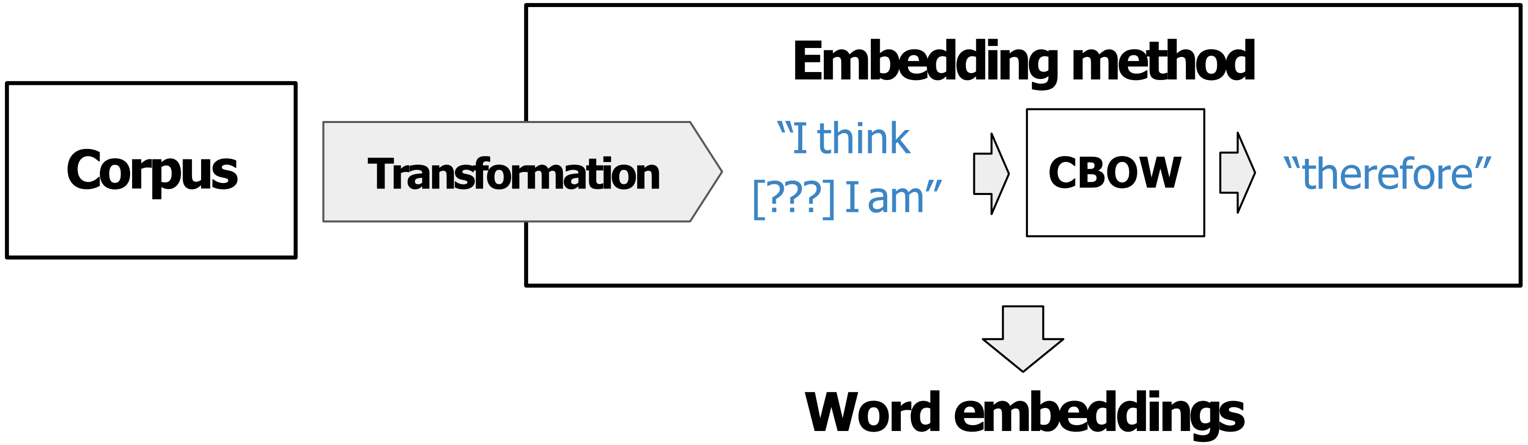
- Basic
  - word2vec (Google, 2013)
    - Continuous bag-of-words (CBOW)
    - Continuous skip-gram / Skip-gram with negative sampling (SGNS)
  - Global Vectors (GloVe) (Stanford, 2014)
  - fastText (Facebook, 2016)
    - Supports out-of-vocabulary (OOV) words

- 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 Model

- Continuous bag-of-words word embedding process



Center word prediction: rationale

FPT Education
FPT UNIVERSITY

The little ? is barking

dog
puppy
hound
terrier
...

Creating a training example

center word

I am **happy** because I am learning

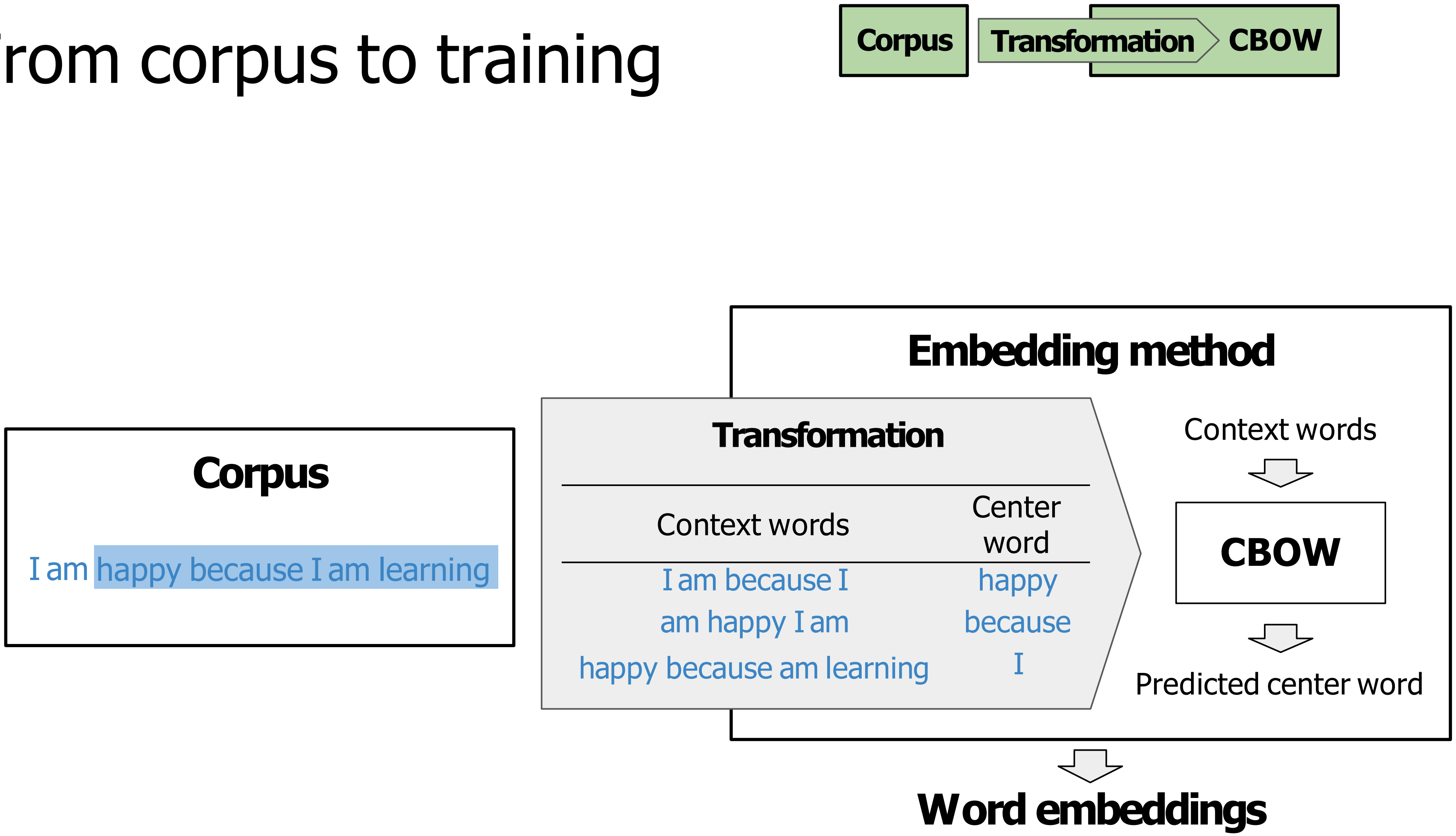
context words

C = 2
context half-size

window

window size = 5

From corpus to training



Cleaning and Tokenization

- 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 , libraries

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

emoji punctuation number

```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
```

```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', '.']
```

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

```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

FPT Education

FPT UNIVERSITY

```
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 center words into vectors

Corpus I am happy because I am learning

Vocabulary am, because, happy, I, learning

One-hot vector

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

Transforming context words into vectors

FPT Education
FPT UNIVERSITY

Average of individual one-hot vectors

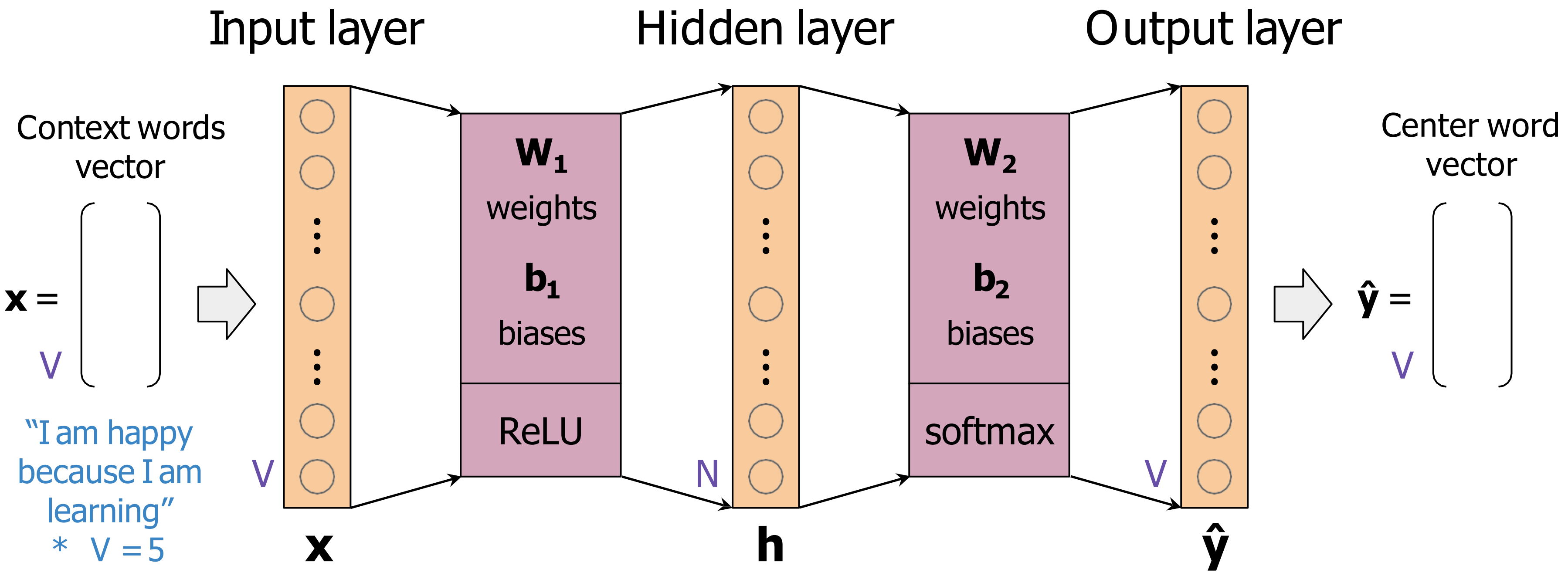
$$\left(\begin{array}{r} \\ \text{am} \\ \text{because} \\ \text{happy} \\ \text{I} \\ \text{learning} \end{array} \begin{array}{c} \text{I} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{array} + \begin{array}{c} \text{am} \\ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{array}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

Final prepared training set

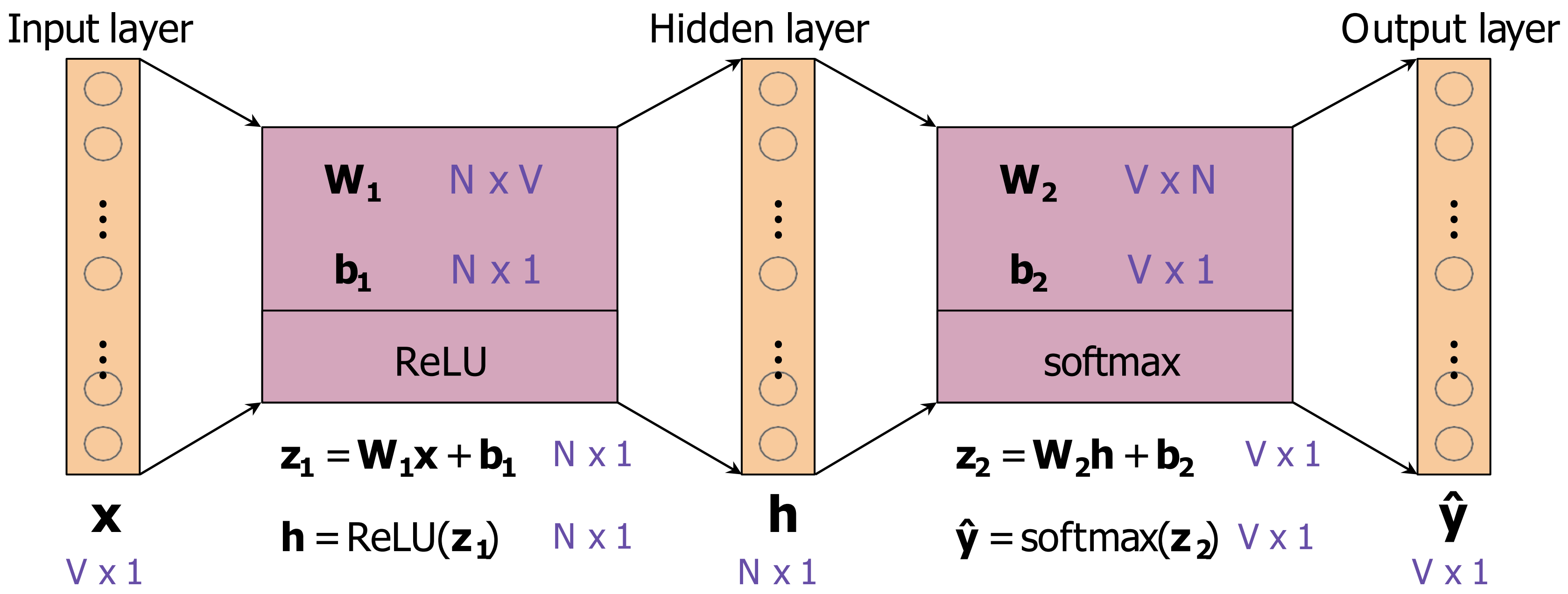
Context words	Context words vector	*Center word*	Center word vector
I am because I	[0.25; 0.25; 0; 0.5; 0]	*happy*	[0; 0; 1; 0; 0]

Architecture of the CBOW model

Hyperparameters
N: Word embedding size ...



Dimensions (single input)



Column vectors

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

$\mathbf{z_1} = (\ )$ N x 1 $\quad \mathbf{W_1} = ($ N x V $) \quad \mathbf{x} = (\ )$ V x 1 $\quad \mathbf{b_1} = (\ )$ N x 1

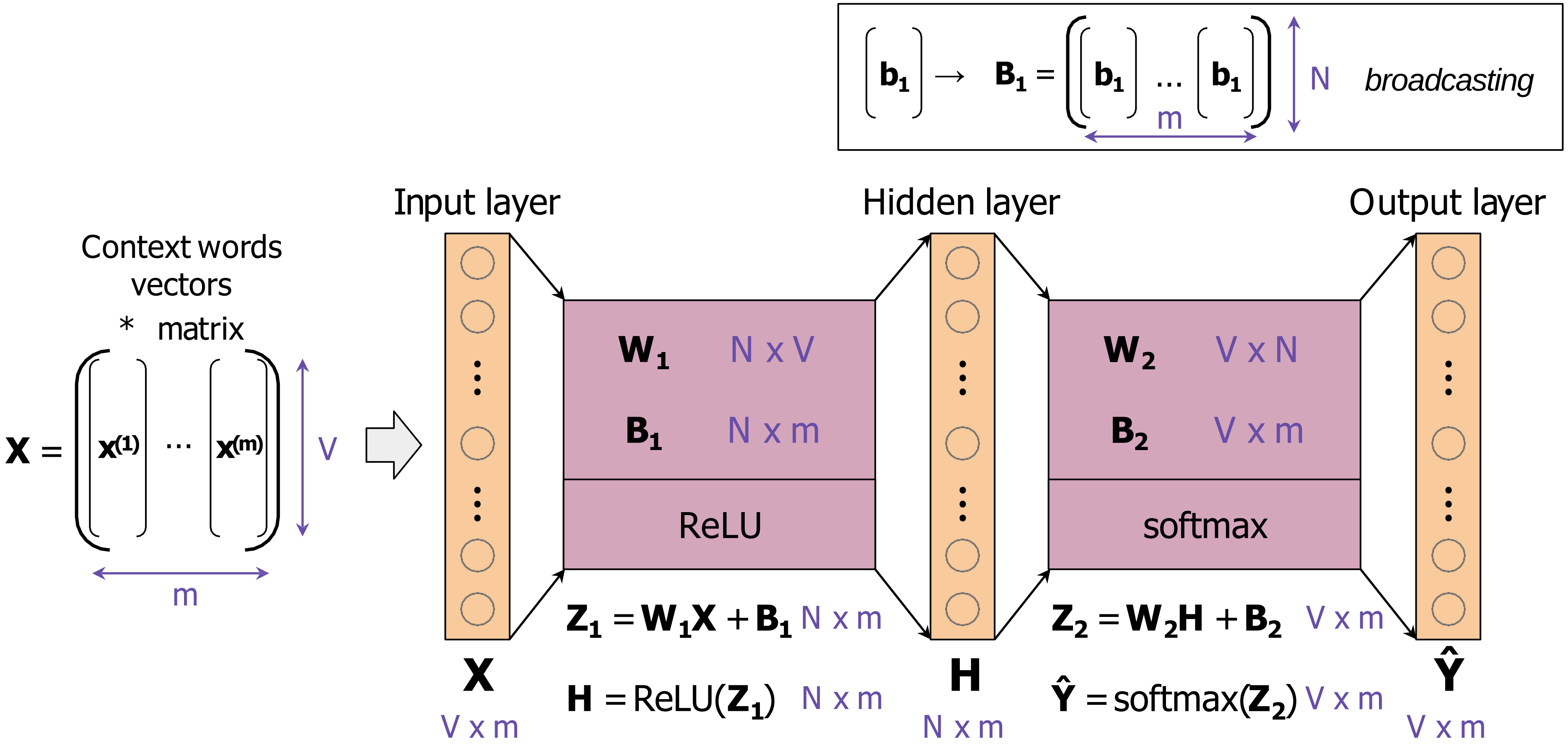
Row vectors

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

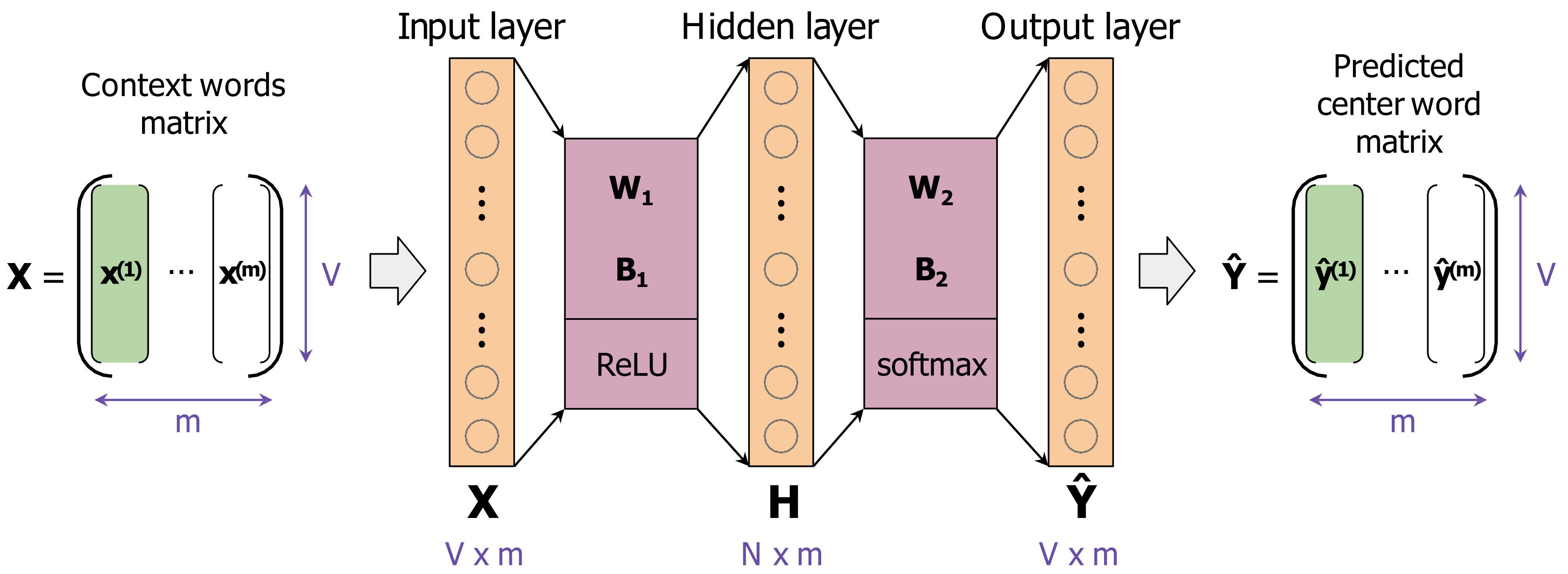
$\mathbf{b_1} = ($ 1 x N $) \quad \mathbf{W_1} = ($ N x V $) \quad \mathbf{b_1} = ($ 1 x N $)$

$\mathbf{x} = ($ 1 x V $)$

Dimensions (batch input)

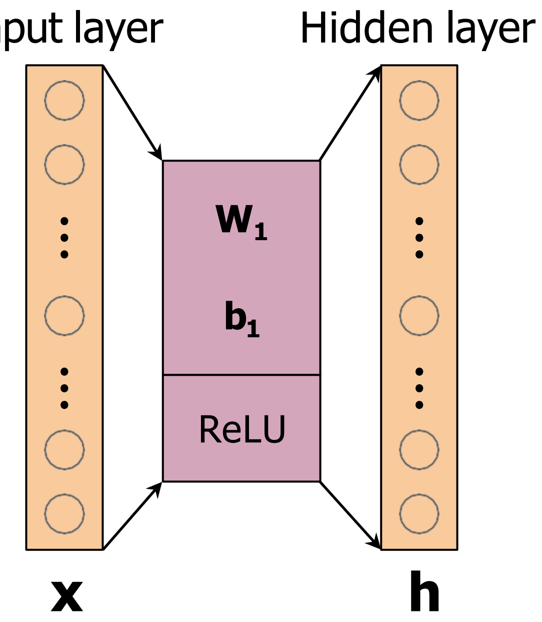


Dimensions (batch input)



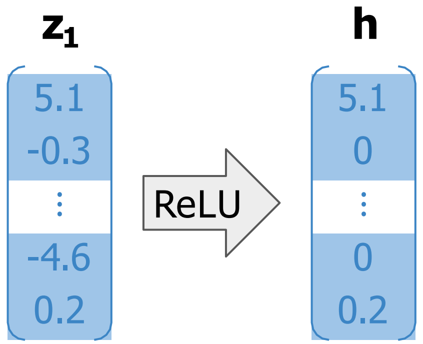
Activation Functions

- Rectified Linear Unit (ReLU)

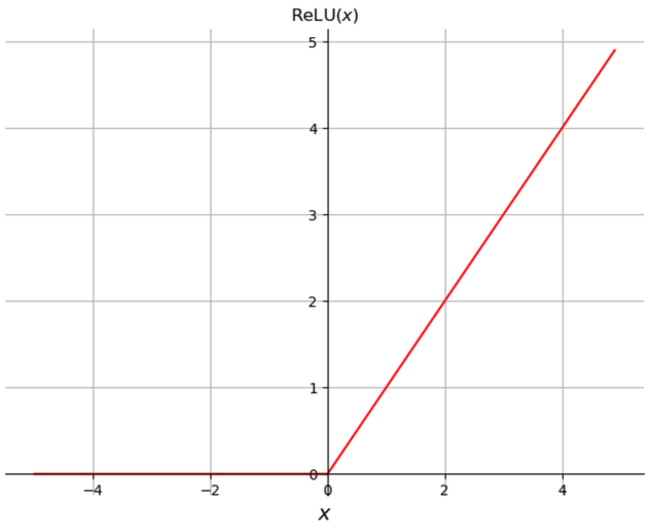


$$\mathbf{z_1} = \mathbf{W_1}\mathbf{x} + \mathbf{b_1}$$

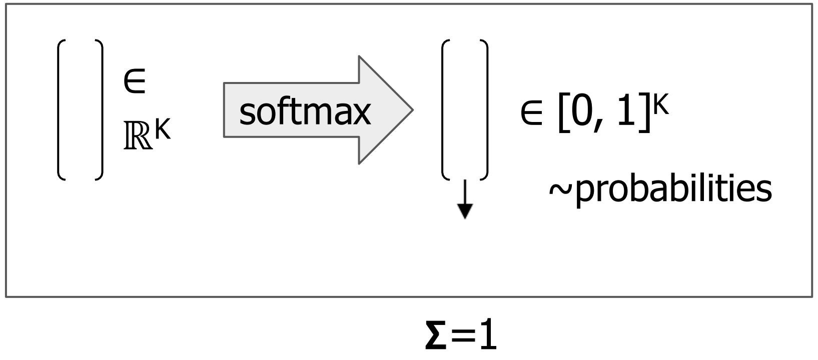
$$\mathbf{h} = \text{ReLU}(\mathbf{z_1})$$

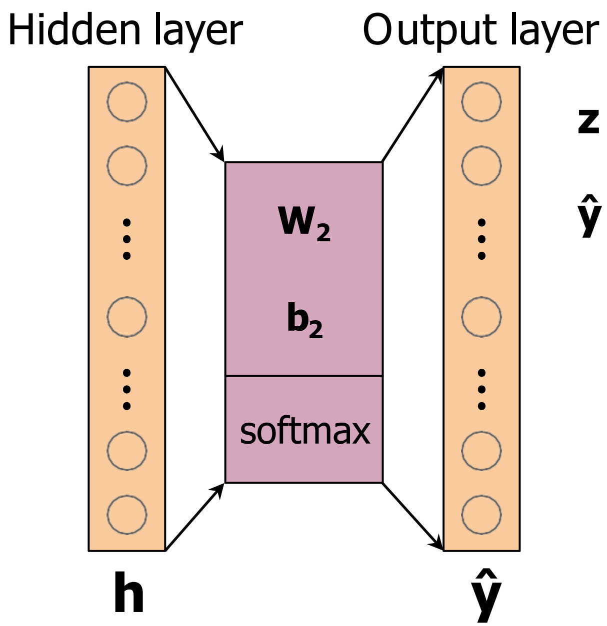


$$\text{ReLU}(x) = \max(0, x)$$



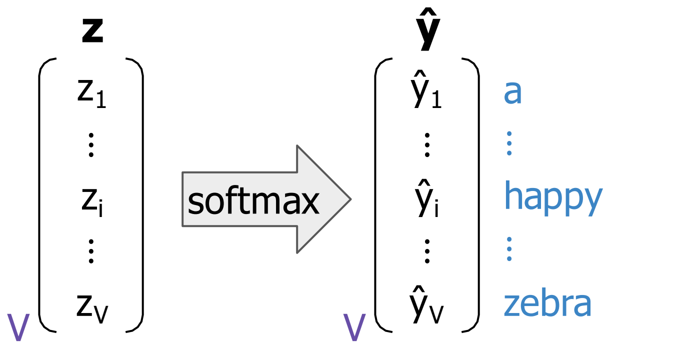
Softmax





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

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



Probabilities of being center word

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

Softmax: example

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

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

$\hat{\mathbf{y}}$	word
0.083	am
0.03	because
0.612	happy
0.225	I
0.05	learning

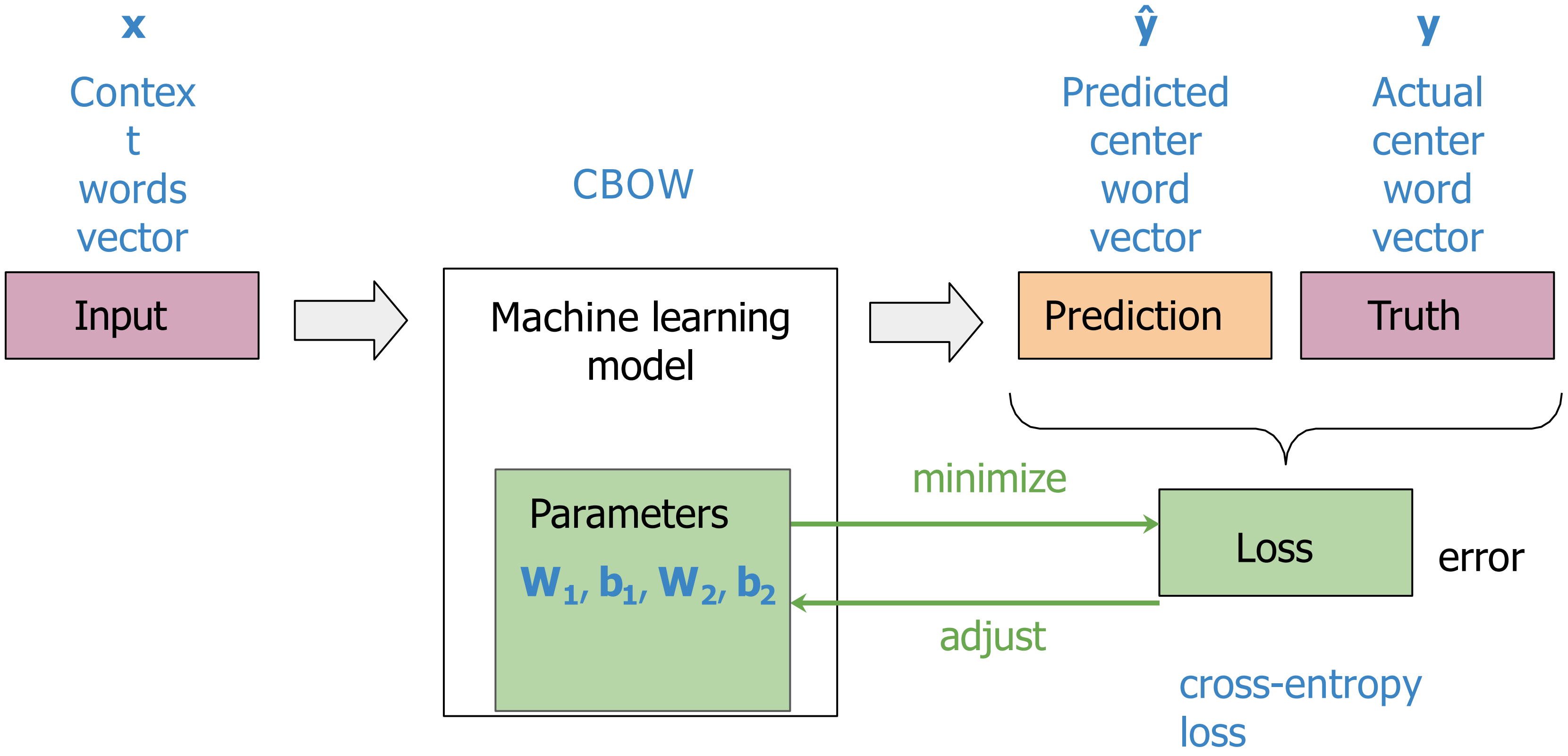
$\exp(\mathbf{z})$: $\Sigma = 97899$

$\hat{\mathbf{y}}$: $\Sigma = 1$

* Predicted center word

Cost Function

- Loss



Cross-entropy loss

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

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

I am happy because I am learning

$\mathbf{y}$		$\hat{\mathbf{y}}$	$\log(\hat{\mathbf{y}})$	$\mathbf{y} \odot \log(\hat{\mathbf{y}})$
0	am	0.083	-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

$\hat{\mathbf{y}} \xrightarrow{\log} \log(\hat{\mathbf{y}}) \rightarrow \mathbf{y} \odot \log(\hat{\mathbf{y}}) \xrightarrow{-\Sigma} J = 0.49$

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

$\hat{\mathbf{y}} \xrightarrow{\log} \log(\hat{\mathbf{y}}) \xrightarrow{\odot\, \mathbf{y}} \mathbf{y} \odot \log(\hat{\mathbf{y}}) \xrightarrow{-\Sigma} J = 4.61$

$$J = 4.61$$
$$>$$
$$J^{(correct)} = 0.49$$

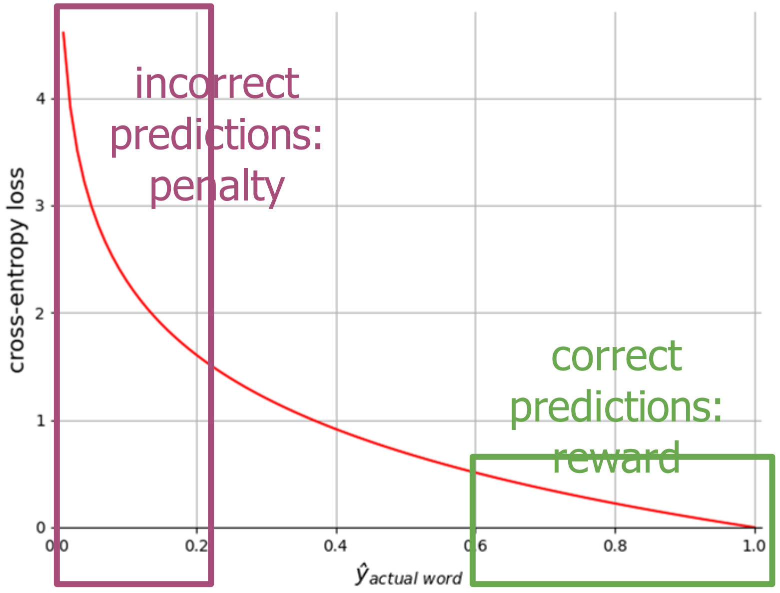
Cross-entropy loss

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

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

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

$* \ J = 4.61$

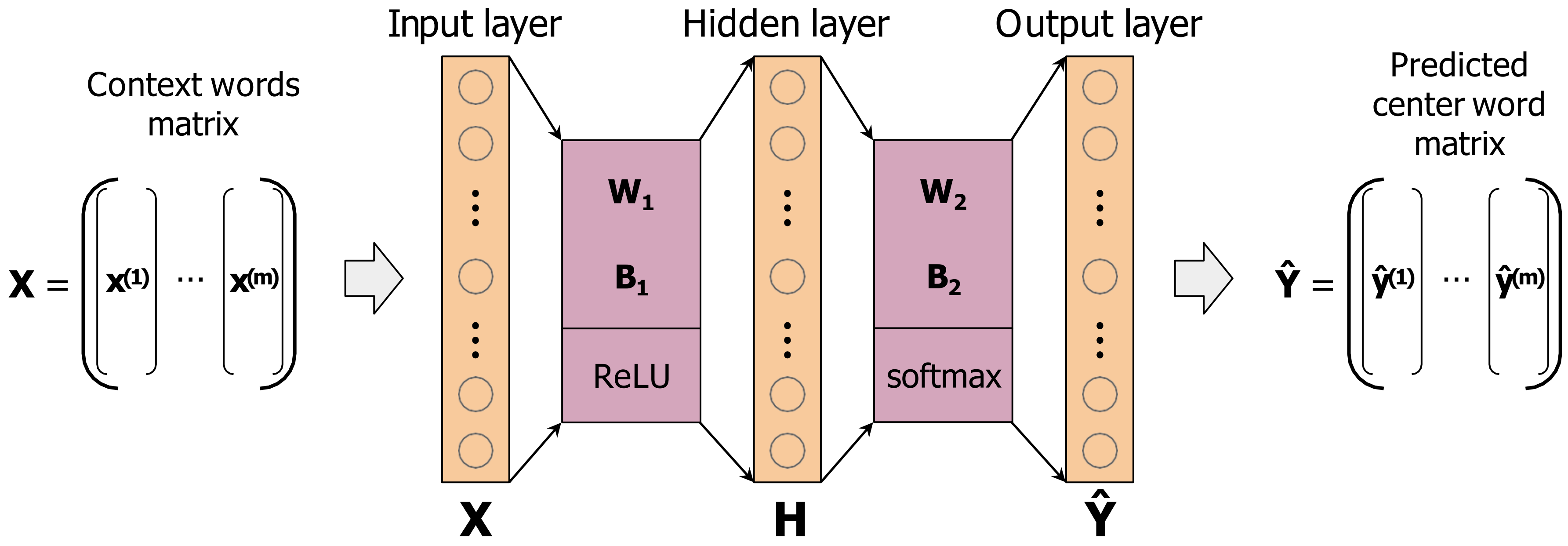


Training process

- Forward propagation

$$\mathbf{Z_1} = \mathbf{W_1 X} + \mathbf{B_1} \qquad \mathbf{Z_2} = \mathbf{W_2 H} + \mathbf{B_2}$$

$$\mathbf{H} = \text{ReLU}(\mathbf{Z_1}) \qquad \hat{\mathbf{Y}} = \text{softmax}(\mathbf{Z_2})$$



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

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

Actual center word matrix

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

Backpropagation and Gradient Descent

FPT Education
FPT UNIVERSITY

- Minimizing the cost

$$\boxed{J_{batch} = f(\mathbf{W_1}, \mathbf{W_2}, \mathbf{b_1}, \mathbf{b_2})}$$

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

$$\frac{\partial J_{batch}}{\partial \mathbf{W_1}}, \frac{\partial J_{batch}}{\partial \mathbf{W_2}}, \frac{\partial J_{batch}}{\partial \mathbf{b_1}}, \frac{\partial J_{batch}}{\partial \mathbf{b_2}}$$

- Gradient descent: update weights and biases

Backpropagation

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

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

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

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

Gradient descent

Hyperparameter: learning rate *a*

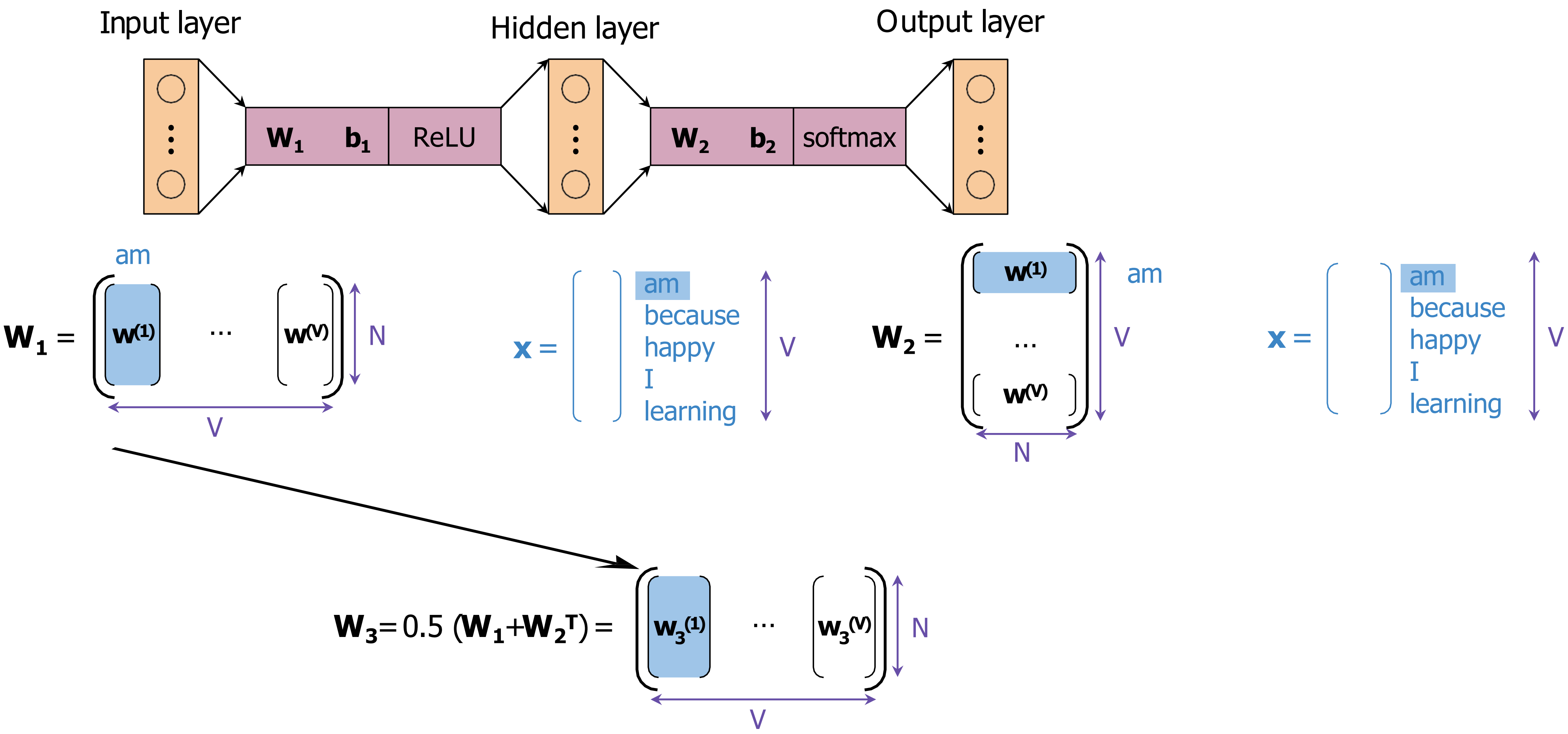
$$\mathbf{W_1} := \mathbf{W_1} - \alpha\frac{\partial J_{batch}}{\partial \mathbf{W_1}}$$

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

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

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

Extracting word embedding vectors: option 1,2,3



Evaluating Word Embeddings

- Intrinsic evaluation
  - Intrinsic evaluation methods assess how well the word embeddings inherently capture the semantic(meaning) or syntactic(grammar) relationships between the words.
  - Test on semantic analogies
  - Using a clustering algorithm to group similar word embedding vectors, and determining of the cluster's capture related words

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

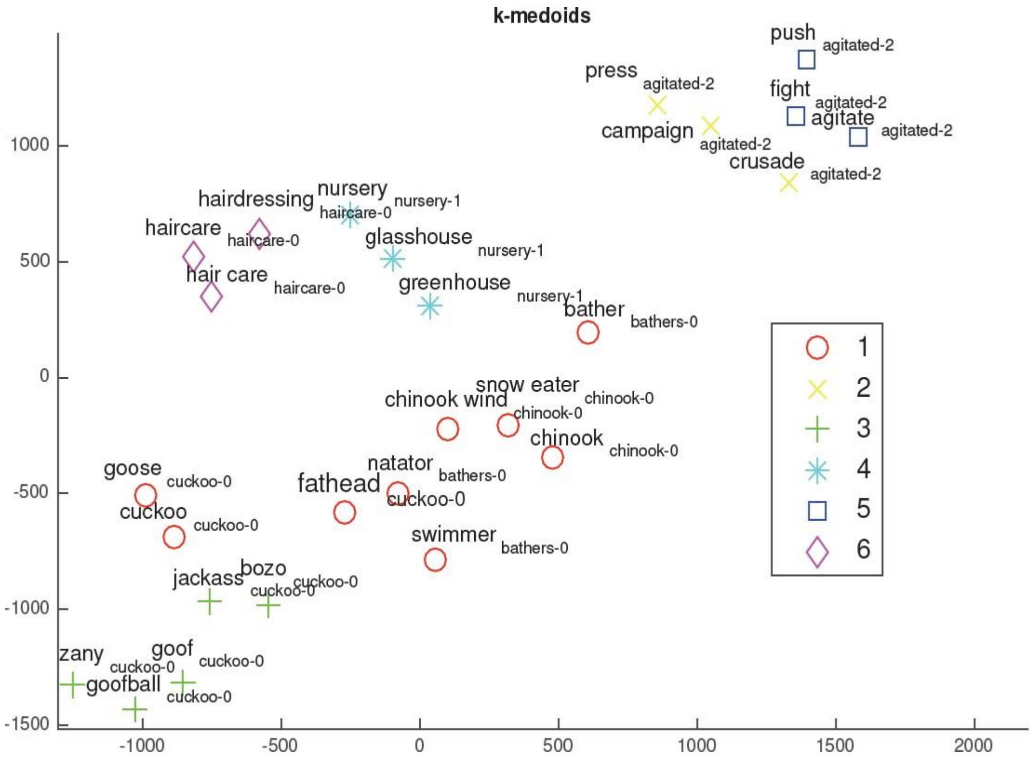
FPT Education
FPT UNIVERSITY

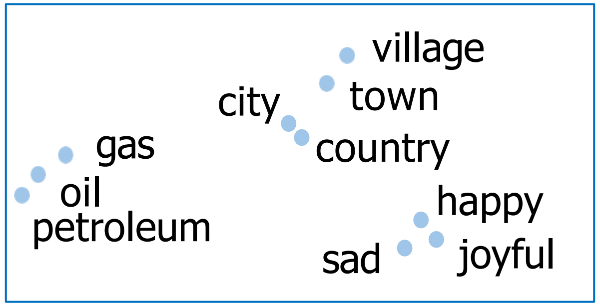
- Test relationships between words
- Analogies

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
  - Visualization





Extrinsic evaluation

FPT Education
FPT UNIVERSITY

- Test the word embeddings to perform an external task, Named Entity recognition, POS tagging
- Evaluate this classifier on the test set with some selected evaluation metric, such as accuracy or the F1 score.
- The evaluation will be more time-consuming than an intrinsic evaluation and more difficult to troubleshoot.
- 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

Named entity

Andrew works at deeplearning.ai

person *organization*