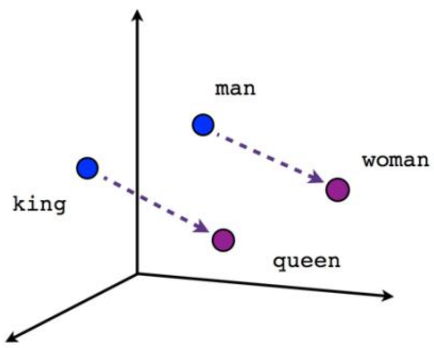
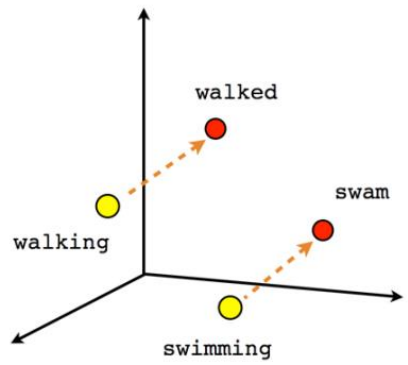
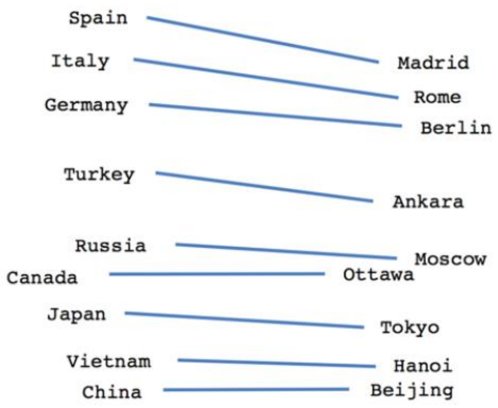


Male-Female



Verb tense



Country-Capital

Lecture 5: Text Representation II Distributed Representations

Pilsung Kang

School of Industrial Management Engineering

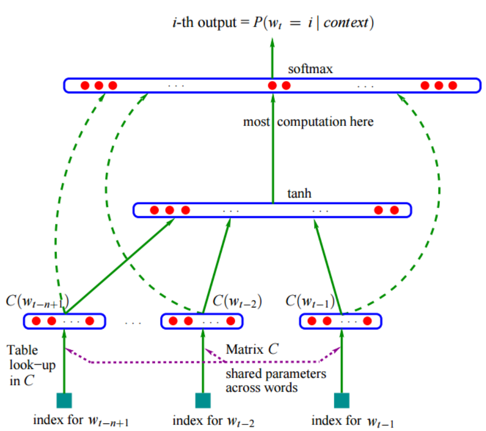
Korea University

AGENDA

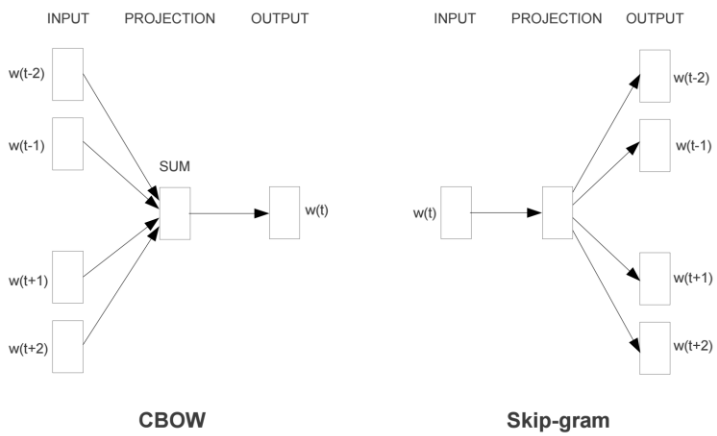
Distributed Representation: Word Embedding

- Word Embedding
  - ✓ The purpose of word embedding is to map the words in a language into a vector space so that <u>semantically similar words are located close to each other</u>.
  - ✓ Hypothetically, the number of token in English is estimated about 13M, there exist a d (< 13M) dimensional optimal space that can embed the meaning of all words.

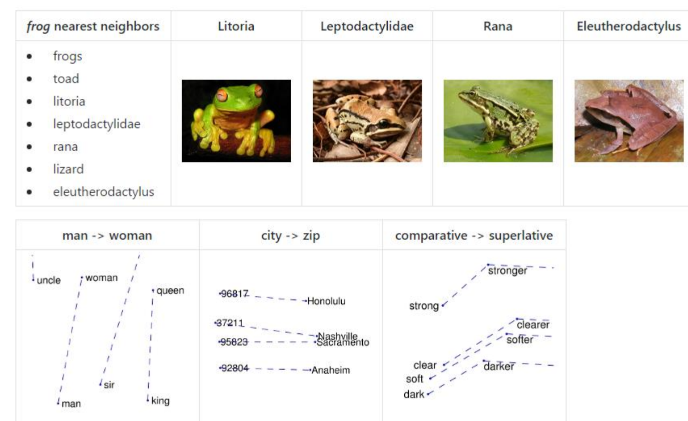
NNLM



Word2Vec



GloVe



Distributed Representation: Word Embedding

cs224d Lecture 2

- Word vectors: one-hot vector
  - ✓ The most simple & intuitive representation

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

  - ✓ Can make a vector representation, but similarities between words cannot be preserved.

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

$$(w^{hotel})^{\top} w^{motel} = (w^{hotel})^{\top} w^{cat} = 0$$

Distributed Representation: Word Embedding

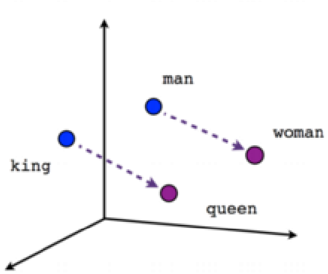
- Word vectors: distributed representation
  - ✓ A parameterized function mapping words in some language to a certain dimensional vectors

$$W : \text{words} \rightarrow \mathbb{R}^n$$

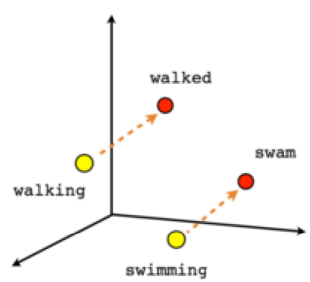
$$W(\text{``cat''}) = (0.2,\ \text{-}0.4,\ 0.7,\ \ldots)$$

$$W(\text{``mat''}) = (0.0,\ 0.6,\ \text{-}0.1,\ \ldots)$$

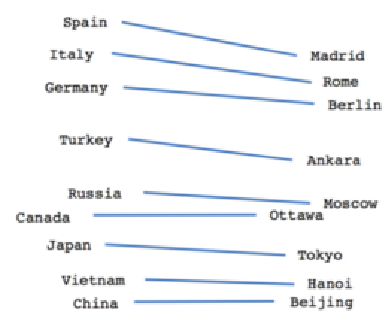
- Interesting feature of word embedding
  - ✓ Semantic relationship between words can be preserved



Male-Female



Verb tense



Country-Capital

https://www.tensorflow.org/versions/r0.7/tutorials/word2vec/index.html

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Purpose
  - ✓ Fighting the curse of dimensionality with distributed representations

✓ Associate with each word in the vocabulary a **distributed word feature vector** (a real valued vector in R^m),

✓ Express the joint **probability function** of word sequences in terms of the feature vectors of these words in the sequence,

✓ Learn simultaneously the **word feature vectors** and the parameters of that **probability function**

Neural Network Language Model (NNLM)

- Why it works?
  - ✓ If we knew that `dog` and `cat` played similar roles (semantically and synthetically), and similarity for (`the`, `a`), (`bedroom`, `room`), (`is`, `was`), (`running`, `walking`), we could naturally generalize from

```
The cat is walking in the bedroom
```

to

```
A dog was running in a room
The cat is running is a room
A dog is walking in a bedroom
The dog was waling in the room
...
```

Neural Network Language Model (NNLM)

Kim et al. (2016)

- Comparison with Count-based Language Models

  ✓ Count-based Language Models

  By the chain rule, any distribution can be factorized as

$$p(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t | w_1, \ldots, w_{t-1})$$

  Count-based n-gram language models make a Markov assumption:

$$p(w_t | w_1, \ldots, w_t) \approx p(w_t | w_{t-n}, \ldots, w_{t-1})$$

  Need smoothing to deal with rare n-grams.

Neural Network Language Model (NNLM)

- Language Model Example

Neural Network Language Model (NNLM)

Kim et al. (2016)

- Comparison with Count-based Language Models

✓ NNLM

- Represent words as dense vectors in $\mathbb{R}^n$ (word embeddings).

  $\mathbf{w}_t \in \mathbb{R}^{|\mathcal{V}|}$: One-hot representation of word $\in \mathcal{V}$ at time t
  $\Rightarrow \mathbf{x}_t = \mathbf{X}\mathbf{w}_t$: Word embedding ($\mathbf{X} \in \mathbb{R}^{n \times |\mathcal{V}|}$, $n < |\mathcal{V}|$)

- Train a neural net that composes history to predict next word.

$$p(w_t = j | w_1, \ldots, w_{t-1}) = \frac{\exp(\mathbf{p}^j \cdot g(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) + q^j)}{\sum\limits_{j' \in \mathcal{V}} \exp(\mathbf{p}^{j'} \cdot g(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) + q^{j'})}$$

$$= \text{softmax}(\mathbf{P}g(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) + \mathbf{q})$$

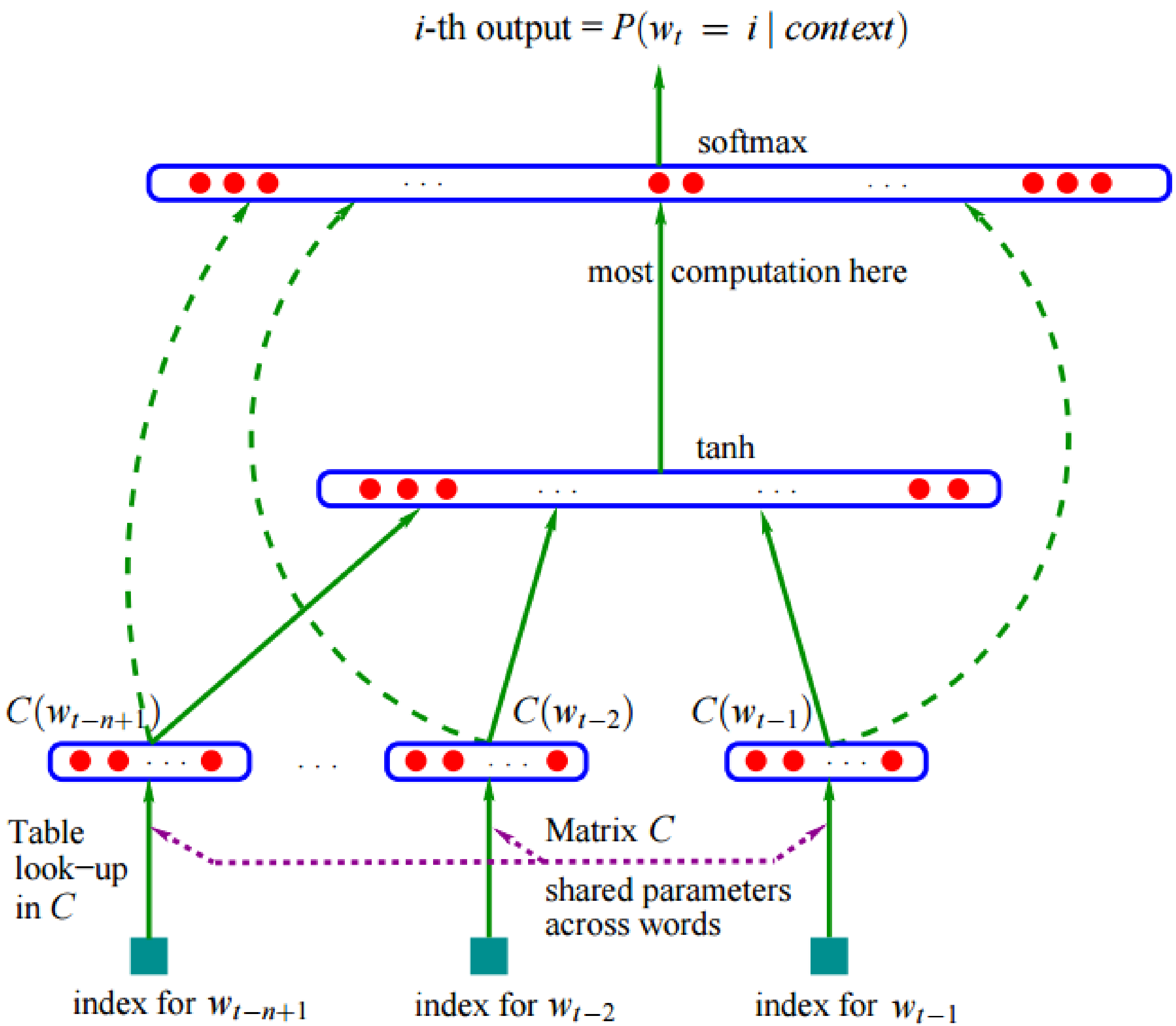
$\mathbf{p}^j \in \mathbb{R}^m$, $q^j \in \mathbb{R}$: Output word embedding/bias for word $j \in \mathcal{V}$

g : Composition function

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM



Neural Network Language Model (NNLM)

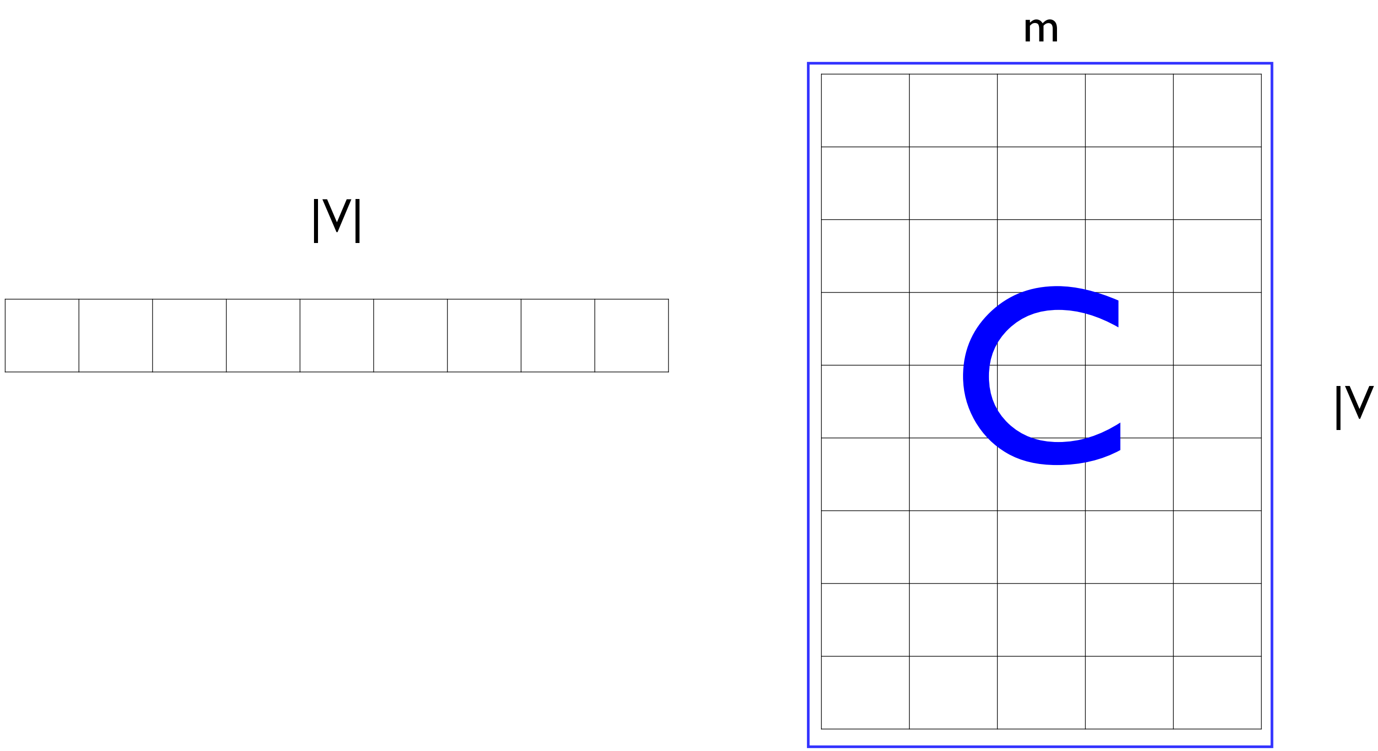
Bengio et al. (2003)

- Learning NNLM
  - ✓ The objective is to learn a good model $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$, in the sense that it gives high out-of-sample likelihood
  - ✓ Two constraints
    - For any choice of $w_1^{t-1}, \sum_{i=1}^{|V|} f(i, w_{t-1} \cdots w_{t-n+1}) = 1$ (어떤 조건에서도 이후 단어들이 생성될 확률의 총 합은 1)
    - f ≥ 0 (각 단어가 생성될 확률은 0보다 크거나 같아야 함)

Neural Network Language Model (NNLM)

Bengio et al. (2003)

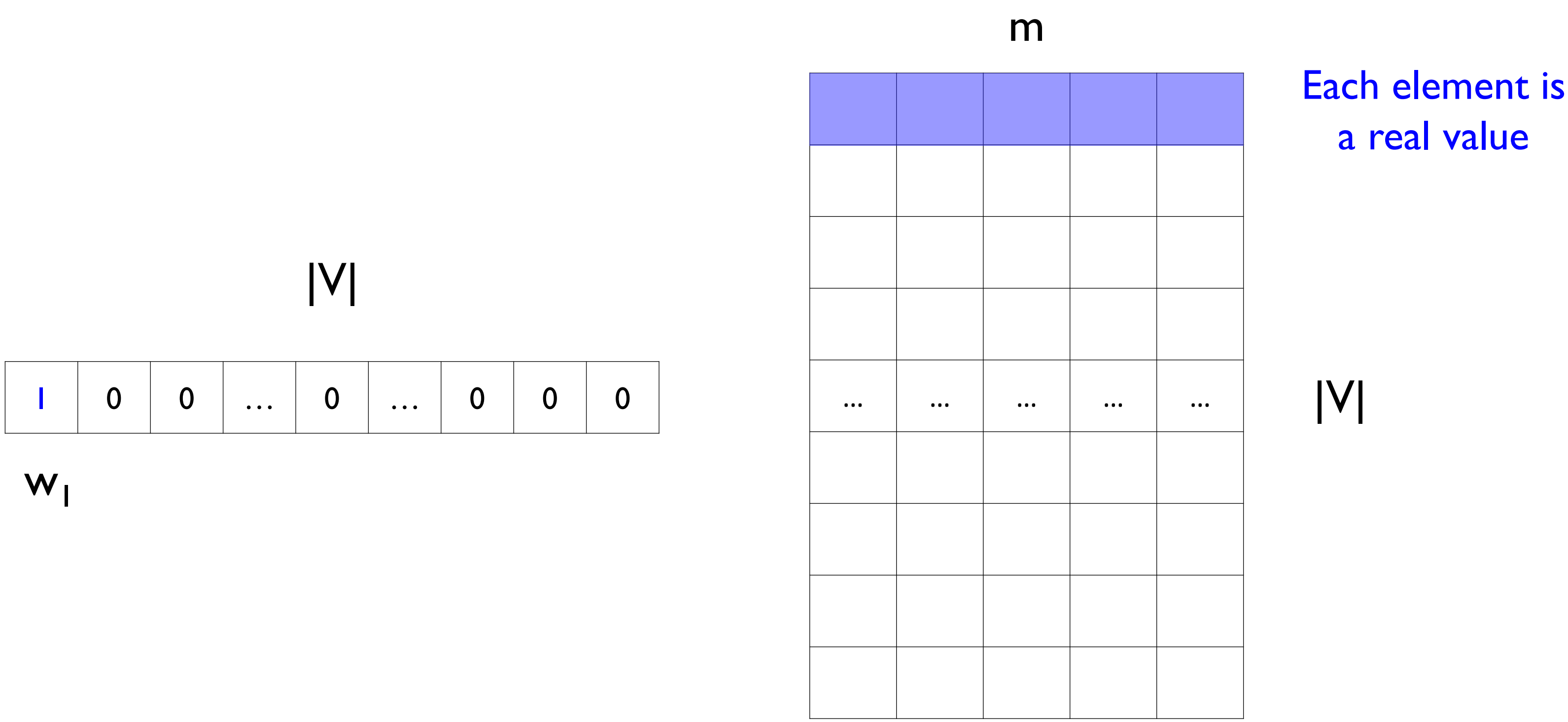
- Learning NNLM
  - ✓ Decompose the function $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t | w_1^{t-1})$ in two parts:
    - A mapping *C*, a.k.a the [lookup table](), from any element *i* of *V* to a real vector $C(i) \in R^m$, it represents the distributed feature vectors associated with each word in the vocabulary. *C* is represented by a $|V| \times m$ matrix of free parameters



Neural Network Language Model (NNLM)

Bengio et al. (2003)

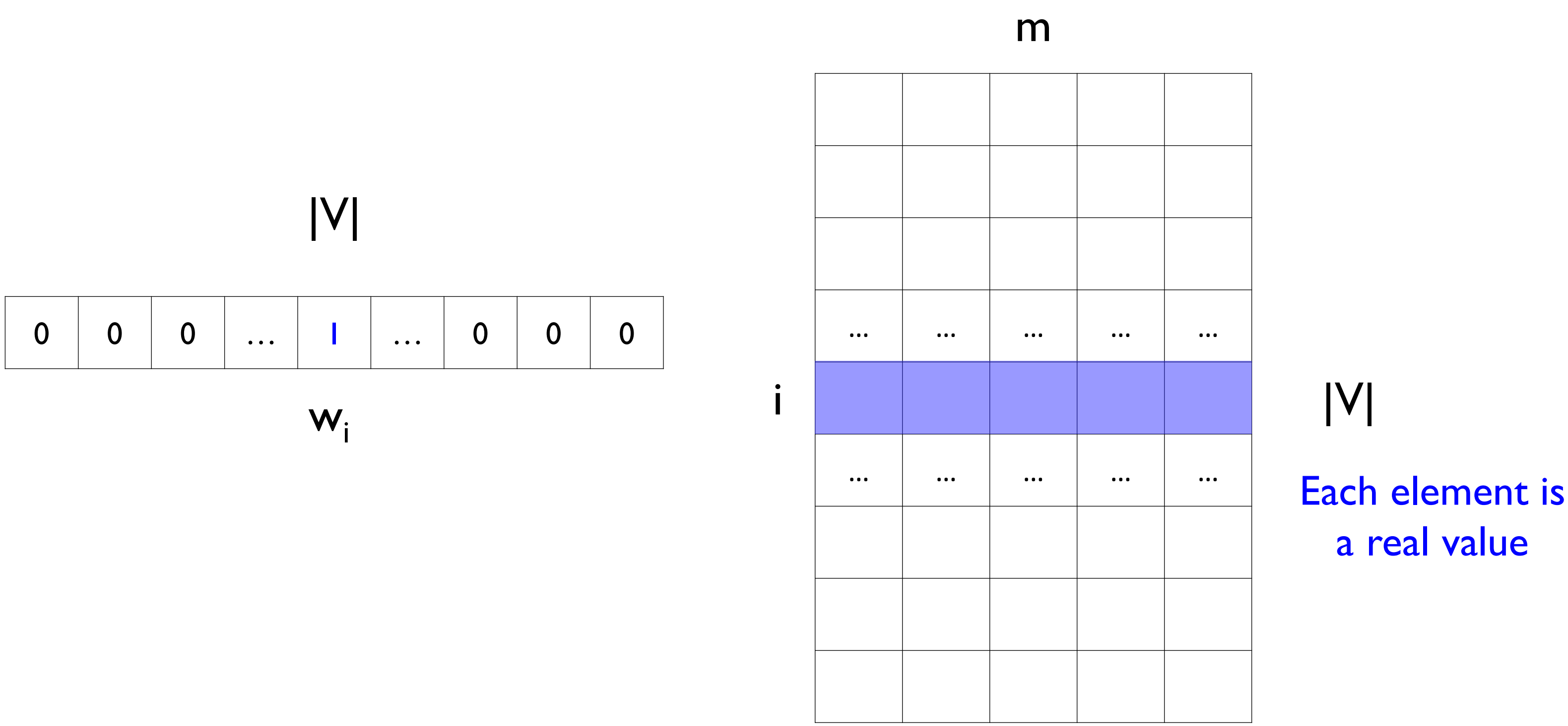
- Learning NNLM
  - ✓ Decompose the function $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t | w_1^{t-1})$ in two parts:
    - A mapping *C*, a.k.a the lookup table, from any element *i* of *V* to a real vector $C(i) \in R^m$, it represents the distributed feature vectors associated with each word in the vocabulary. *C* is represented by a $|V| \times m$ matrix of free parameters



Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM
  - ✓ Decompose the function $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t | w_1^{t-1})$ in two parts:
    - A mapping *C*, a.k.a the lookup table, from any element *i* of *V* to a real vector $C(i) \in R^m$, it represents the distributed feature vectors associated with each word in the vocabulary. *C* is represented by a $|V| \times m$ matrix of free parameters



Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM
  - ✓ Decompose the function $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$ in two parts:
    - The probability function over words, expressed with *C*: a function g maps an input sequence of feature vectors for words in context, $(C(w_{t-n+1}), \cdots C(w_{t-1}))$, to a conditional probability distribution over words in V for the next word w_t. The output of g is a vector whose i^{th} element estimates the probability $\hat{P}(w_t = i|w_1^{t-1})$

g(준다|너에게, 나의 입술을, 처음으로) = ?

g(지운다|너에게, 나의 입술을, 처음으로) = ?

g(맡긴다|너에게, 나의 입술을, 처음으로) = ?

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM

$$f(i, w_{t-1} \cdots w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$$

✓ The function f is a composition of these two mappings (C and g), with C being shared across all the words in the context.

- The parameters of the mapping C are simply the feature vectors themselves, represented by a $|V| \times m$ matrix C whose row i is the feature vector C(i) for word i
- The function g may be implemented by a feed-forward or recurrent neural network or another parameterized function, with parameters ω.

✓ Training is done by maximizing the penalized log-likelihood of the training corpus

$$L = \frac{1}{T} \sum_t \log f(i, w_{t-1} \cdots w_{t-n+1}; \theta) + R(\theta)$$

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM
  - ✓ The neural network has one hidden layer beyond the word features mapping, and optionally, direct connections from the word features to the output.
  - ✓ Computation of the output layer

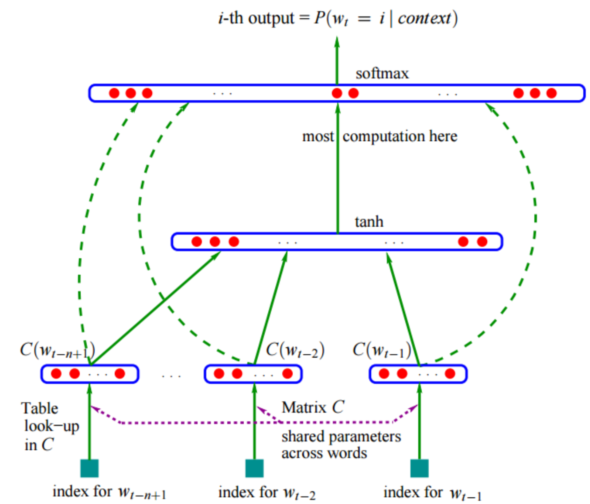
$$\hat{P}(w_t | w_{t-1}, \cdots, w_{t-n+1}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

$$y = b + Wx + U \cdot tanh(d + Hx)$$

  - W is optionally zero (no direct connections from the input to the output)
  - x is the word features layer activation vector $x = (C(w_{t-1}), \cdots, C(w_{t-n+1}))$
  - h is the number of hidden units

Neural Network Language Model (NNLM)

Bengio et al. (2003)



- Learning NNLM
  - ✓ The free parameters of the model

$$y = b + Wx + U \cdot tanh(d + Hx)$$

    - the output bias b (|V| elements)
    - the hidden layer biases d (with h elements)
    - the hidden-to-output weights U (a |V| by h matrix)
    - the word features to output weights W (a |V| by (n-1)m matrix)
    - the hidden layer weight H (a h by (n-1)m matrix)

  - ✓ Stochastic gradient ascent

$$\theta \leftarrow \theta + \varepsilon \frac{\partial \log \hat{P}(w_t | w_{t-1}, \cdots, w_{t-n+1})}{\partial \theta}$$


ANY
questions ?