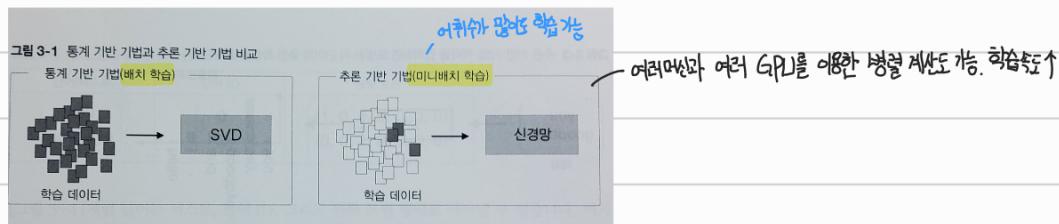


## 추론 기반 기법

### 신경망을 이용

- 분포가설을 기반으로 함 : 단어의 동시 발생 가능성을 얼마나 잘 모델링 하는가가 중요한 연구 주제
- 추측하는 것이 목적이며, ① 부산물로 단어의 분산 표현을 얻음
- 현업에서 자주 어휘 수는 어마어마 함 Ex. 영어는 100만을 훨씬 넘음 100만 x 100만의 행렬에 SVD 적용은 현상적지 않음
- 전체 dataset을 1회 차원화 단어의 분산 표현 얻음

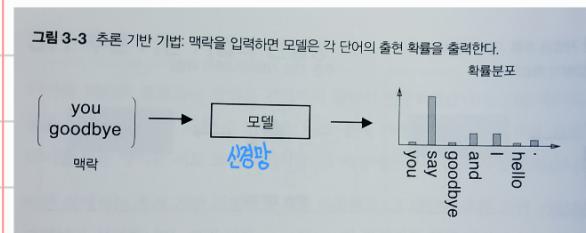


## 추론 기반 기법 개요

you [?] goodbye and I say hello.

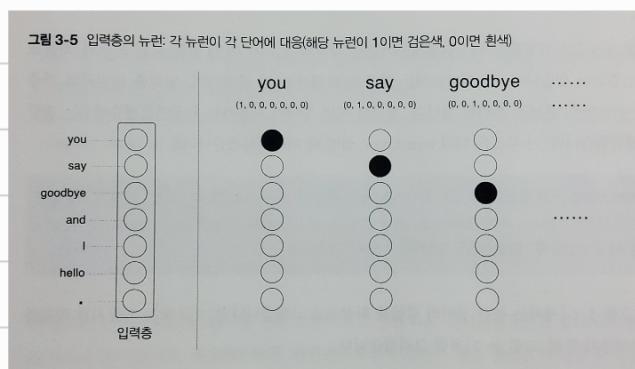
주변 단어들을 맥락으로 [?]에 어떤 단어가 들어갈지 추론

- 위와 같은 추론 문제를 반복해서 풀면서 단어의 출현 패턴 학습



## 신경망에서의 단어 처리

- 단어를 고정길이 벡터로 변환해야 함 : 원-핫 표현 사용
- 총 어휘 수 만큼의 원소를 갖는 벡터. 인덱스가 단어 [?]와 같은 원소는 1. 나머지는 0.



• 단어를 벡터로 나타낼 수 있음

• 신경망을 구성하는 계층들은 벡터를 처리할 수 있음

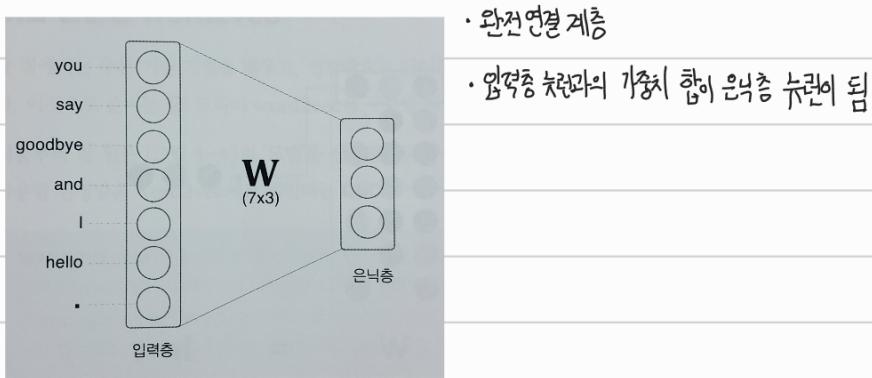
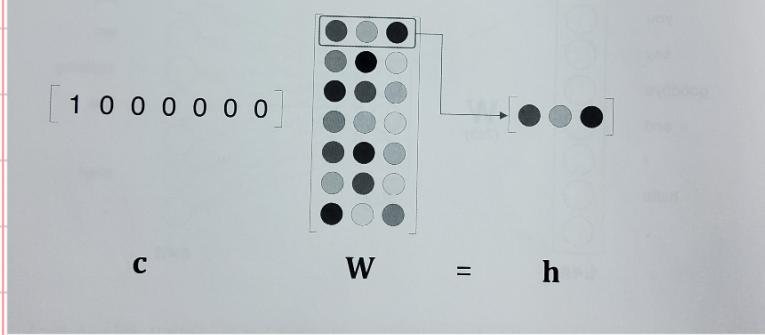


그림 3-8 맥락  $c$ 와 가중치  $W$ 의 곱으로 해당 위치의 행벡터가 추출된다(각 요소의 가중치 크기는 흑백의 진하기로 표현).



· C가 원핫벡터이기 때문에  $c$ 와  $W$ 의 곱은  $W$ 의 행추출과 같음

행을 추출할 때마다 행렬곱은 비례적.

→ 나중에 개선

## 단순한 word2vec

### CBOW 모델

- Continuous bag-of-words 모델
- 맥락을 부여 target을 추측하는 신경망  
↳ Input: 맥락

그림 3-9 CBOW 모델의 신경망 구조

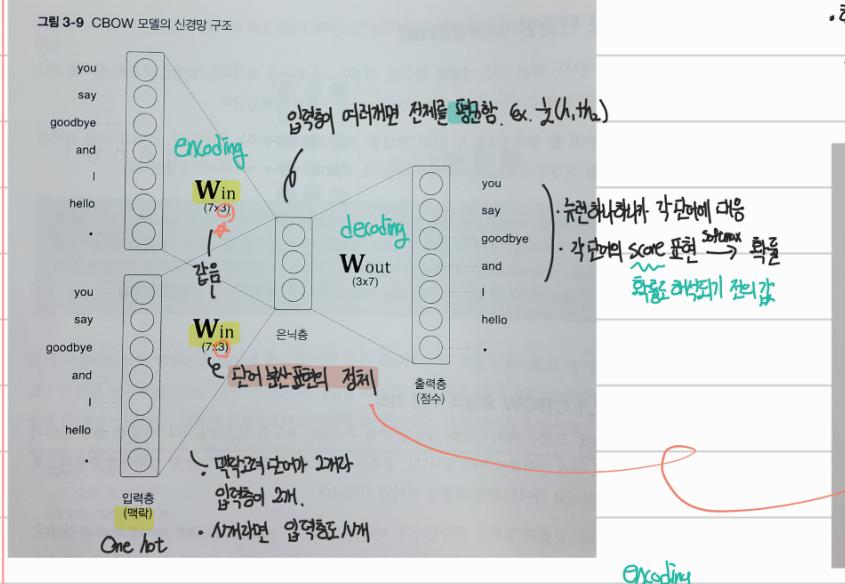
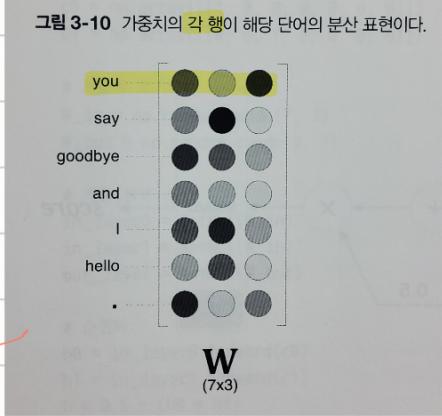


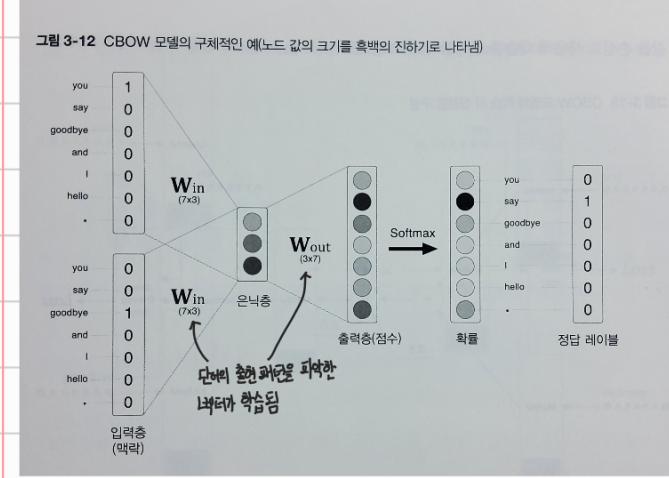
그림 3-10 가중치의 각 행이 해당 단어의 분산 표현이다.



- 웃기지만 노드수로 입력층의 노드수보다 적게 하는 것이 핵심  $\rightarrow$  간결한 밀집벡터를 얻을 수 있음
- 활성화 함수를 사용하지 않음

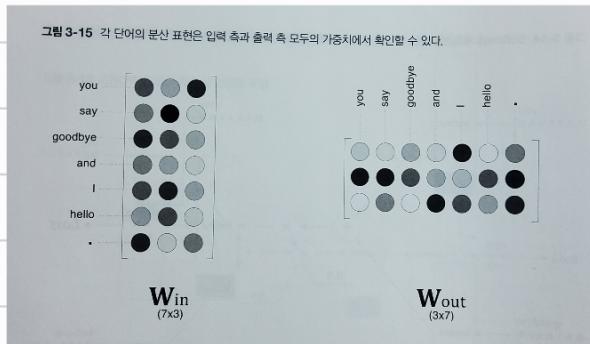
### CBOW 모델의 학습

그림 3-12 CBOW 모델의 구체적인 예(노드 값의 크기를 흑백의 진하기로 나타냄)



- 결국 다중 클래스 분류를 수행하는 신경망  $\Rightarrow$  학습시에 softmax와 cross entropy error만 있으면 됨

## Win과 Wout 모듈에 단어의 분산 표현이 학습됨



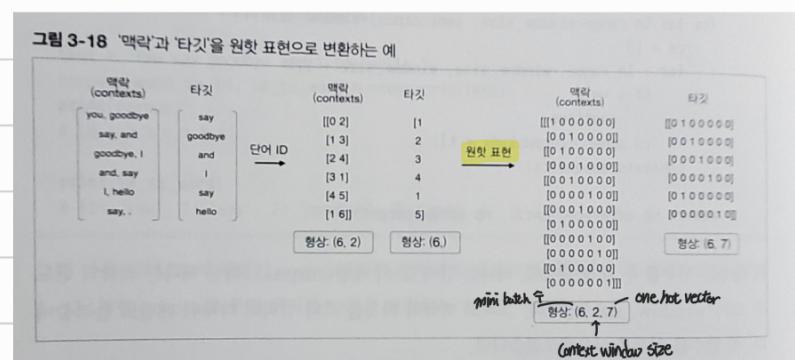
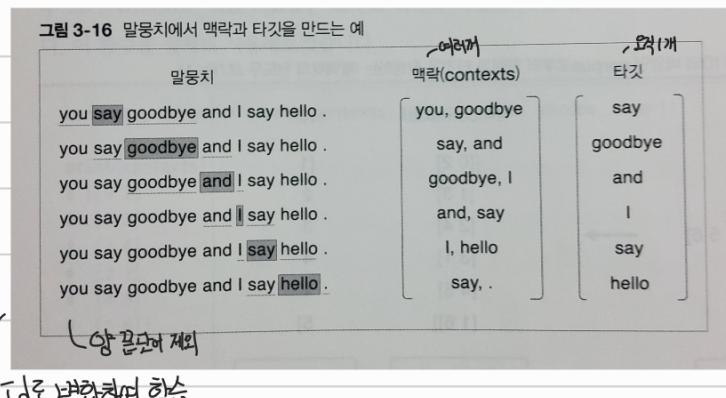
• Win만 사용 - Word2vec에서 대중적인 선택

• Wout만 사용

• 둘 다 사용 - Glove에서는 둘을 더했을 때 좋은 결과를 얻었음

## 학습 데이터 준비

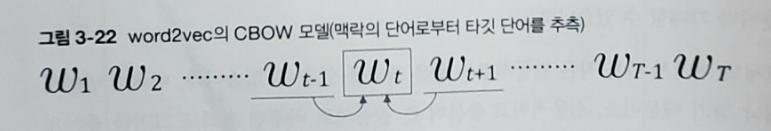
- Context와 target 준비. 신경망이 Context 입력시 target이 출현할 확률을 높이도록 학습



## Word2vec 보충

### (Bow) 모델과 확률

- 사후 확률  $P(A|B)$  : 사건 A 일어난 후의 확률 = B라는 사건 주어졌을 때 A가 일어날 확률



(Bow는  $P(w_t | w_{t-1}, w_{t+1})$ 을 모델링 하고 있음)

t<sub>w</sub>=1, 나머지는 0

$$L = -\sum_k t_k \log y_k = -\sum_k t_k \log P(w_k | w_{t-1}, w_{t+1}) = -\log P(w_t | w_{t-1}, w_{t+1})$$

(교차 엔드로프)

단순히 log를 취하고 -를 넣음 (negative log likelihood)

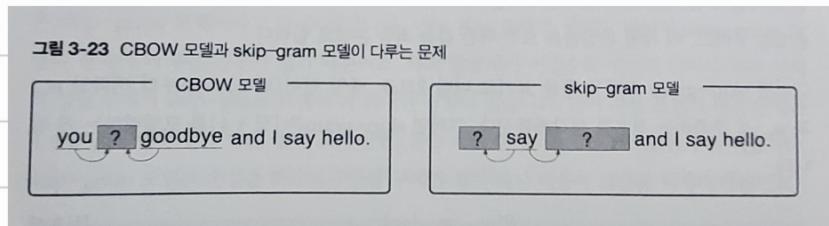
말뭉치 전체

$$L = -\frac{1}{T} \sum_t \log P(w_t | w_{t-1}, w_{t+1})$$

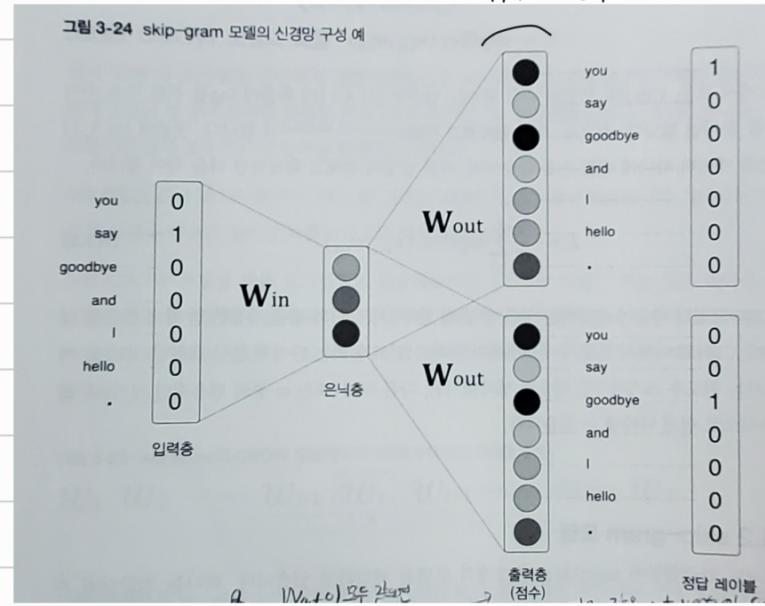
Cbow는 이 값을 가능한 적게 만들자 함

## Skip-gram 모델

- Target와 Context 추출



### 맥락의 수만큼 추출



- 각 출력층에서 softmax with loss로 개별적 손실을 구하고 이 개별 손실들을 모두 다른 값을 최종 손실로 함

## 확률 표현

- $P(w_{t+1}, w_{t+2} | w_t)$  를 모델링
- 맥락의 단어들 사이에 관련성이 없다고 가정하고 다음과 같이 분해:

$$P(w_{t+1}, w_{t+2} | w_t) = P(w_{t+1} | w_t) P(w_{t+2} | w_t)$$

- 위의식을 교차엔트로피에 적용

$$\begin{aligned} L &= -\log P(w_{t+1}, w_{t+2} | w_t) = -\log P(w_{t+1} | w_t) P(w_{t+2} | w_t) \\ &\text{（샘플 개체리）} \quad = -(\log P(w_{t+1} | w_t) + \log P(w_{t+2} | w_t)) \\ &\text{（말뭉치 전체）} \quad \text{（맥락별 손실 함수를 구한 다음 모니터함）} \end{aligned}$$

$$L = -\frac{1}{T} \sum_t (\log P(w_{t+1} | w_t) + \log P(w_{t+2} | w_t))$$

Skip-gram 모델은 맥락의 수만큼 추출  $\rightarrow$  손실행수는 각 맥락에서 구한 손실의 총합

CBOW는 target 하나의 손실을 구함

- 단어 분산 표현의 정밀도 면에서 skip-gram의 결과가 더 좋음

（말뭉치가 거칠수록 차별로 구애나 유추문제의 성능 ↑）

- CBOW는 학습 속도가 빠름

## 동제기반 vs 추론기반

### ① 어휘에 추가할 새 단어가 생겨서 단어의 분산표현을 갱신해야 하는 상황

- 동제기반은 계산을 처음부터 다시해야 함

기존 경험을 학습하지 않아서 효율적으로 학습

- word2vec은 매개변수를 다시학습 - 지금까지의 학습한 기중치를 초기값으로 이용

### ② 분산 표현의 성격이나 정밀도

- 동제기반 개념에서는 주로 단어사이의 웃사성이 인코딩됨

- word2vec은 웃사성 + 복잡한 단어사이의 패턴

ex. king - man + woman = queen

• 두 세계는 특정 조건 하에서 서로 연결되어 있음!

• 우열을 가릴 수 없음

• Glove: 추론기반 + 동제기반, 말뭉치 전처리 통계 정보를 소실함수에 둘째 미니버치로 학습함