10. 워드 임베딩

워드 임베딩(Word Embedding)

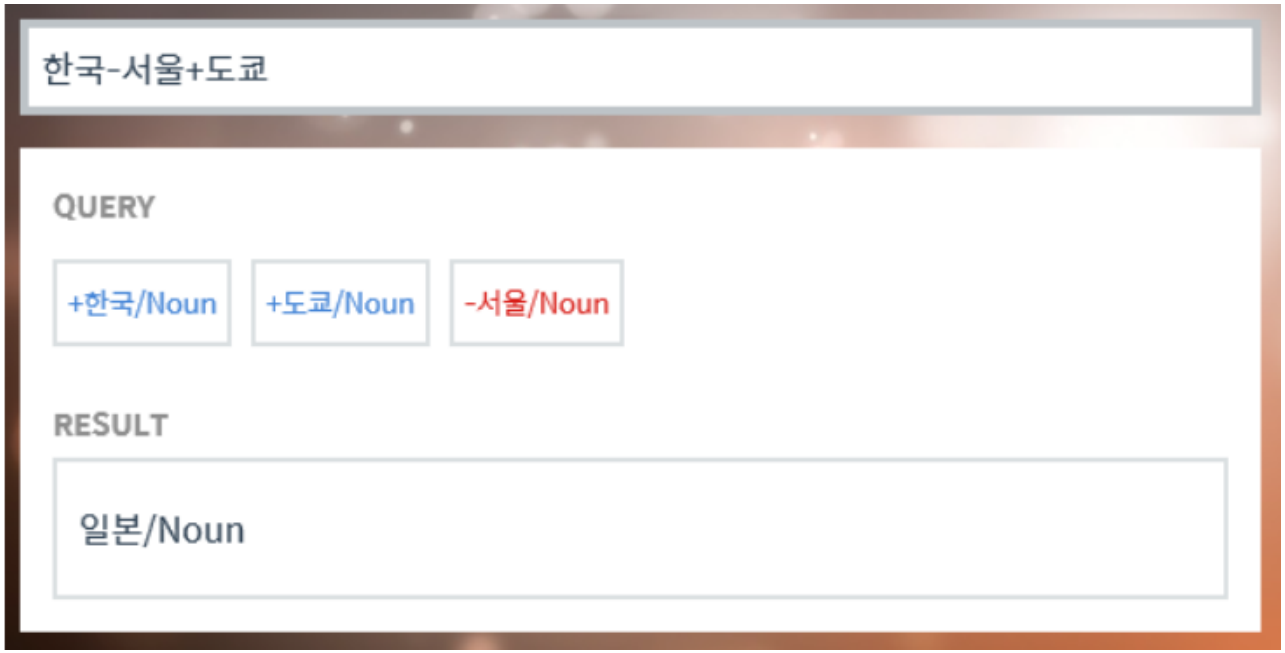
- **Word embedding:** 단어를 밀집 벡터로 표현하는 방법
- **희소 표현 방식**: One-hot vector와 같이 인덱스의 대부분이 0으로 표현됨
  - 예: 강아지 = [0 0 0 0 1 0 0 ... 0]
  - 단어의 개수가 늘어나면 벡터의 차원이 계속 커짐
  - 벡터가 단어의 의미를 표현하지 못함
- **밀집 표현 방식:** 단어를 일정 길이의 벡터로 표시. 각 인덱스는 실수 값을 가짐
  - 예: 강아지 = [0.2 1.8 0.3 0.1 ...] # 벡터 차원은 64, 128, 256 등
- **워드 임베딩:** 단어를 밀집 벡터의 형태로 표현. 밀집 벡터를 **임베딩 벡터**라고 함
  - 워드 임베딩 방법으로 LSA, Word2Vec, FastText, Glove 등이 있음

One-hot vector와 embedding vector의 차이

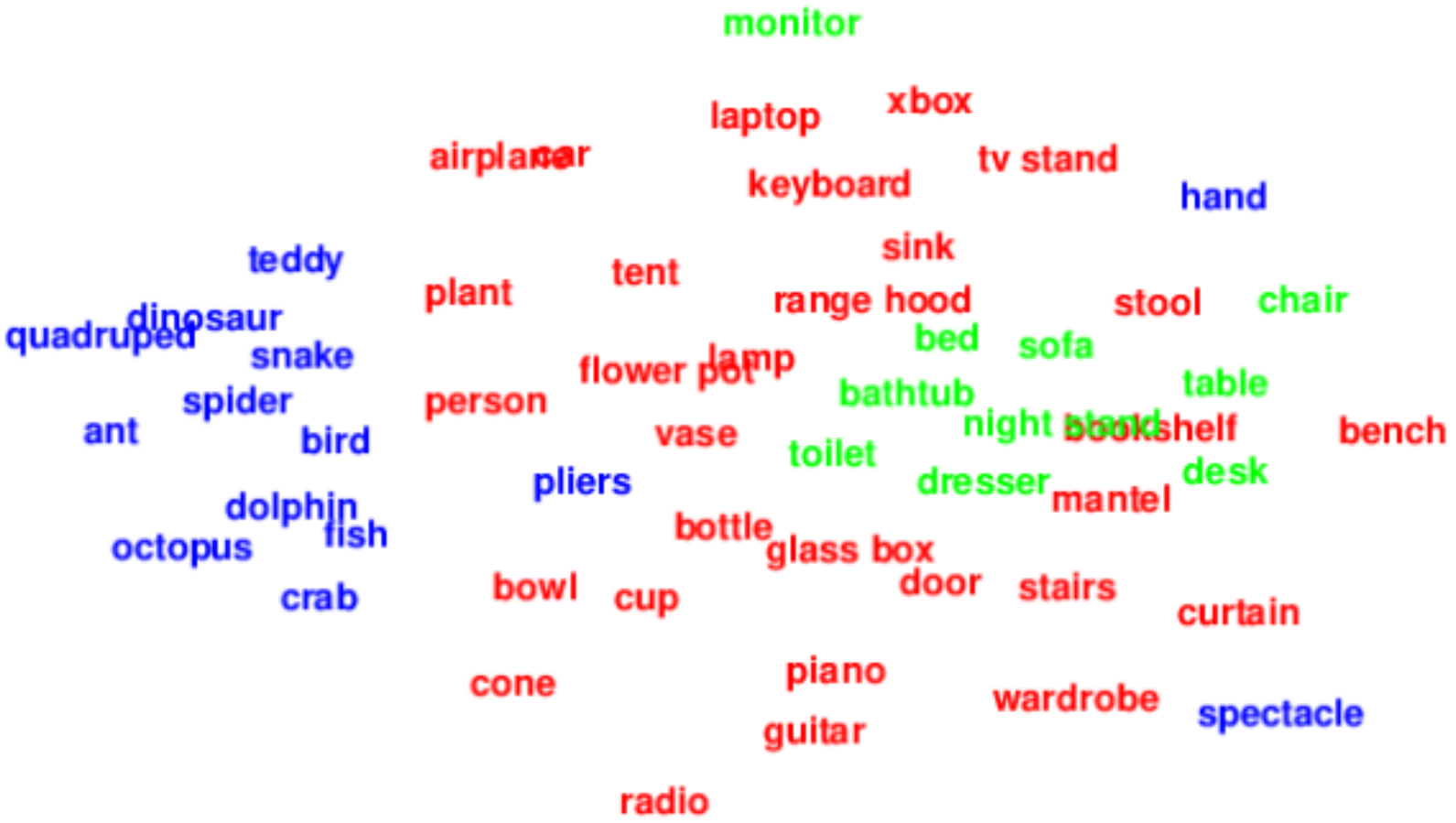
구분	One-hot vector	Embedding vector
차원	고차원(단어 집합의 크기)	저차원
표현 형태	희소 벡터	밀집 벡터
표현 방법	수동	훈련 데이터로 학습
값의 유형	1과 0	실수

Word2Vec

- 단어의 의미를 벡터화하여 단어간 유사도가 반영됨
- 희소 표현과는 다른 분산 표현(Distributed representation)을 보이고 있음
- Word2Vec에는 **CBOW(Continuous Bag of Words)**와 **Skip-gram**이 있음
- 사례: http://w.elnn.kr/search/

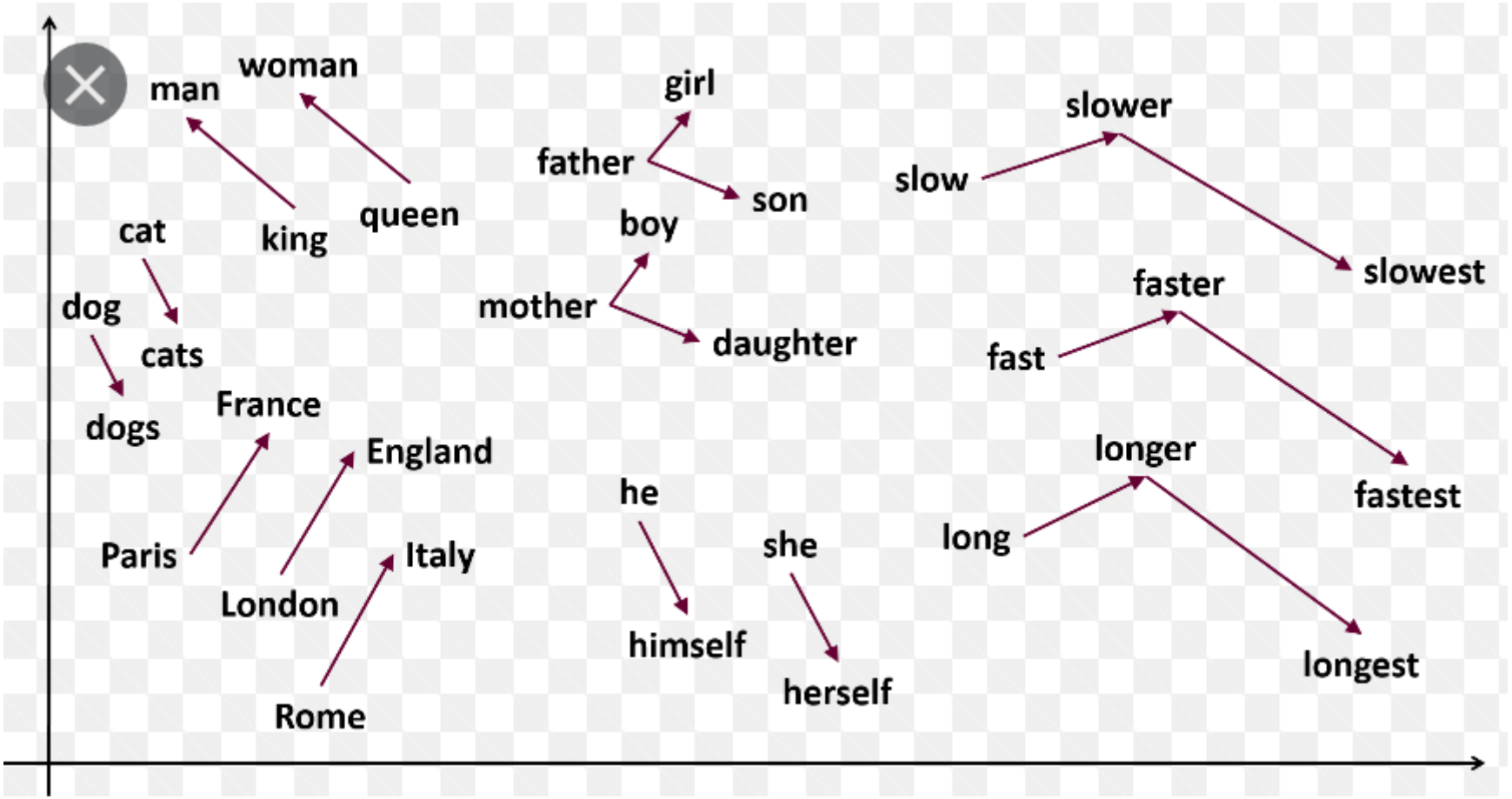


Word embedding 사례



Word embedding 사례

- 단어간의 관계에 따라 유사한 각도가 나타남



CBOW(Continuous Bag of Words)

- 중심 단어의 예측을 위해 주변 단어들을 살펴봄
- 훈련 문장들을 이용하여 신경망을 학습시켜서 Word2Vec을 얻게 됨

중심 단어 주변 단어

The fat cat sat on the mat

The fat cat sat on the mat

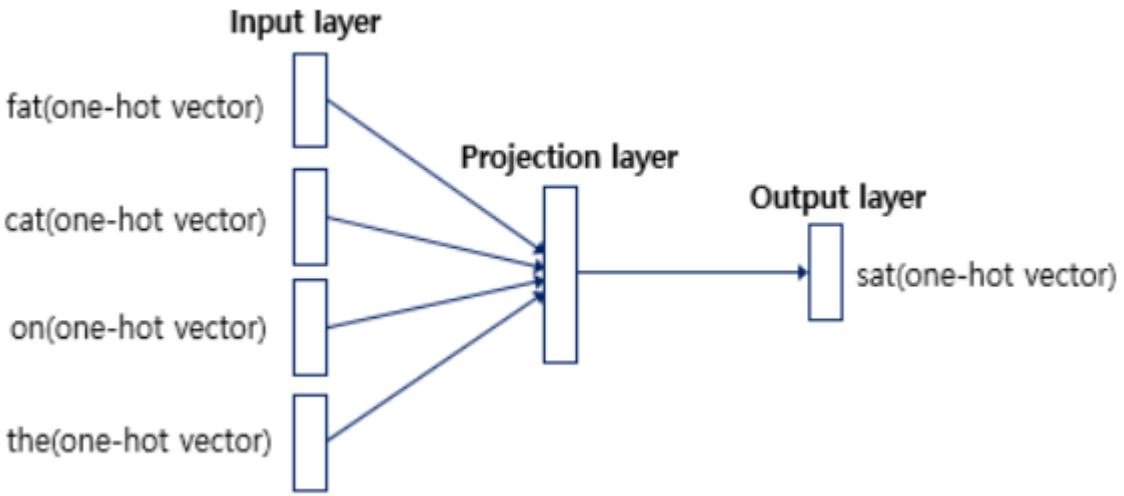
The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

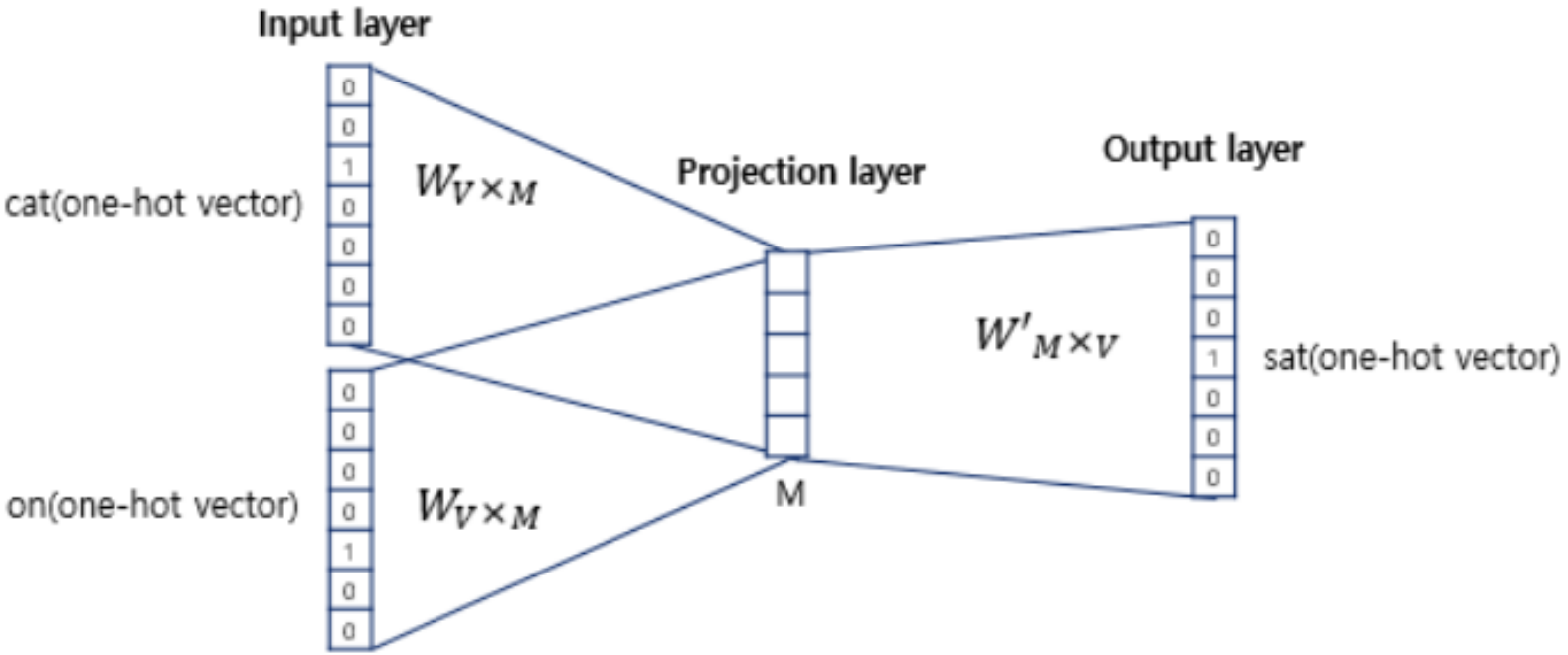
The fat cat sat on the mat

The fat cat sat on the mat



CBOW 학습 신경망 구조

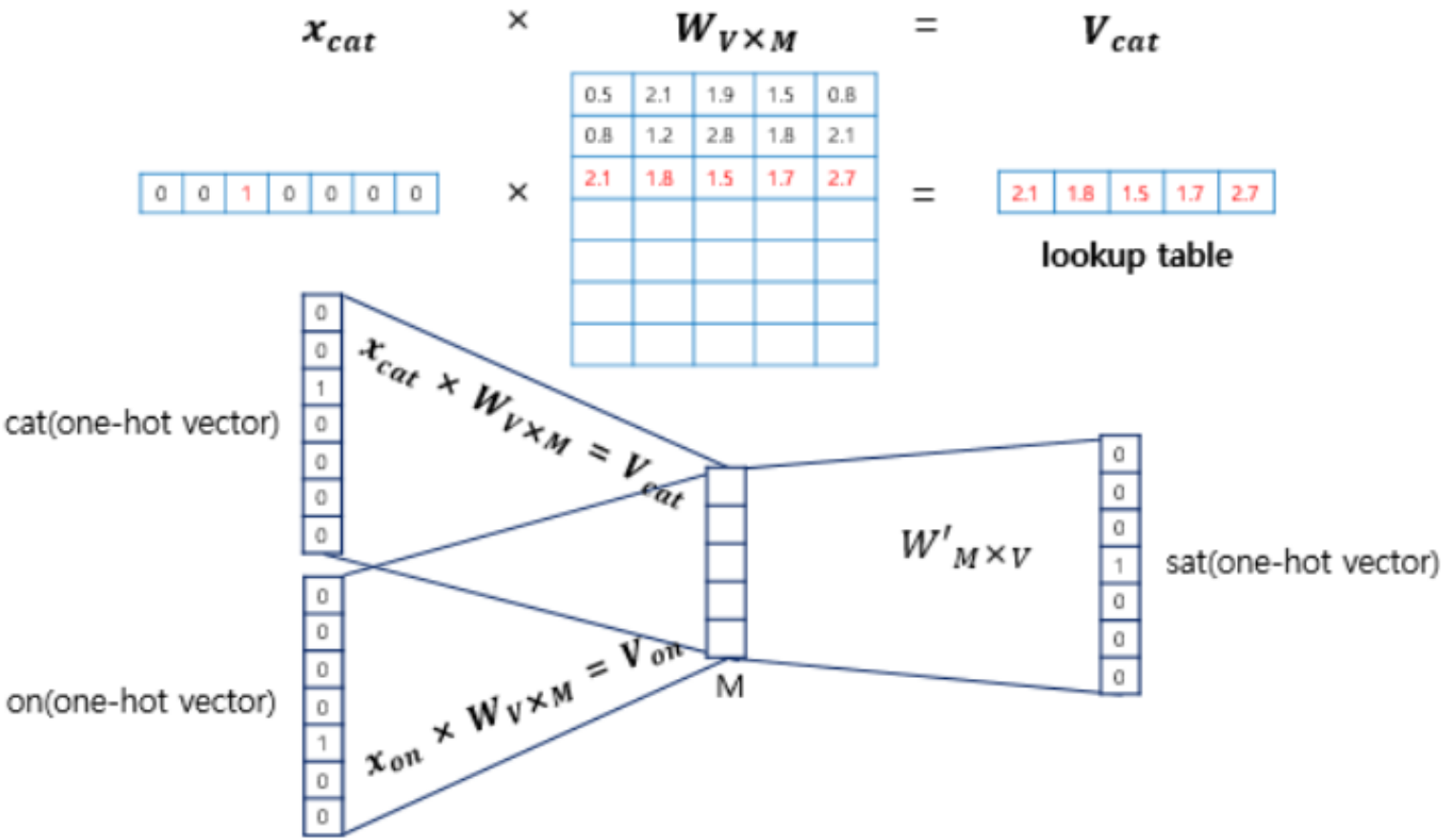
- 입력단과 출력단에서는 단어를 one-hot vector로 표현
- 은닉층은 한 층으로만 구성: 크기는 **임베딩 벡터의 차원인 M**으로 표현



V: 단어 집합의 크기
M: 임베딩 벡터의 차원

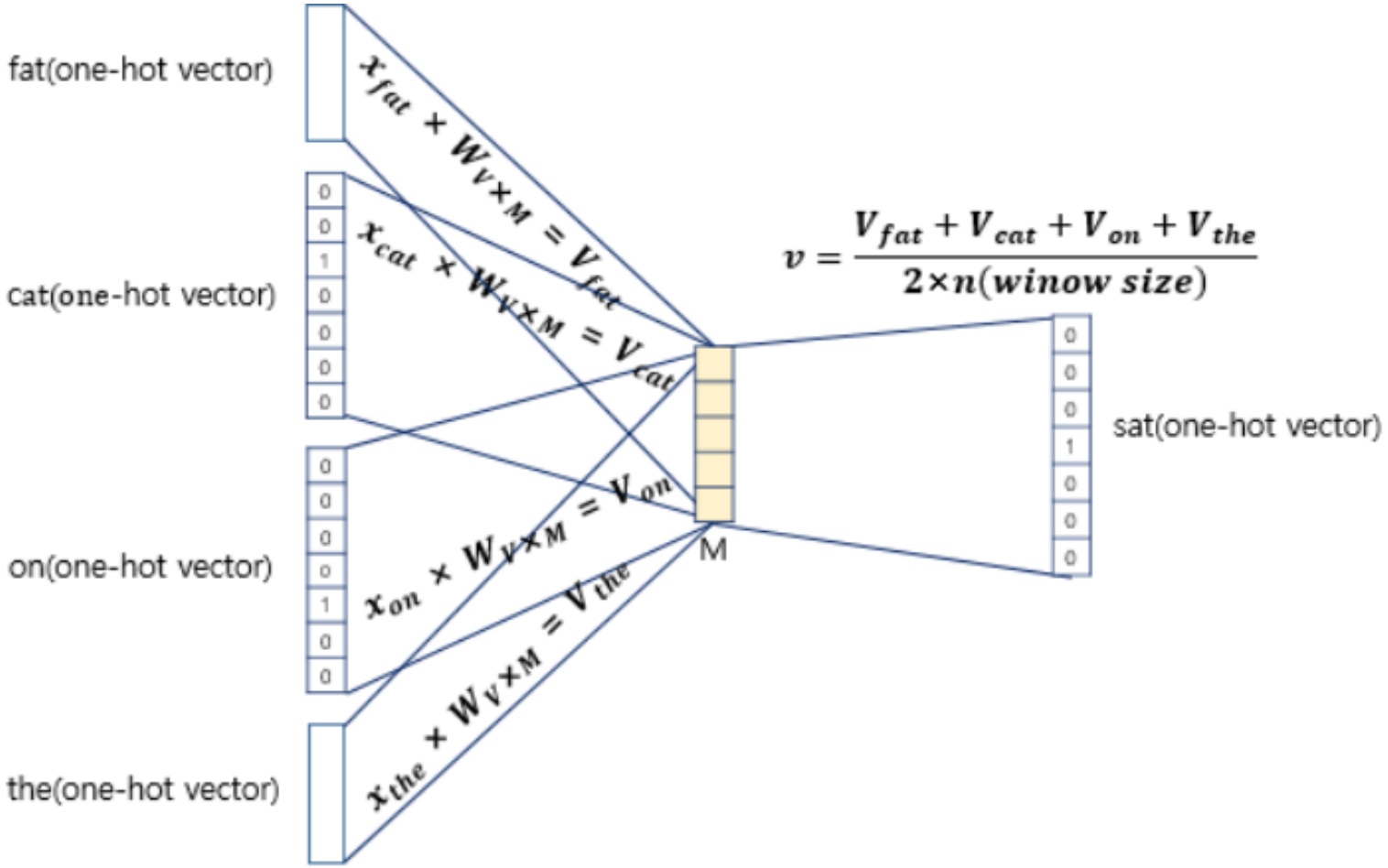
CBOW 순방향 계산

- 주변 단어에 대해 W 행렬의 lookup을 통해 단어 벡터를 계산



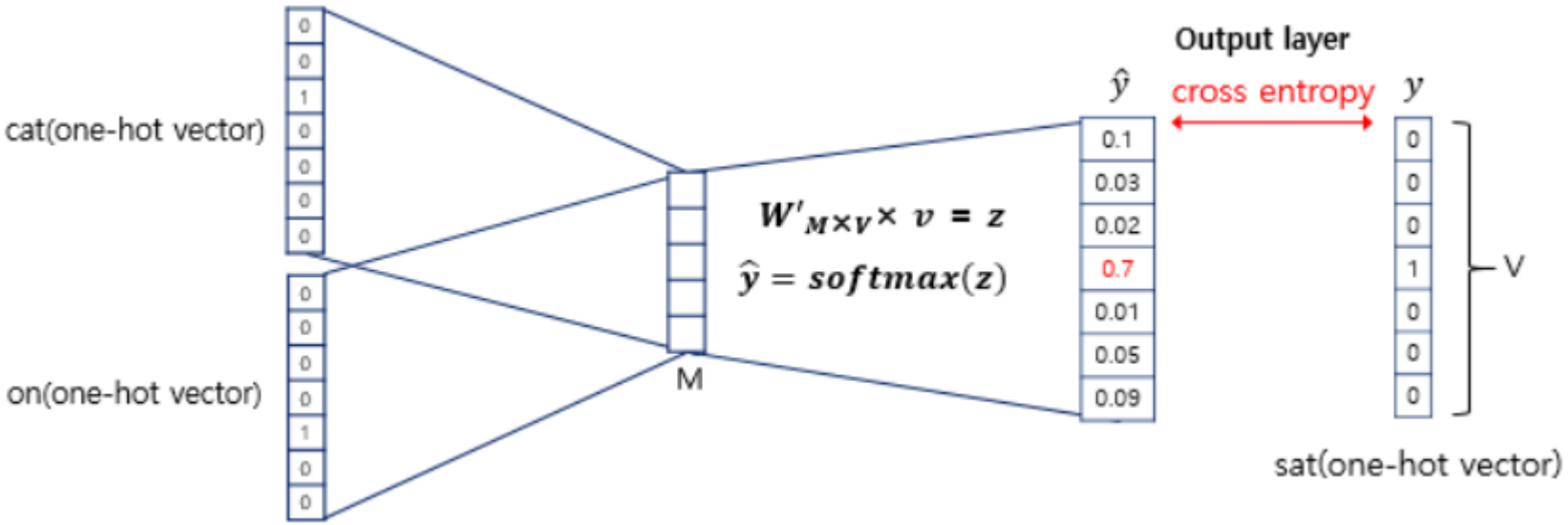
CBOW 주변 단어벡터로 부터 현재 단어벡터 계산

- 주변 단어 벡터의 평균을 현재 단어의 벡터로 예측



CBOW 출력과 오차 계산

- W의 transpose인 W'을 이용하여 $z = W'v$과 $\hat{y} = softmax(z)$ 를 계산
- $\hat{y}$ 과 y 를 이용하여 오차를 계산



$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\widehat{y_j}) = -y_i \log(\widehat{y_i})$$

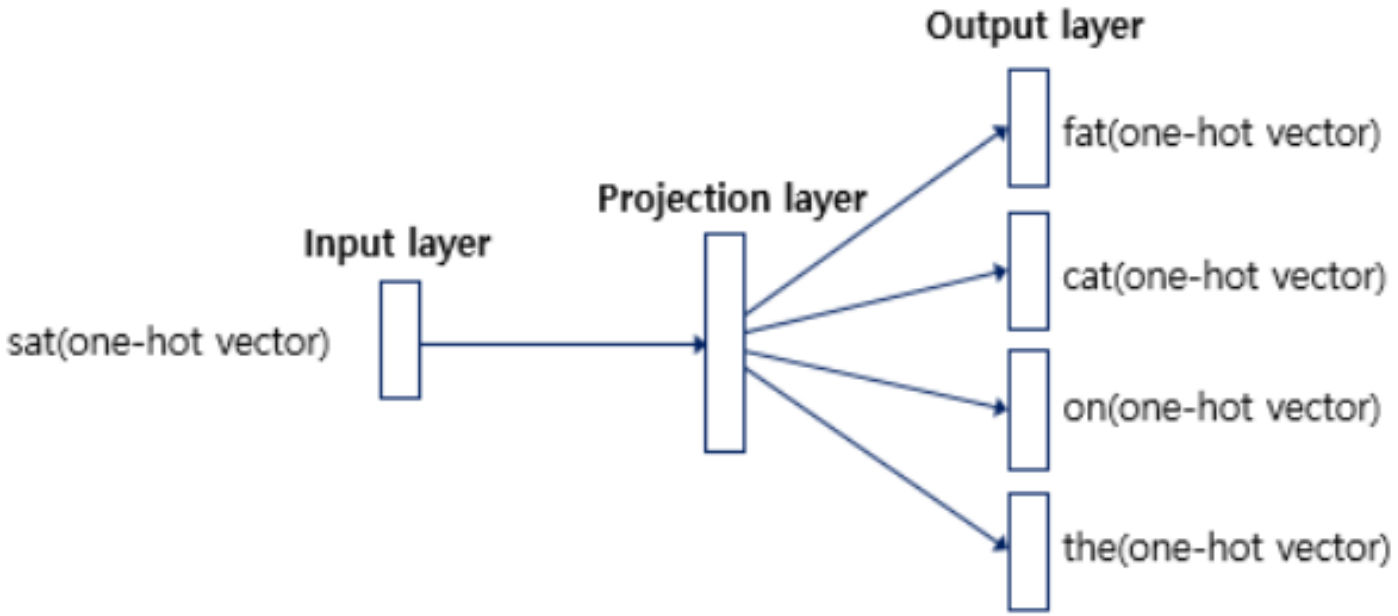
W 의 학습

- 오차 H를 최소화하는 방향으로 W를 학습시킴

$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\widehat{y_j}) = -y_i \log(\widehat{y_i})$$

Skip-gram

- 중심 단어에서 주변 단어를 예측하려는 방식으로 임베딩 벡터를 학습시킴
- Skip-gram이 CBOW보다 성능이 좋다고 알려져 있음



Word2vec 실습

- 각 언어의 코퍼스 자료를 이용하여 임베딩 벡터를 만들어 볼 수 있음
- 한국어의 경우 네이버 영화 리뷰 파일을 이용하여 실험할 수 있음

```
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings.txt", filename="ratings.txt")
```

```
train_data = pd.read_table('ratings.txt')
```

- 위키피디어 한국어 덤프 파일을 이용하는 경우

https://dumps.wikimedia.org/kowiki/latest/

git **clone** https://github.com/attardi/wikiextractor.git

python WikiExtractor.py kowiki-latest-pages-articles.xml.bz2

FastText

- Facebook에서 개발한 단어 임베딩 기법으로 각 단어를 문자 단위 n-gram으로 표현
- 단어 벡터 표현 방식만 다르고 이외의 내용은 Word2Vec과 같음
- Word2Vec은 훈련되지 않은 단어(예: 서울특별시)를 처리하지 못하지만 FastText는 처리할 수 있음

FastText 기본 구조

- 시나브로라는 단어를 tri-gram(n=3)으로 표현하면 다음과 같음 (<, > 는 단어 경계를 나타내는 특수문자임)

  <시나, 시나브, 나브로, 브로>, <시나브로>

- 시나브로 단어 벡터는 다음과 같이 표현

$$\mathbf{u}_{\text{시나브로}} = \mathbf{z}_{<\text{시나}} + \mathbf{z}_{\text{시나브}} + \mathbf{z}_{\text{나브로}} + \mathbf{z}_{\text{브로}>} + \mathbf{z}_{<\text{시나브로}>}$$

$$\mathbf{u}_t = \sum_{g \in G_t} \mathbf{z}_g$$

FastText 훈련 방식

- 각 입력 단어쌍 (t, c)가 대응 관계가 있는 포지티브인지 네거티브인지에 따라 비용을 정의함
- 시나브로가 타깃 단어(t), 쌓였다가 문맥 단어의 포지티브 샘플(c) 이면 <시나, 시나브, 나브로, 브로>, <시나브로> 등의 문자 n-gram 벡터(z) 각각을 쌓였다에 해당하는 단어 벡터와의 유사도를 높임

FastText skip-gram 모델의 유사도 상위 단어

희망	절망	학교	학생	가족	자동차
행복	고뇌	초등	재학생	미혼모	경자동차
희망찬	절망감	중학교	대학생	부부	안전자동차
소망	몸부림친	·중학교	교직원	편부모	승용차
땀방울	슬픔	개교	교내	대가족	상용차
희망특강	상실감	분교	학부모	노부모	경승용차

FastText “하였다”와 유사 단어

[(‘하’, 0.9296),

(‘다’, 0.9073),

(‘했’, 0.8930),

(‘였으며‘, 0.8633),

(‘했으며‘, 0.8549)]

FastText 미등록단어 “서울특벌시”와 유사 단어

[(‘서울색’, 0.7196),
(‘서울한강체’, 0.6617),
(‘서울새남굿’, 0.6590),
(‘철화문’, 0.6521),
(‘서울서체’, 0.6516)]