

Word2Vec & Doc2Vec

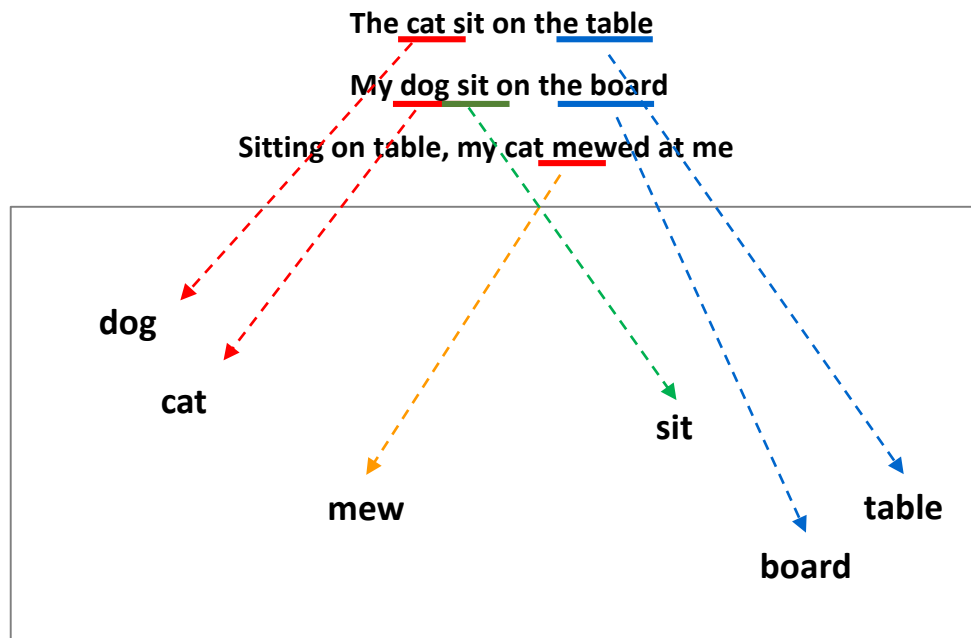
김현중 (soy.lovit@gmail.com)

Embedding

- 임베딩은 x 라는 공간의 데이터에서 **원하는 정보를 잘 저장**하며,
 y 라는 새로운 공간으로 보내는 $f: x \rightarrow y$ 함수
 - (예시) 10,000개 단어로 이뤄진 문서 (1만차원)들의 유사도를 잘 보존하여 2차원으로 보내는 것
 - **어떤 정보를 보존할 것이냐**에 따라서 다양한 임베딩 방법이 존재

Word2Vec

- Word2Vec은 단어 주변에 등장하는 다른 단어들의 분포의 유사성을 보존하는 벡터 표현 방법
 - 주위에 등장하는 단어 분포가 유사한 두 단어는 비슷한 벡터를 지님
'cat' 대신 'dog'을 넣어도 큰 무리가 없음



< 단어의 의미 공간 >

Word2Vec

- Word2Vec은 단어 주변에 등장하는 다른 단어들의 분포의 유사성을 보존하는 벡터 표현 방법
 - 각 컬럼이 어떤 의미를 지니지는 않음
 - 아래 그림은 비슷한 벡터가 비슷한 문맥적 의미를 지니는 것을 표현한 것일 뿐, 모든 차원에 값들이 골고루 차있는 dense vector 형태임

'dog'= [0.31, 0.21, 0.01, 0.01, ...]

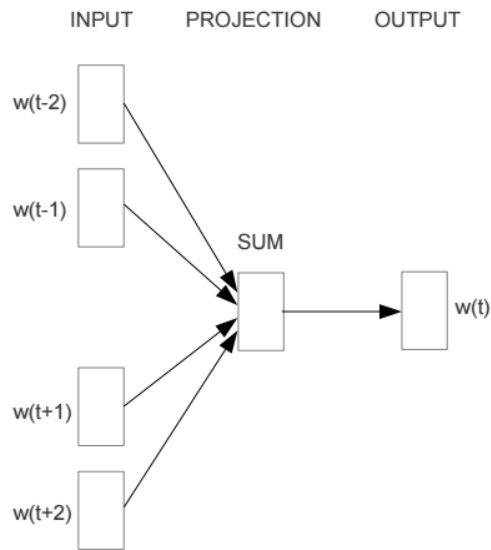
'cat'= [0.45, 0.17, 0.01, 0.01, ...]

'topic modeling'= [0.01, 0.01, 0.22, 0.54, ...]

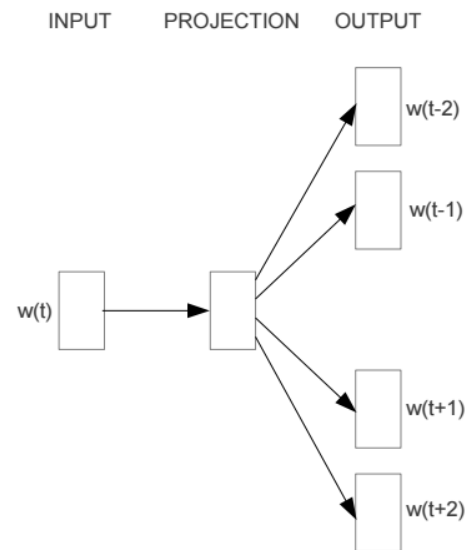
'dim. reduction'= [0.01, 0.01, 0.19, 0.45, ...]

Word2Vec

- Word2Vec은 CBOW, Skip-gram 두가지 형태가 있음
 - CBOW는 주위 단어로 현재 단어 $w(t)$ 를 예측하는 모델
 - Skipgram은 현재 단어 $w(t)$ 로 주위 단어 모두를 예측하는 모델



CBOW



Skip-gram

Word2Vec

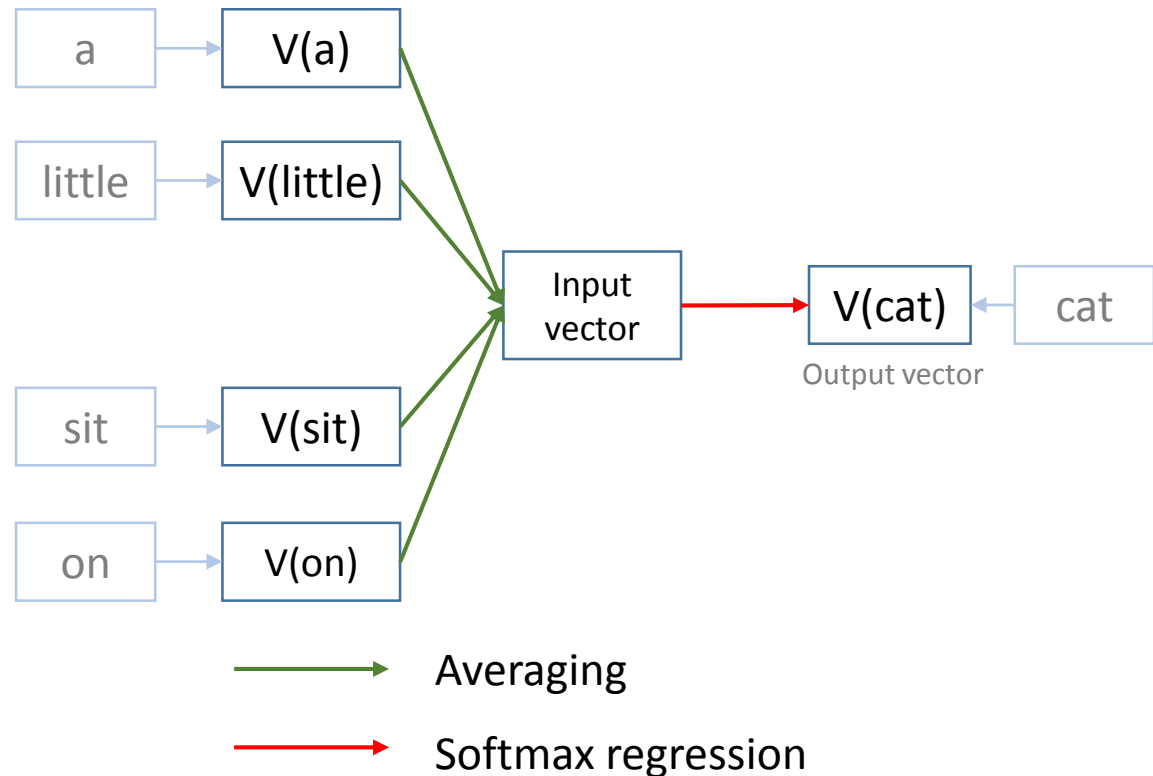
- Word2Vec은 $|V|$ class Softmax regression을 이용하여 각 단어의 벡터를 학습시키는 classifier

Lookup table

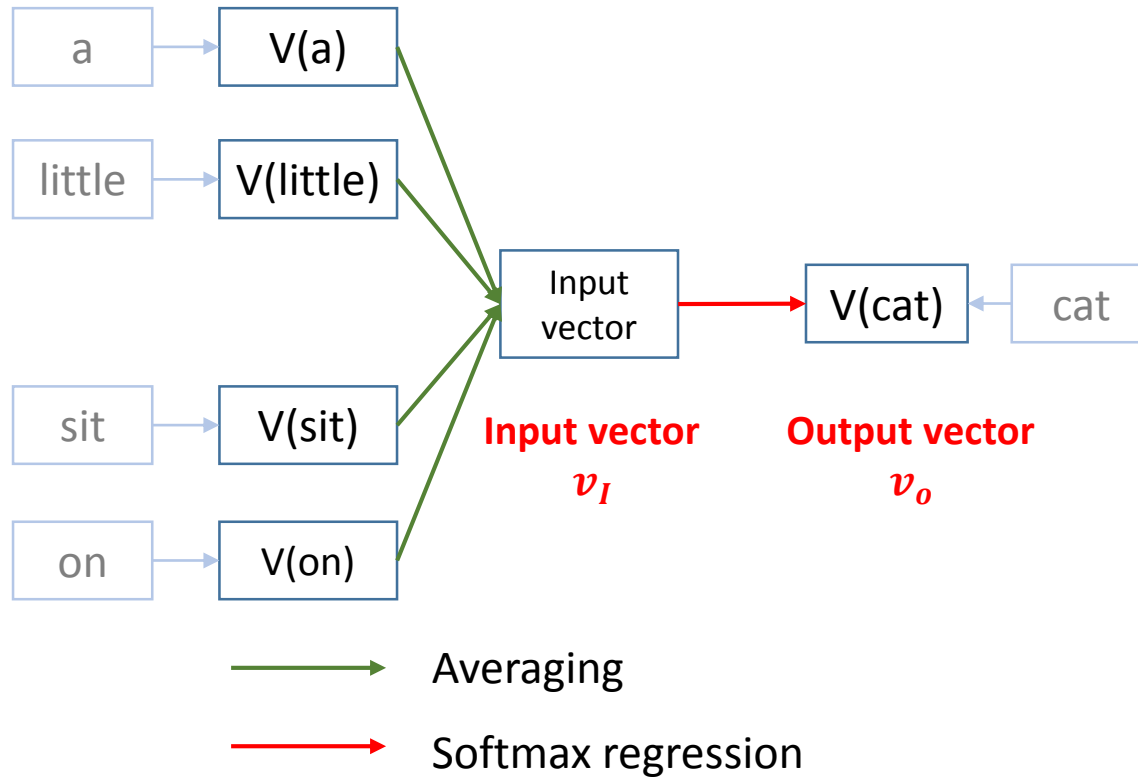
```
{cat: [.31, -.22, .54]  
dog: [.22, -0.3, -0.52]  
on: ...  
the: ...  
little: ...  
.... }
```

Lookup word vector

$V('cat') = [.31, -.22, .54]$



Word2Vec



$$y_j = \frac{\exp(v_I^T v_{O_j})}{\sum_j \exp(v_I^T v_{O_j})}$$

Logistic Regression

- **Logistic Regression (LR)은 대표적인 binary classification 알고리즘**
 - positive class에 속할 점수를 $[0, 1]$ 사이로 표현하기 때문에 확률 모형처럼 이용

$$y_{\theta}(x) = \frac{1}{1 + \exp(-\theta^T x)}$$

- 학습 데이터 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ 가 주어졌을 때, loss function은

$$J(\theta) = - \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right]$$

Logistic Regression

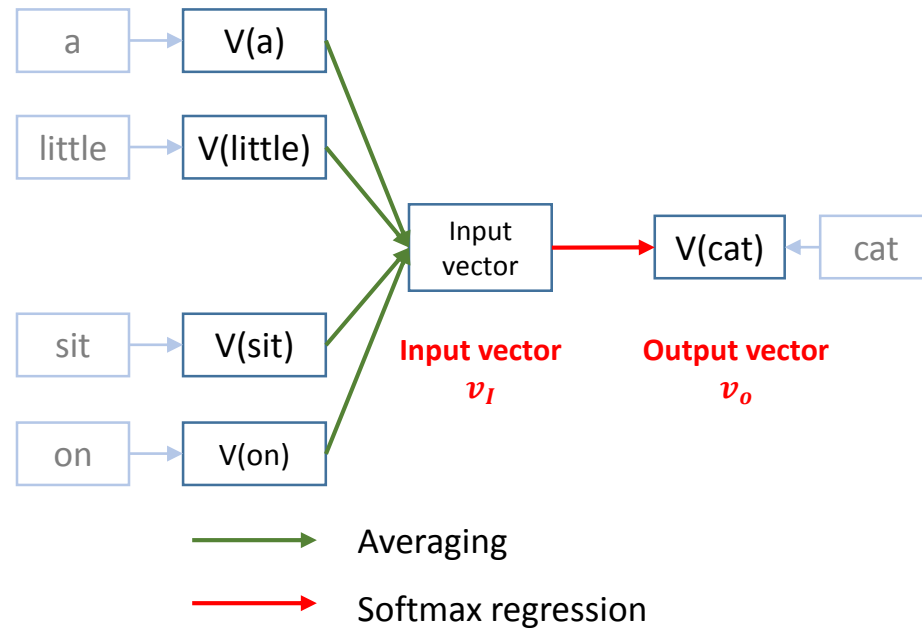
- Softmax regression은 multi class classification 알고리즘으로, Logistic Regression은 Softmax regression의 특별한 경우

$$h_{\theta}(x) = \begin{bmatrix} P(y = 1|x; \theta) \\ \vdots \\ P(y = K|x; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^K \exp(\theta^{(j)T} x)} \begin{bmatrix} \exp(\theta^{(1)T} x) \\ \vdots \\ \exp(\theta^{(K)T} x) \end{bmatrix}$$

- 학습 데이터 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ 가 주어졌을 때, loss function은

$$J(\theta) = - \left[\sum_{i=1}^m \sum_{k=1}^K 1\{y^{(i)} = k\} \log \frac{\exp(\theta^{(k)T} x^{(i)})}{\sum_{j=1}^K \exp(\theta^{(j)T} x^{(i)})} \right]$$

Word2Vec

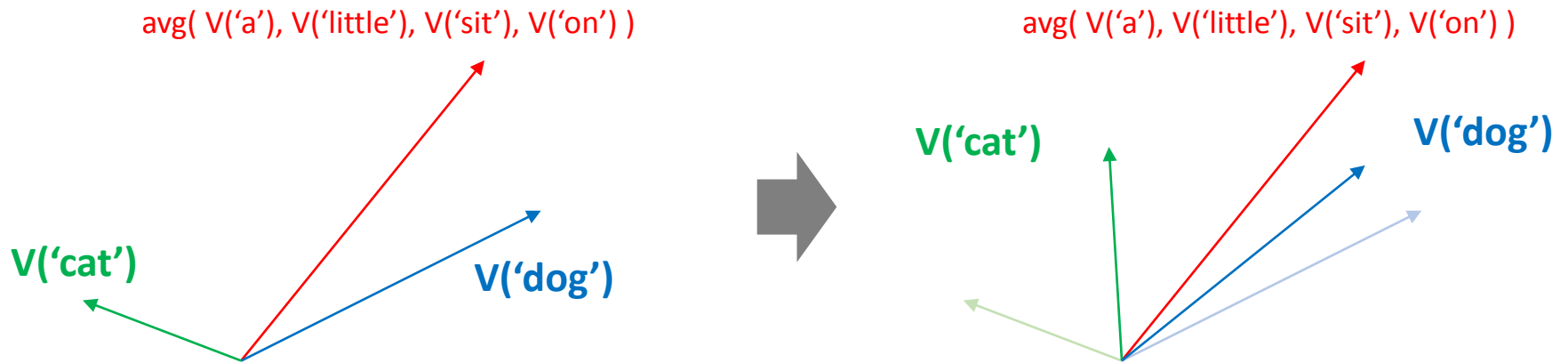


$$h_{\theta}(x) = \begin{bmatrix} P(w_{(t)} = cat) \\ P(w_{(t)} = dog) \\ P(w_{(t)} = table) \\ \dots \\ P(w_{(t)} = Vocab) \end{bmatrix} = \frac{1}{\sum_{j=1}^{|V|} \exp(\theta^{(j)T} x)} \begin{bmatrix} \exp(v(cat)^T v_I) \\ \exp(v(dog)^T v_I) \\ \exp(v(table)^T v_I) \\ \dots \\ \exp(v(Vocab)^T v_I) \end{bmatrix}$$

x : Input vector (average of contextual word vector)

Word2Vec

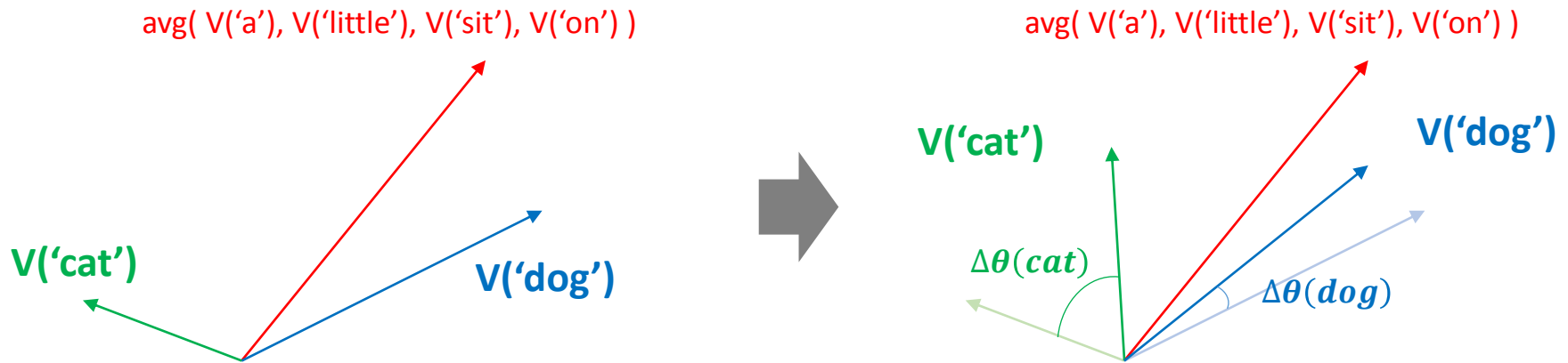
- 'cat'과 'dog'이 비슷한 word embedding vector를 가지는 것은, 두 단어가 비슷한 문맥(contextual word distribution)을 가지기 때문이다
 - 'cat', 'dog'의 두 단어 벡터 모두 context vector에 가깝게 이동



Word2Vec


- 'cat'과 'dog'이 비슷한 word embedding vector를 가지는 것은, 두 단어가 비슷한 문맥(contextual word distribution)을 가지기 때문이다
 - 'cat', 'dog'의 두 단어 벡터 모두 context vector에 가깝게 이동
 - Context vector와 차이가 많이나는 단어 'cat'은 **학습량 (벡터의 변화량)**도 큼

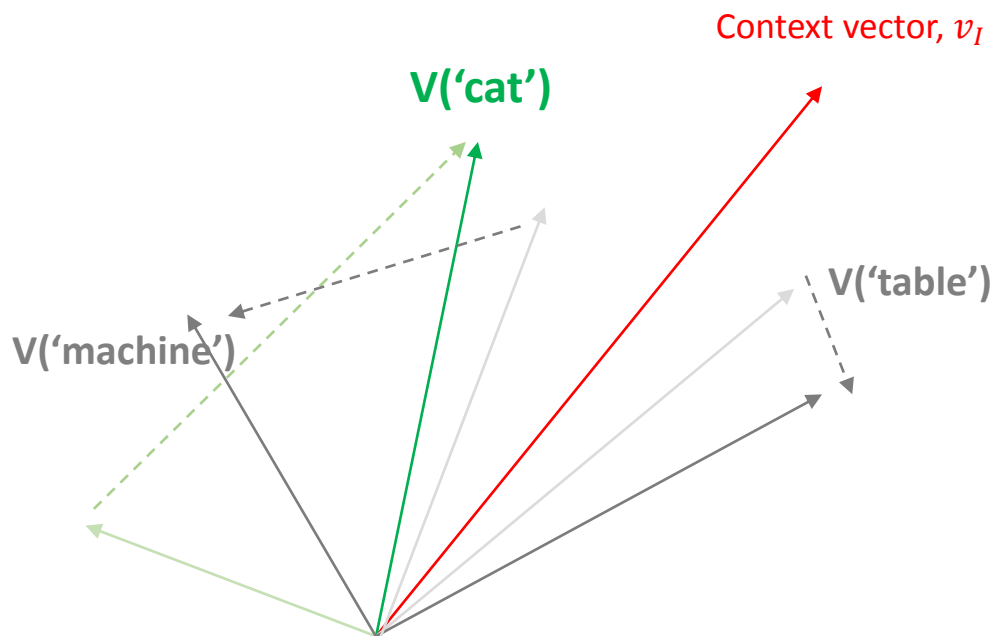
Softmax regression loss



Word2Vec

- Softmax regression은 알맞은 단어는 context vector 방향으로 당기고, 틀린 단어는 반대 방향으로 밀어내는 역할

$$h_{\theta}(x) = \begin{bmatrix} P(w_{(t)} = cat) \\ P(w_{(t)} = dog) \\ P(w_{(t)} = table) \\ \dots \\ P(w_{(t)} = Vocab) \end{bmatrix} = \frac{1}{\sum_{j=1}^{|V|} \exp(\theta^{(j)T} x)} \begin{bmatrix} \exp(v(cat)^T v_I) \\ \exp(v(dog)^T v_I) \\ \exp(v(table)^T v_I) \\ \dots \\ \exp(v(Vocab)^T v_I) \end{bmatrix}$$




Word2Vec

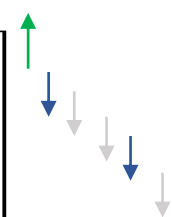
- Softmax regression은 한 단어의 벡터를 학습하기 위해 v 개의 모든 단어의 벡터를 수정하기 때문에 계산량이 매우 큼

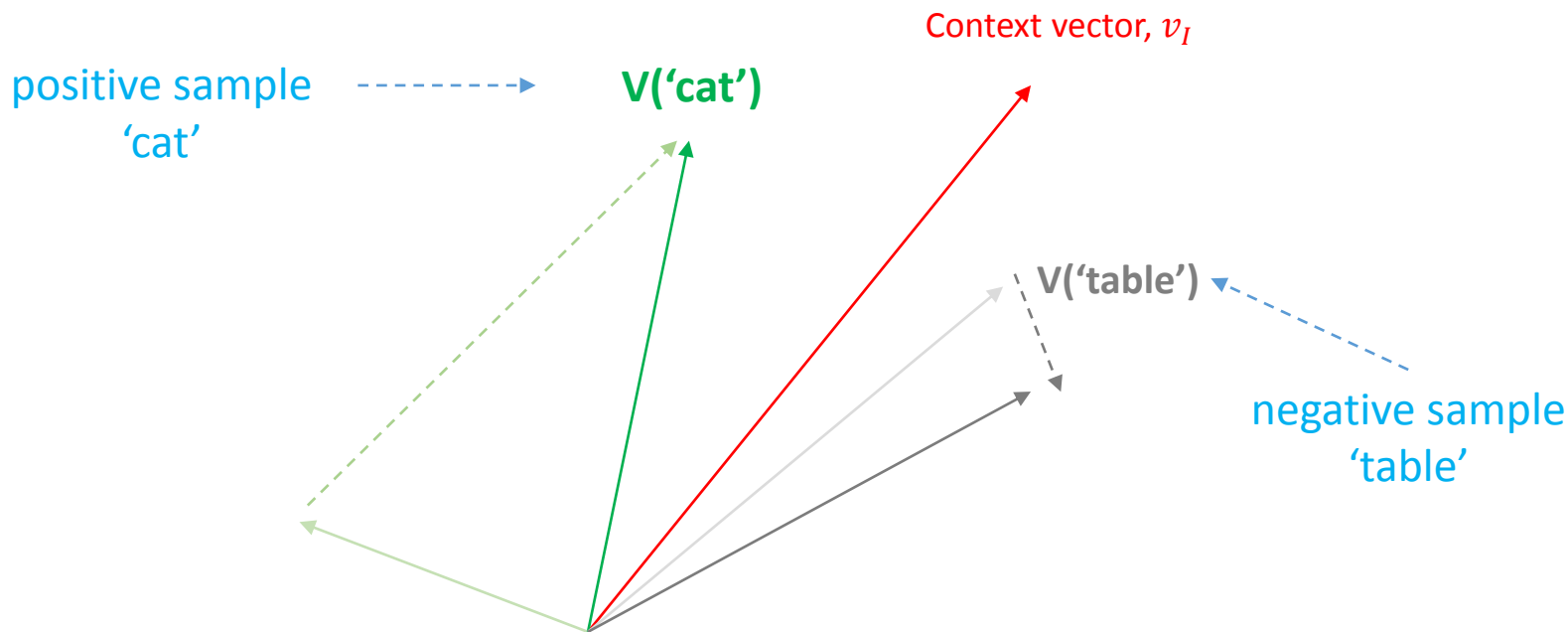
◦ 학습 데이터 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ 가 주어졌을 때, loss function은

$$J(\theta) = - \left[\sum_{i=1}^m \sum_{k=1}^V 1\{w_{(t)} = cat\} \log \frac{\exp(\theta^{(j)T} x^{(i)})}{\sum_{j=1}^K \exp(\theta^{(j)T} x^{(i)})} \right]$$

Word2Vec

- Negative sampling은 cat이 아닌 모든 단어를 밀어내는 것이 아니라, 임의로 샘플링한 단어들에 대해서만 context vector에서 밀어내는 것
 - A positive sample 'cat'과, 수십개 수준의 임의로 선정된 단어 negative samples

$$h_{\theta}(x) = \begin{bmatrix} P(w_{(t)} = cat) \\ P(w_{(t)} = dog) \\ P(w_{(t)} = table) \\ \dots \\ P(w_{(t)} = Vocab) \end{bmatrix} = \frac{1}{\sum_{j=1}^{|V|} \exp(\theta^{(j)T} x)} \begin{bmatrix} \exp(v(cat)^T v_I) \\ \exp(v(dog)^T v_I) \\ \exp(v(table)^T v_I) \\ \dots \\ \exp(v(Vocab)^T v_I) \end{bmatrix}$$


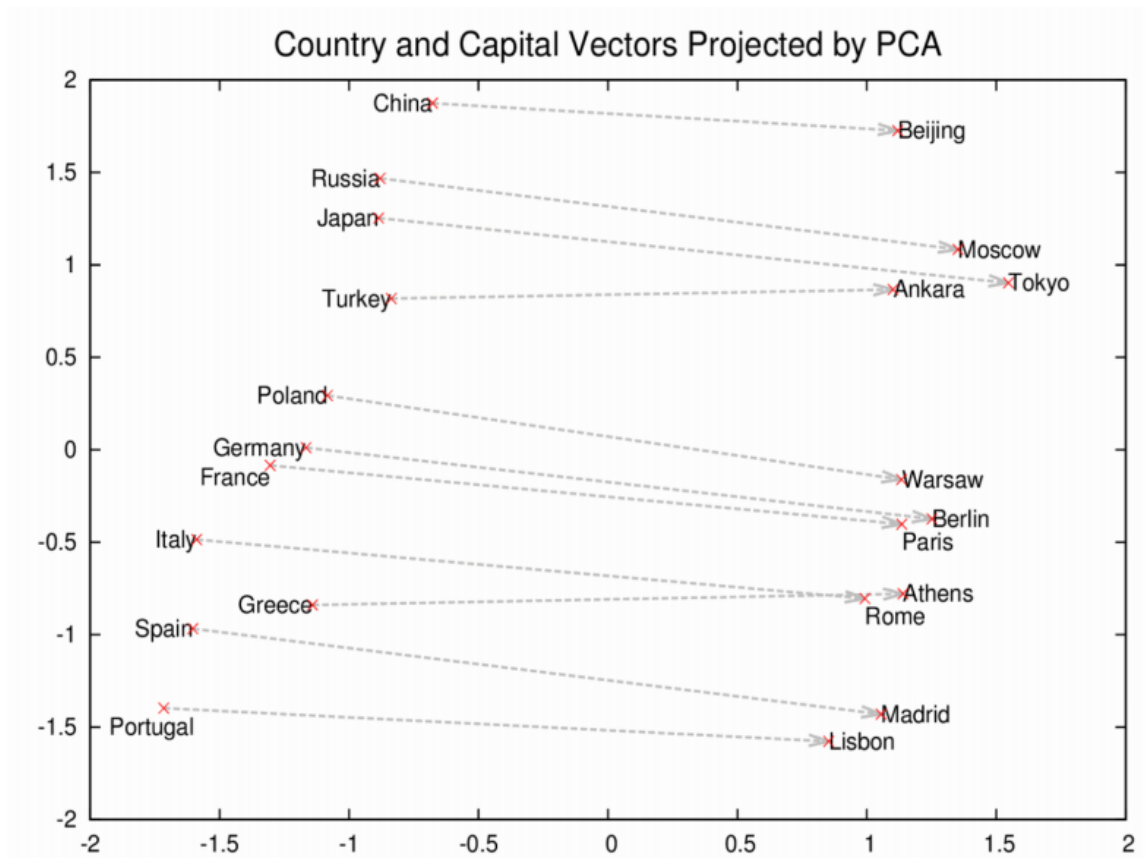


Word2Vec

- ‘cat’이 등장한 많은 문장이 있기 때문에, 한 번에 조금씩 $v(\text{'cat'})$ 을 학습하면 여러 문맥들을 모두 고려할 수 있음.
 - [a, little, cat, sit, on, table],
[my, pretty, cat, ran, out],
....
 - 조금씩 학습한다 = 작은 learning rate를 이용한다

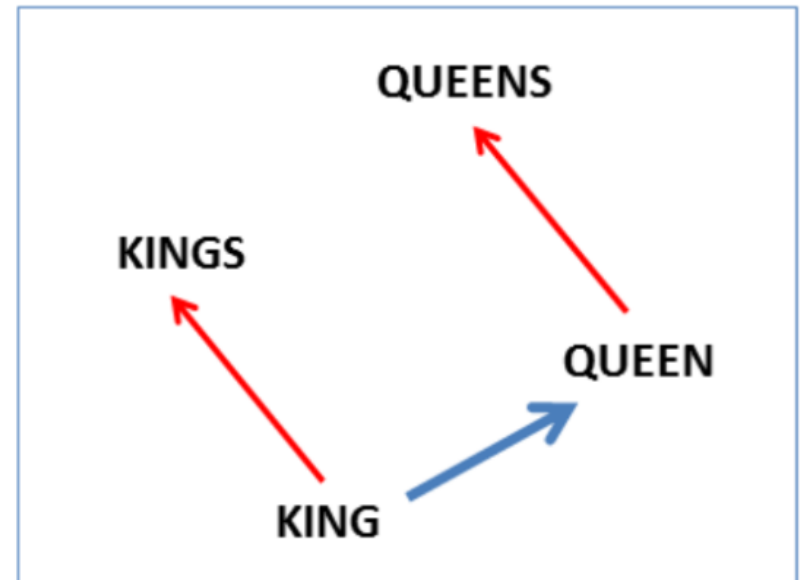
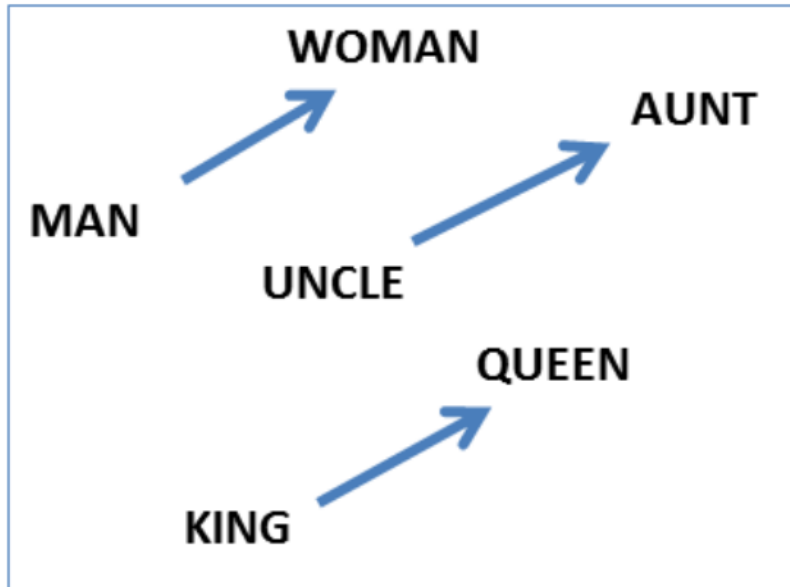
Word2Vec

- Word2Vec은 단어간의 관계성이 추출되는 효과가 있음
 - “ $v(\text{나라}) - v(\text{수도})$ ” 벡터가 나라마다 비슷



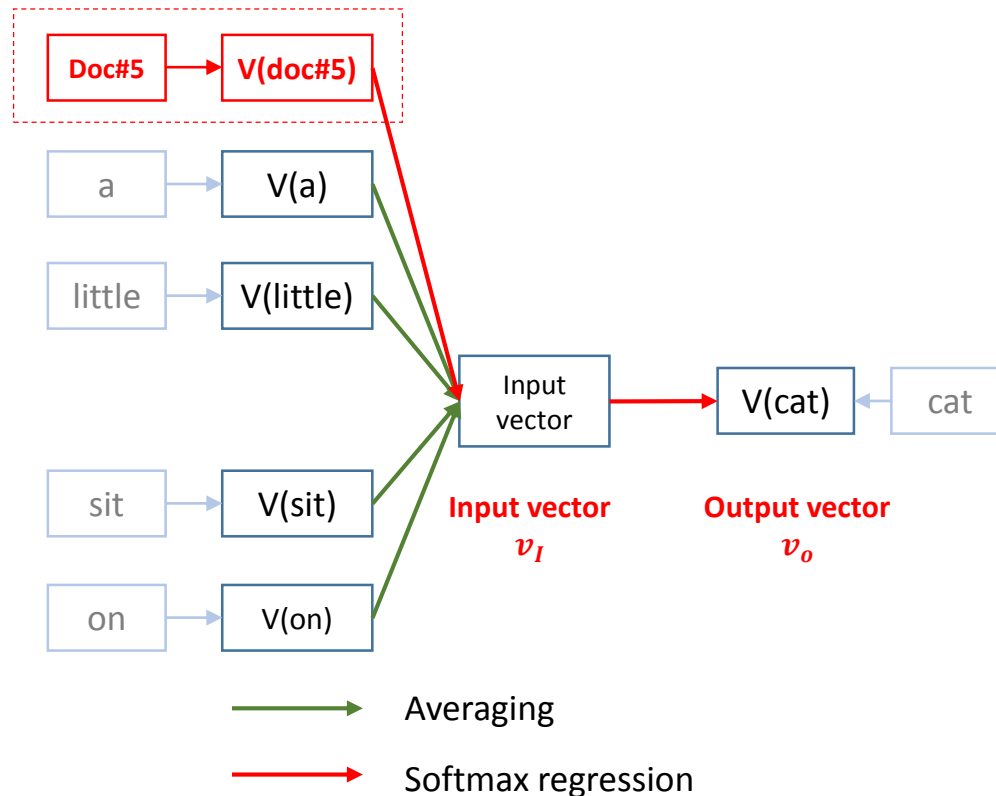
Word2Vec

- Word2Vec은 단어간의 관계성이 추출되는 효과가 있음
 - $V(\text{kings}) - V(\text{king}) = V(\text{queens}) - V(\text{queen})$



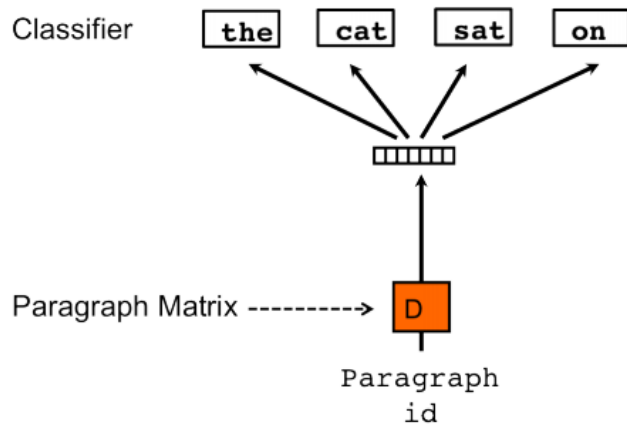
Doc2Vec

- Doc2Vec은 가상의 단어 Document Id를 삽입한 뒤, Word2Vec을 학습하는 것
 - Document id와 word가 같은 embedding 공간에 위치

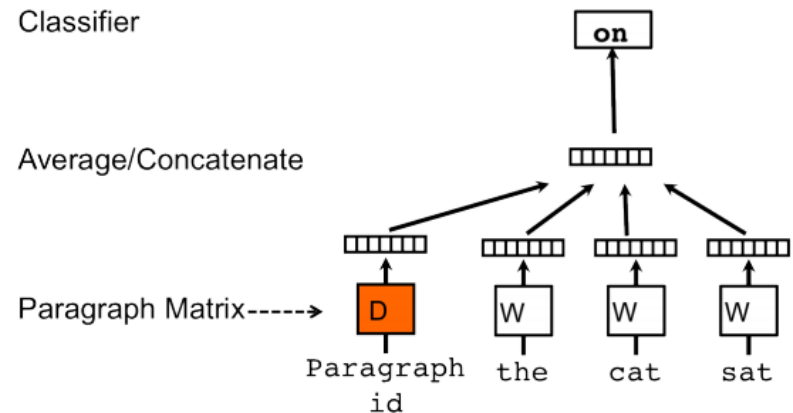


Doc2Vec

- Doc2Vec은 가상의 단어 Document Id를 삽입한 뒤, Word2Vec을 학습하는 것
 - Word2Vec처럼 2가지 모델 버전이 있음



Document id로 모든 단어를 예측하는
Softmax regression을 학습



Document id를 단어처럼 취급하여
the, cat, sat 다음의 단어를 예측하는
Softmax regression을 학습

Doc2Vec

- Doc2vec은 한 문서에 등장한 단어 벡터들의 방향으로 document id vector를 위치시키도록 함

e.g) doc = my pet, a little cat sit on table

