논문 제목 : 벡터 공간에서 단어 표현의 효율적 추정 (2013, Google)

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

==========

자연어 처리 (Natural Language Processing)

- 컴퓨터에게 언어를 학습시키려는 이유

사람들의 의사소통 도구로써 많이 사용되고 있는 것은 언어이며 컴퓨터에게 언어를 학습시킴으로써 사람들의 삶의 질을 더 좋은 방향으로 도와줄 수 있게 하기 위함

- 핵심용어 정의

1. 언어 (Language) :

   언어에 대한 여러 가지 정의 :
   1-1. 사람들이 자신의 생각을 다른 사람들에게 나타내고 전달하기 위해 사용하는 체계
   1-2. 사물, 행동, 생각, 그리고 상태를 나타내는 체계
   1-3. 사람들 사이에 공유되는 의미들의 체계
   1-4. 문법적으로 맞는 말의 집합
   1-5. 언어 공동체 내에서 이해될 수 있는 말의 집합

   - 언어는 "자연어"와 "인공언어"로 분류할 수 있음

1. 자연어 (Natural Language) :

   - 사람들이 일상적으로 쓰는 "언어"를 인공적으로 만들어진 언어인 "인공언어"와 구분하여 부르는 개념
   - 인류의 각 민족이 오래전부터 생활 속에서 사용해 왔던 언어
   - 어떤 정돈된 완벽한 문법이나 형식적인 의미가 없는 언어

1. 인공언어 (Constructed Language, Conlang(콘랭)) :

   - 한 사람 혹은 여러 사람의 의도와 목적에 따라 만든 언어(예 : 프로그래밍 언어, 수학 기호, 음악 기호 등)

1. 자연어 처리 NLP(Natural Language Processing) :

   - 인간의 언어를 컴퓨터가 이해하고 처리할 수 있는 형태로 변환하는 기술

1. 언어 모델 LM(Language Model) :

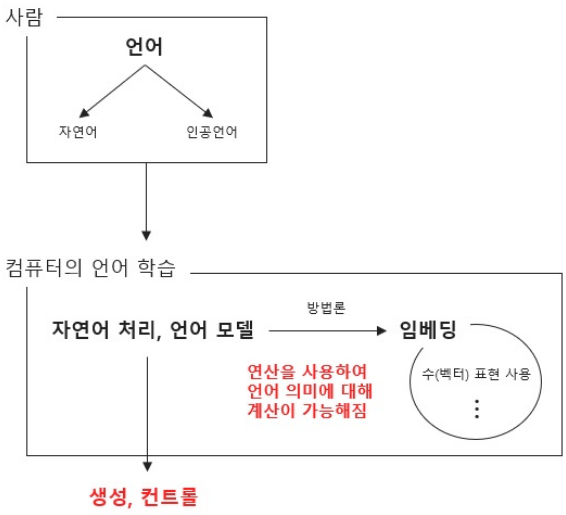
   - 문장이나 문서 등의 자연어 데이터를 분석하고, 다음 단어 또는 문장이 등장할 확률을 계산해주는 모델

1. 임베딩 (Embedding) :

   - 한 도메인에서 정의된 개체를 다른 도메인으로 매핑하는 기술
   - 자연어 처리 분야에서는 주로 단어나 문장 등을 벡터 형태로 변환하는 과정을 일컬음(유사성 등을 연산하기 위함)

1. 워드 임베딩 (Word Embedding) :
    - 단어를 벡터 형태로 변환하는 과정

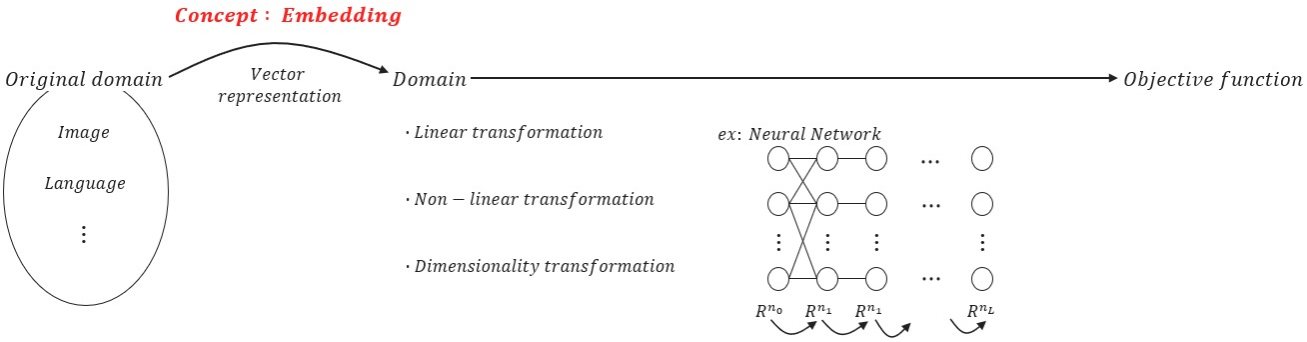
용어간 관계 그래프



- 자연어 처리 목표는 컴퓨터에게 언어를 학습시켜서 상황에 맞게 적절한 언어를 생성하고 컨트롤하기 위함
- 현재까지 이뤄진 수준으로 보면
- 컴퓨터에게 언어를 어떻게 학습시킬래?
    1. 임베딩 개념을 사용하여 언어-> 수 표현으로 바꾸자(연산이 정의되어 있으니 의미를 계산해볼 수 있음)
    2. 오늘날은 함수 근사화 방법인 Deep Neural Network 방법을 사용함

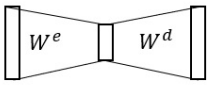
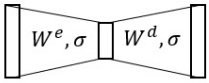
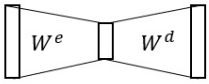
좀 더 근본적으로 접근해보면

- Embedding 이란 개념은 자연어뿐만 아니라 이미지와 같은 다른 도메인에도 있는 개념임
- Embddding을 수 혹은 벡터로 매핑하여 표현함(연산이 정의되어 있으므로, 이를 사용하여 original domain의 의미를 다뤄보려함)
- 수 혹은 벡터로 어떻게 표현할 건지는 연산에 따라 달라짐(선형변환, 비선형변환, 차원변환의 조합)
- 오늘날에는 Deep Neural Network(선형변환, 비선형변환, 차원변환) 을 주류로 사용하고 있음
- 그리고 더해서, 목적(손실)함수를 어떻게 정의하냐에 따라 Convolutional Neural Network, Auto-encoder, Transformer 그리고 Machine translation, Text generation 등처럼 다양하게 불림



$-$ $Original\ domain, Vector\ representation, Objective\ function$: 이 3가지가 무엇으로 결정되는지에 따라 특정한 이름이 붙는 형태임

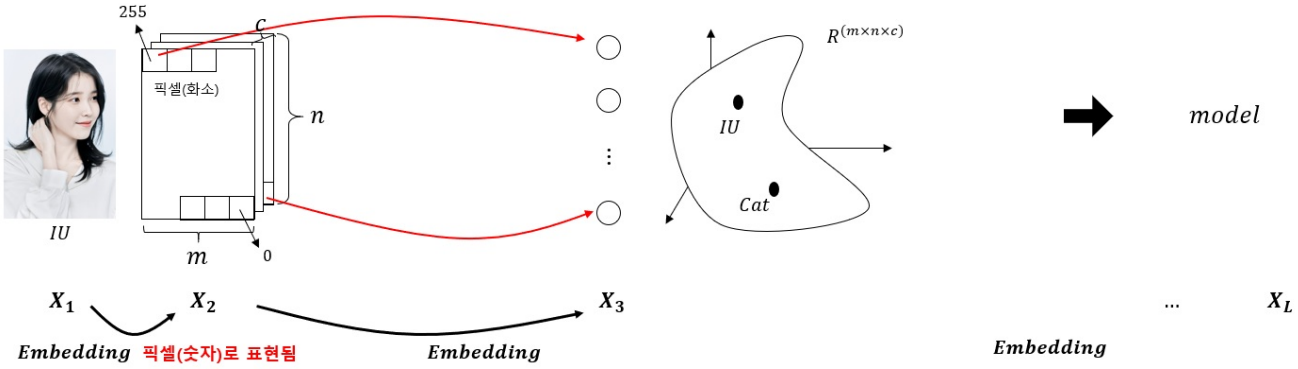
- Original domain, Vector representation, Objective function 관점으로 해석해본 모델들

Original domain	Vector representation (Architecture)	Objective(Loss) function	Name
–	· Linear transformation · Dimension transformation 	Sample variance 를 최대화	PCA
–	· Linear transformation · Non – linear transformation · Dimension transformation 	입력과 같은 형태의 출력 간의 차이 최소화	Auto – encoder
Language	· Linear transformation · Dimension transformation 	Context word 평균과 Center word의 차이	Word2Vec CBOW

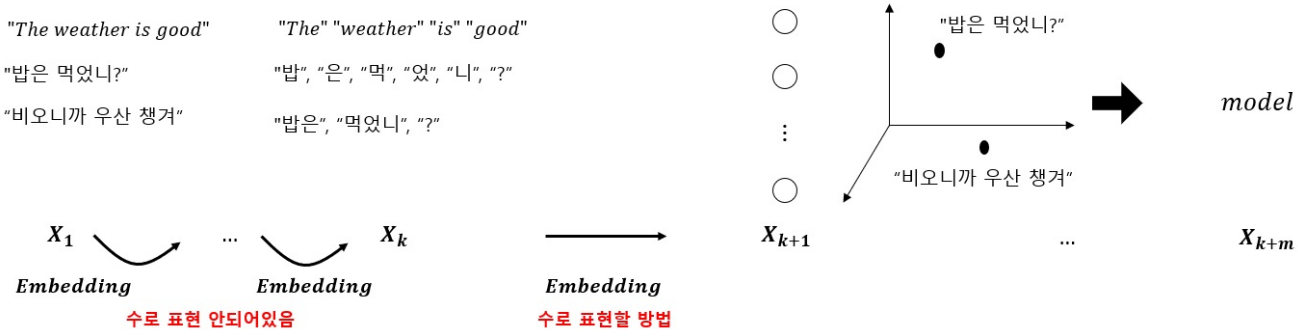
- 자연어 처리 특성

- 이미지 처리와 다르게 자연어 처리는 바로 수로 표현되지 않으므로 이를 변환하는 과정이 추가적으로 필요함

이미지 처리



자연어 처리



==========

논문 제목 : 벡터 공간에서 단어 표현의 효율적 추정 (2013, Google)

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

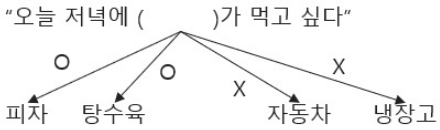
Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

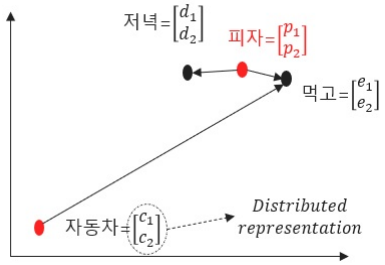
- 논문 링크 : https://arxiv.org/pdf/1301.3781.pdf

- 워드 임베딩의 핵심 단어 정의 :

1. Distributional hyphothesis(분포 가설) :
   "단어의 의미는 주변 단어들과의 관계에 의해 형성된다" 는 주장
   예) "오늘 저녁에 (__)가 먹고 싶다" 문장에는 주변 단어들에 의해 음식(피자, 탕수육 등)이 등장할 확률이 높음



1. Distributed representation(분산 표현) :
   하나의 개체를 나타내는 정보를 여러 개의 요소로 분산시켜 표현하는 방법



- 핵심 요약 :

1. Word2Vec 은 분포가설에 기반한 워드임베딩 방법론 중 하나임

2. Word2Vec 은 Google에서 개발한 방법으로 CBOW와 Skip-gram 모델 두 가지를 제안함

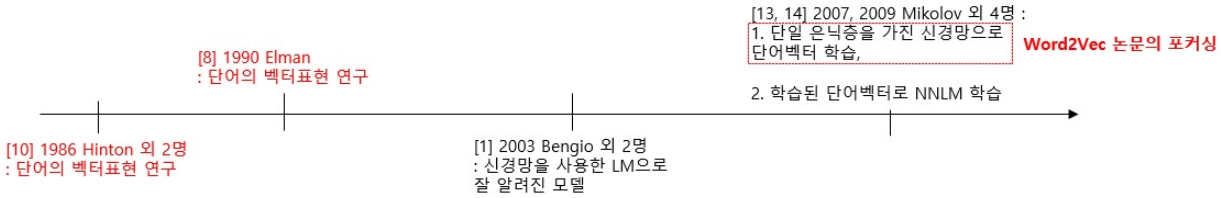
3. 이들 모델은 단어를 저차원 공간 상의 밀도 있는 벡터로 표현하고 단어 간의 유사성 및 관련성을 측정하려고 고안됨

- Introduction

- 현재 대부분의 자연어처리에서는 단어간 유사도를 고려하지 않는 원핫인코딩과 같은 희소표현을 사용하고 있음
- 대표적으로 통계적 언어 모델인 N-gram이 있음
- 기계학습의 발전으로 큰 데이터셋에 대해 복잡한 모델 학습이 가능해졌고 일반적으로 이전에 제안된 간단한 모델의 성능보다 좋음
- 신경망 기반 방식의 언어 모델이 기존 통계 방식 언어 모델(N-gram) 보다 좋았다는 연구가 있음

- Previous Work

- word embedding 관점에서



[ ] 논문의 참고문헌 번호임

- CBOW(Continuous Bag-Of-Words)

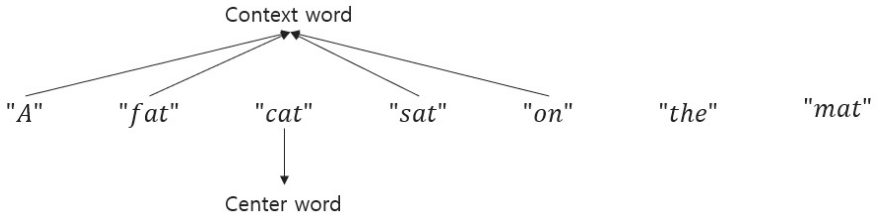
핵심단어 :

- center word : 관심있는 단어
- context word : 관심있는 단어의 주변 단어
- word embedding : 단어를 벡터로 변환하는 기법으로, 여기서는 n^* 차원으로의 벡터 변환을 word embedding 이라 하겠음, 즉 n 차원의 희소한 공간보다 작은 저차원 공간 상의 밀도 있는 벡터로의 변환 과정을 의미

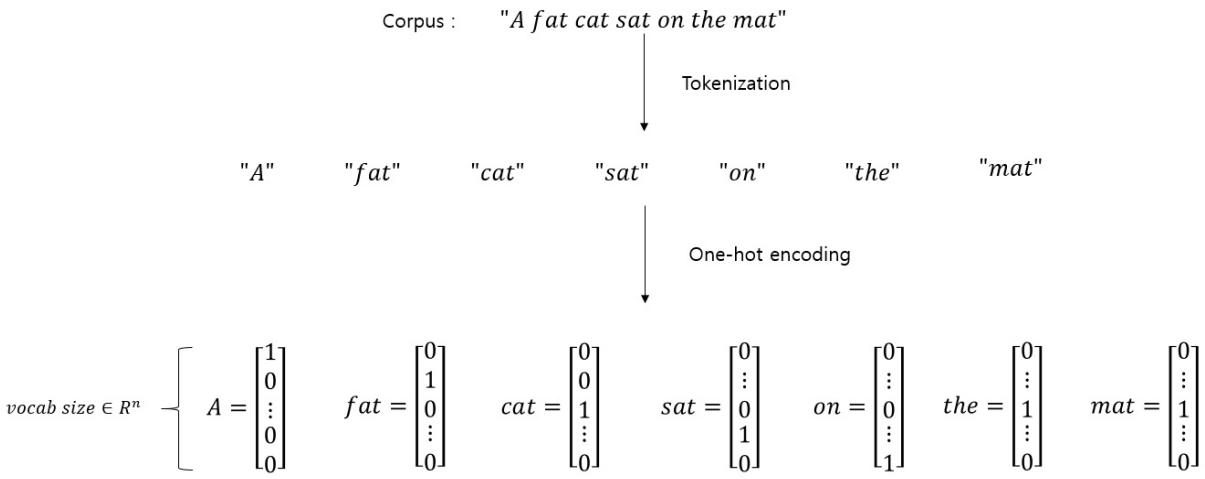
CBOW 아이디어의 핵심 :

-> 주변 단어들의 word embedding 벡터를 통해 관심있는 단어의 word embedding 벡터를 표현하고자함

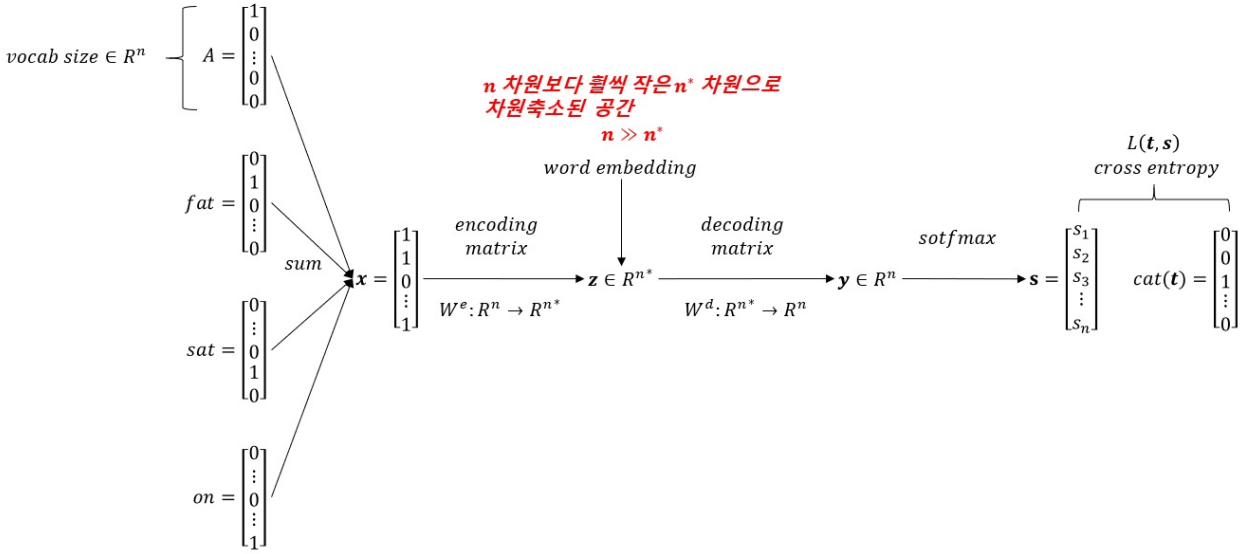
center word와 context word 예시 :



기본 상황 :



- CBOW model architecture



Notation :

CBOW의 목적함수 : $\min L(\boldsymbol{t}, \boldsymbol{s}) = \min -\sum_{i=1}^{n} t_i \log s_i$

$$\boldsymbol{s} = softmax\left(W^d(W^e\boldsymbol{x})\right)$$

$$\boldsymbol{y} = W^d\boldsymbol{z}$$

$$\boldsymbol{z} = W^e\boldsymbol{x}$$

$\boldsymbol{t}$: n차원으로 one-hot encoding된 center word 벡터(cat), $\boldsymbol{t} = \begin{bmatrix}0\\0\\1\\\vdots\\0\end{bmatrix}$

$(n \times 1)$

$\boldsymbol{x}$: n 차원으로 one-hot encoding된 context word 벡터들의 합, $\boldsymbol{x} \begin{bmatrix}1\\1\\0\\\vdots\\1\end{bmatrix} = \text{A} \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix} + \text{fat} \begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix} + \text{sat} \begin{bmatrix}0\\\vdots\\0\\1\\0\end{bmatrix} + \text{on} \begin{bmatrix}0\\\vdots\\0\\\vdots\\1\end{bmatrix}$

$(n \times 1)$

W^e : n 차원에서 n^* *word embedding* 차원으로 선형변환시키는 *encoding matrix* $W^e = \begin{bmatrix} w^e_{11} & w^e_{12} & \cdots & w^e_{1n} \\ w^e_{21} & w^e_{22} & \cdots & w^e_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w^e_{n^*,1} & w^e_{n^*,2} & \cdots & w^e_{n^*,n} \end{bmatrix}$

$(n^* \times n)$

W^d : n^* *word embedding* 차원에서 n 차원으로 선형변환시키는 decoding *matrix* $(W^e)^T = W^d$

$(n \times n^*)$

- CBOW 계산 과정과 의미해석

1. word embedding 단계

$$W^e \boldsymbol{x} = \boldsymbol{z}$$

$(n^* \times n)$ $(n \times 1)$ $(n^* \times 1)$

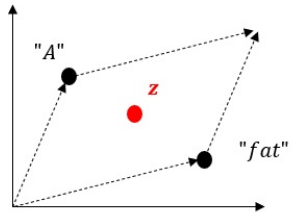
(평균화 시켜줌)

$$= \begin{bmatrix} w^e_{11} & w^e_{12} & \cdots & w^e_{1n} \\ w^e_{21} & w^e_{22} & \cdots & w^e_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w^e_{n^*,1} & w^e_{n^*,2} & \cdots & w^e_{n^*,n} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} w^e_{11} + w^e_{12} + w^e_{1s} + w^e_{1o} \\ w^e_{21} + w^e_{22} + w^e_{2s} + w^e_{2o} \\ \vdots \\ w^e_{n^*,1} + w^e_{n^*,2} + w^e_{n^*,s} + w^e_{n^*,o} \end{bmatrix} \times \frac{1}{4} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n^*} \end{bmatrix}$$

"A" "fat" "sat" "on"

n^* *word embedding* 차원에서의 A, fat, sat, on 벡터합의 평균

이를 기하적 관점으로 해석하면 $\boldsymbol{z} \in R^{n^*}$



z 는 n^* *word embedding* 차원으로 매핑된
Context word (A, fat, sat, on) 벡터합의 평균값임.

1. 디코딩 단계

$$(W^e)^T \boldsymbol{z} = W^d \boldsymbol{z} = \boldsymbol{y}$$

$(n \times n^*)$ $(n^* \times 1)$ $(n \times 1)$

$$= \begin{bmatrix} w^e_{11} & w^e_{21} & \cdots & w^e_{n^*,1} \\ w^e_{12} & w^e_{22} & \cdots & w^e_{n^*,2} \\ \vdots & \vdots & \ddots & \vdots \\ w^e_{n,1} & w^e_{n,2} & \cdots & w^e_{n^*,n} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n^*} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

n^* *word embedding* 차원에서의 A, fat, sat, on 벡터합의 평균 $\boldsymbol{z}$ 을 선형변환하여 vocab_size인 n 차원으로 매핑된 벡터

1. 손실함수 계산 단계

$$softmax(\boldsymbol{y}) = \boldsymbol{s}$$

0과 1 사이 확률값으로 변환하기 위함

$$softmax\left(\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}\right) = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix}, s_k = \frac{e^{y_k}}{\sum_{j=1}^{n} e^{y_j}}$$

예:
```
y : [-5   -3   0    3    5]
s : [0.   0.   0.01 0.12 0.88]
```

$$L(\boldsymbol{t}, \boldsymbol{s}) = -\sum_{i=1}^{n} t_i \log s_i$$

cat

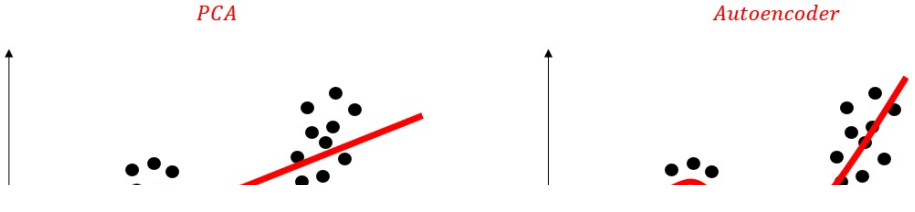
$$cross\ entropy\left(\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}\right)$$

s_3 은 1에 가까워져야하고,
나머지 s_i 들은 0에 가까워지라고 하는 의미
즉, cat을 나타내는 원핫벡터와 가까워지라는 의미

- 가정

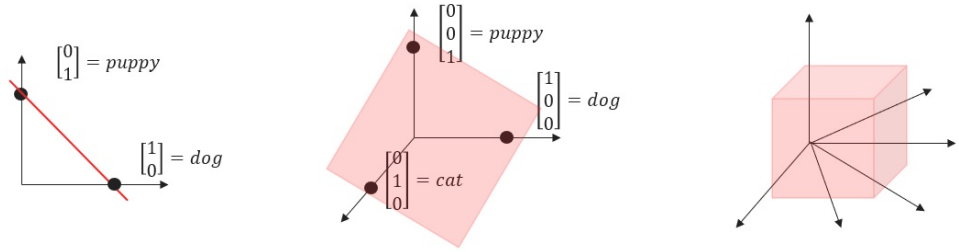
$\mathbf{s} = softmax(W^d W^e \mathbf{x})$ 에서 해석을 위해 softmax 함수를 제외하여 생각해보자, 그렇다면 선형변환만 있는 선형차원축소 방법임

예를 들어 선형차원축소 방법인 PCA와 비선형차원축소 방법인 Autoencoder에 대해 기하적 그림은 아래와 같음



tokenization 을 통해 center word와 context word들은 원핫 벡터 상태임

1. 만약 $L(\mathbf{t}, \mathbf{s}) = 0$ 혹은 $L(\mathbf{t}, \mathbf{s}) < \delta$ 이라고 해보자 $(\mathbf{s} = W^d W^e \mathbf{x})$

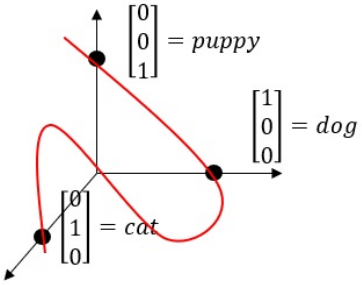


차원 : $2 \rightarrow 1$ 차원 : $3 \rightarrow 2$ 차원 : $n \rightarrow (n-1)$

- 2차원의 경우 선형차원축소를 통해 1차원으로 줄인 경우 $L(\mathbf{t}, \mathbf{s}) = 0$ 의 의미는 두 원핫벡터를 지나는 직선을 의미함
- 3차원의 경우 선형차원축소를 통해 2차원으로 줄인 경우 $L(\mathbf{t}, \mathbf{s}) = 0$ 의 의미는 세 원핫벡터를 지나는 평면을 의미함
- n차원의 경우 선형차원축소를 통해 (n-1)차원으로 줄인 경우 $L(\mathbf{t}, \mathbf{s}) = 0$ 의 의미는 n개 원핫벡터를 지나는 초평면을 의미함
- 이 의미는 선형변환만을 사용한 선형차원축소 방법의 경우에 $L(\mathbf{t}, \mathbf{s}) = 0$ 이 되는 상황은 (n-1) 차원, 즉 vocab_size 차원보다 한 차원 작은 경우에만 가능함

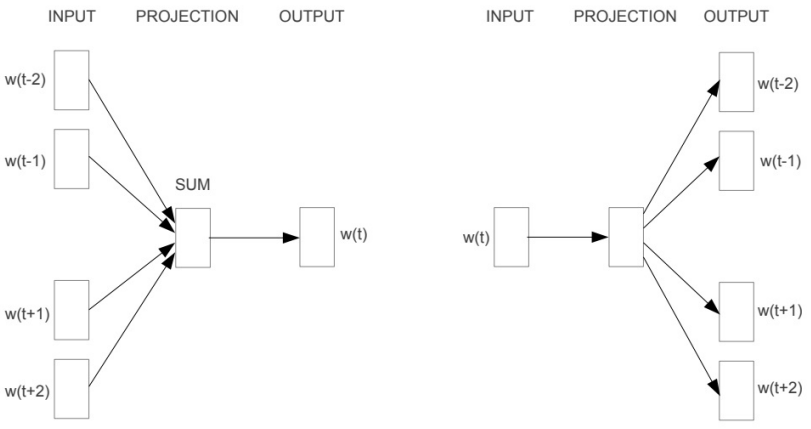
1. 만약 $L(\mathbf{t}, \mathbf{s}) > \delta$ 이라고 해보자

- $n >> n^*$ 에서는 많은 손실이 발생할 가능성이 높음
- 비선형차원축소 방법을 통해 $L(\mathbf{t}, \mathbf{s}) < \delta$ 으로 만들 가능성이 있음



- 비선형차원축소 방법 중 Autoencoder 의 Activation function, Parametric activation function, Combined parametric activation function 등을 사용해볼 수 있음

- Skip-gram



- Word2Vec 한계점

- OOV(Out Of Vocabulary) : 단어 사전에 없는 단어에 대해 표현할 방법이 없음
- 문장의 순서 반영 X
- 생성에 대한 구체적인 개념 반영 X

In [ ]: