6장 단어 임베딩

2022-11-02

Keyword

NLP Lab
Natural Language Processing

- 차원 축소
- Word2Vec
- GloVe

한림대학교
HALLYM UNIVERSITY

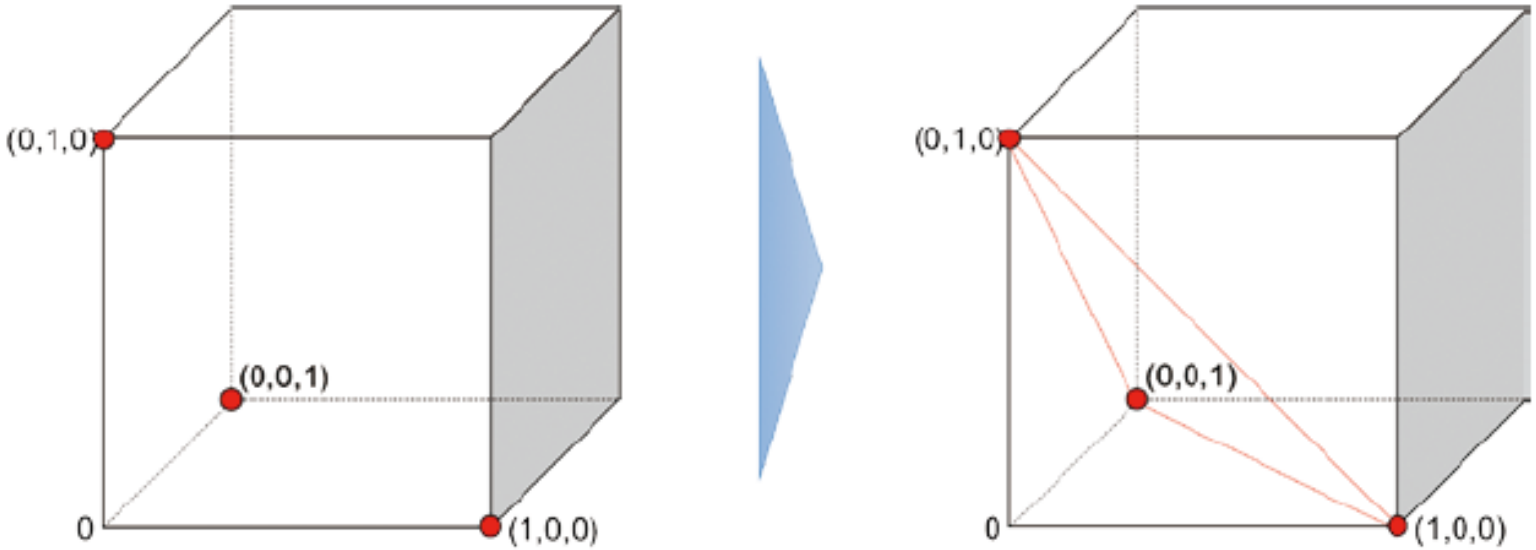
6.1 들어가며

NLP Lab
Natural Language Processing

- 단어의 의미와 모호성
  - 사람의 언어는 불연속적인 형태의 단어로 구성
  - 형태가 다른 단어들의 의미 연관성을 파악하기 어려움
- 단어나 문장, 문서를 벡터로 표현
  - 컴퓨터가 이해하기 쉬운 형태
  - 자연어의 형태를 벡터로 변환할 수 있는 함수나 매핑 테이블이 중요
- 단어의 벡터
  - 코퍼스로부터 단어의 특징(feature)을 추출하여 벡터로 구성
    - 정해진 크기의 윈도우 내에 함께 등장한 단어들의 출현 빈도
  - 희소 벡터의 문제
    - 차원의 저주
    - 가능한 낮은 차원으로 표현해야 함 -> 덴스(dense) 벡터로 표현
    - 단어 임베딩

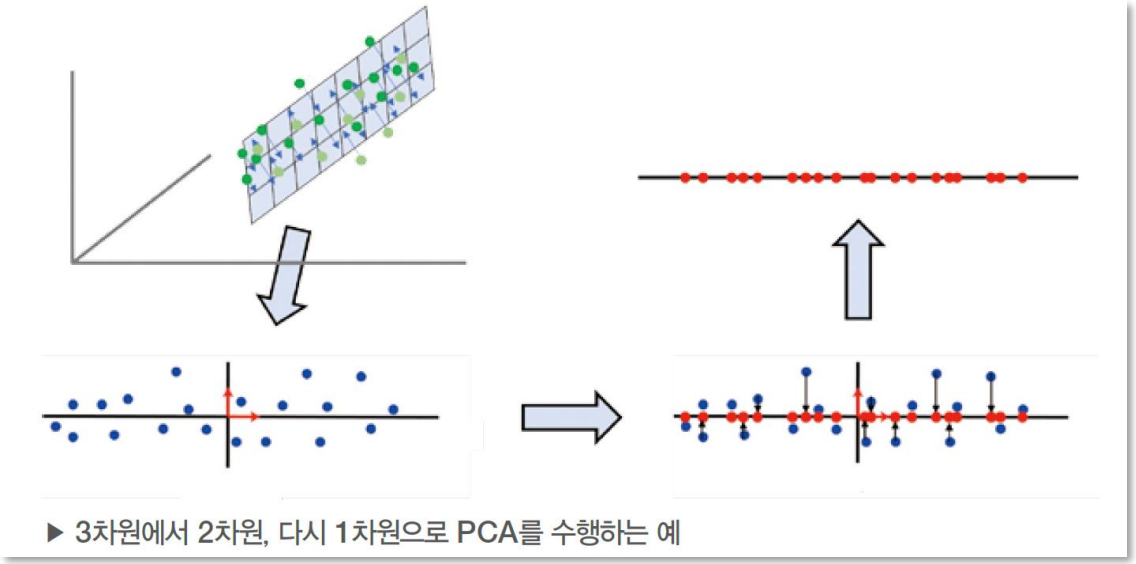
6.2 차원 축소

- 높은 차원에서 데이터 표현 -> 희소성 문제
- 같은 정보를 표현할 때 더 낮은 차원을 사용하는 것이 중요.



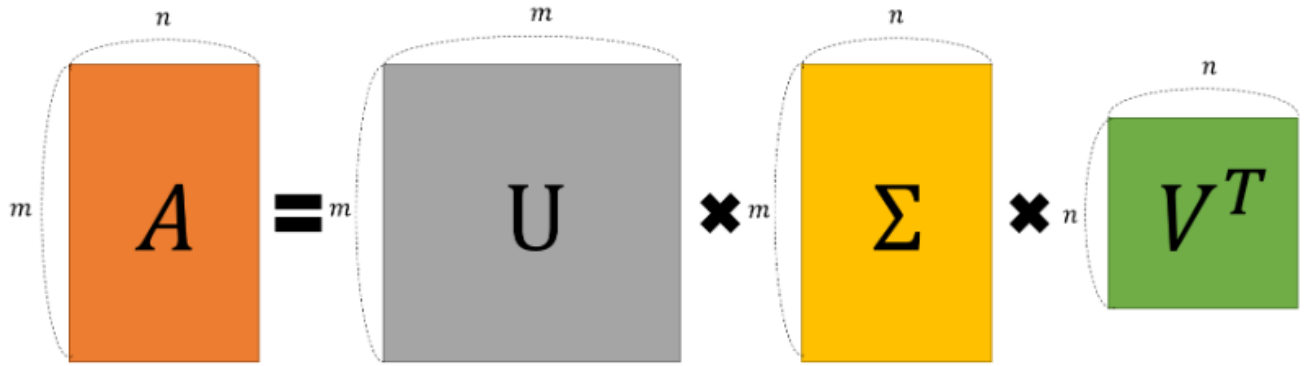
▶ 3개의 요소를 갖는 원핫 벡터의 경우에는 그림과 같이 차원 축소가 가능할 것입니다.

6.2.1 주성분 분석(PCA)

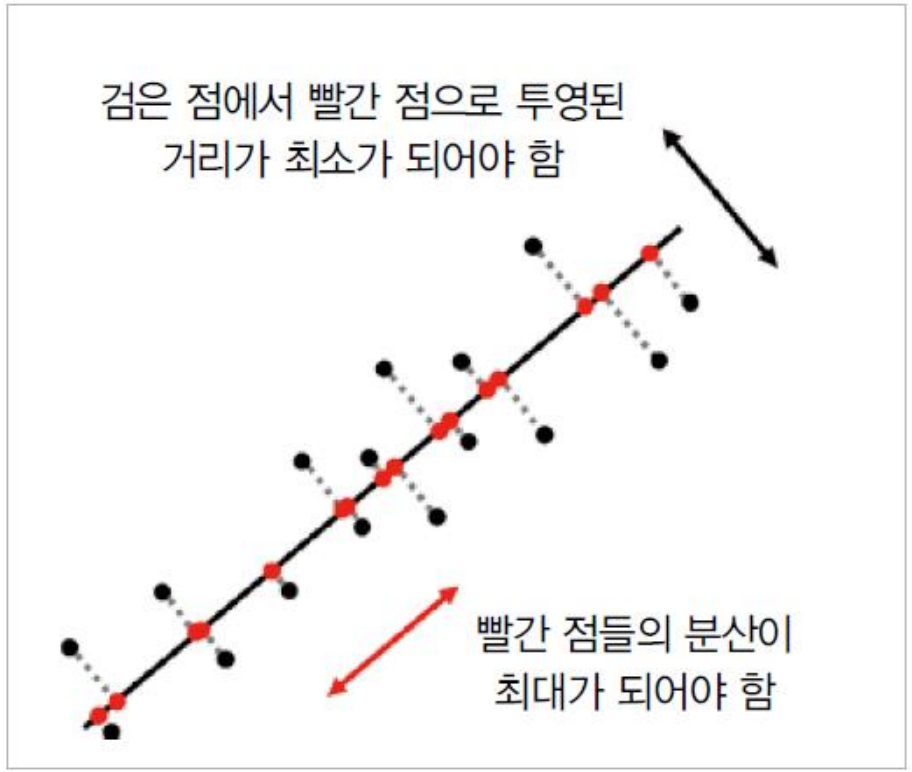


▶ 3차원에서 2차원, 다시 1차원으로 PCA를 수행하는 예

- Principal Component Analysis (PCA)
- 대표적인 차원 축소 방법
- 고차원 -> 낮은 차원
- 특잇값 분해 (singular value decomposition, SVD) 로 주성분 분석



6.2.1 주성분 분석(PCA)



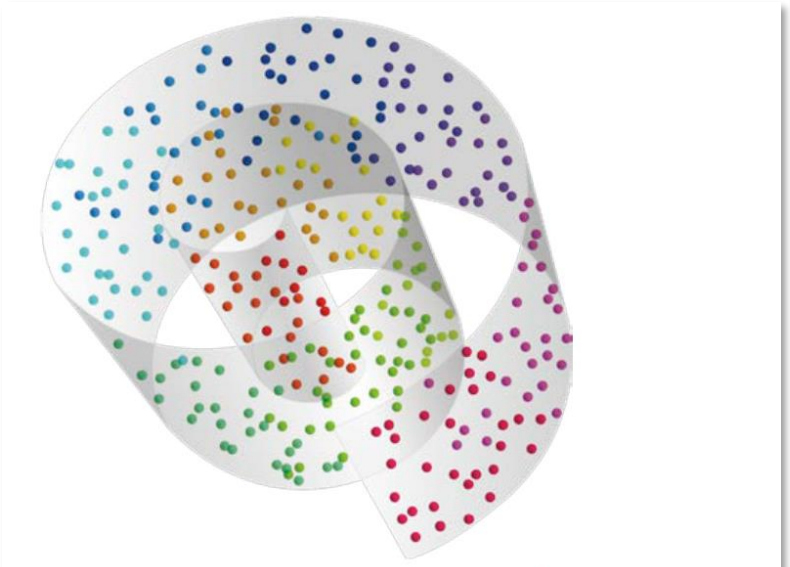
▶ 주성분의 조건

- 주성분의 조건
  - 1. 투사점들의 사이가 서로 최대한 멀어져야 함
  - 2. 투사된 거리가 최소가 되어야함.

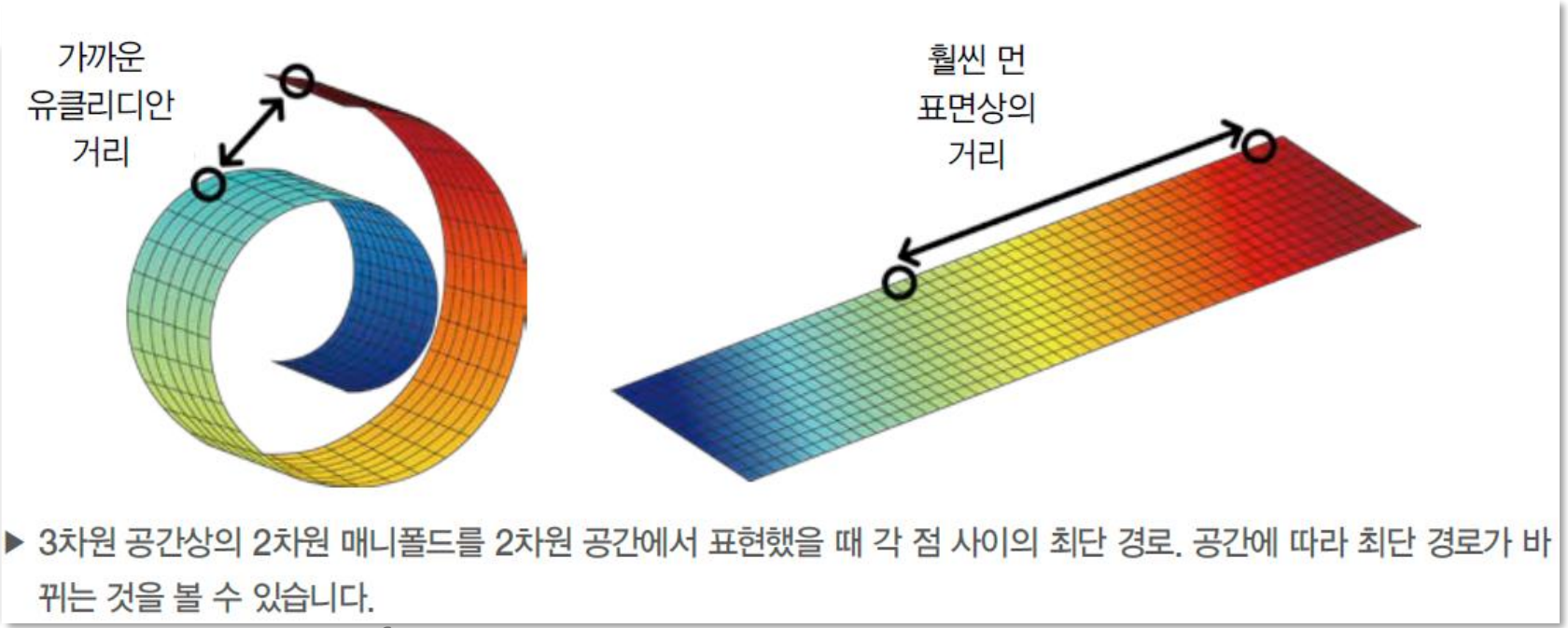
- 고차원 -> 저차원
  - 실제 데이터 위치와 고차원 평면에 투사된 점의 거리가 생길 수 밖에 없음. -> 정보의 손실
  - 주성분은 직선 혹은 평면이므로 불가피하게 정보의 손실이 있음.
  - 너무 많은 정보가 손실되면 효율적으로 데이터를 사용할 수 없음.
  - 따라서 높은 차원에 표현된 정보를 지나치게 낮은 차원에 축소하여 표현 하기가 어렵다.

6.2.2 매니폴드 가설

- 매니폴드 가설을 통해 차원 축소에 더 효율적으로 접근
- 매니폴드 가설이란 ?
  - 높은 차원에 존재하는 데이터들의 경우, 실제로는 해당 데이터들을 아우르는 낮은 차원의 다양체(manifold) 역시 존재한다.



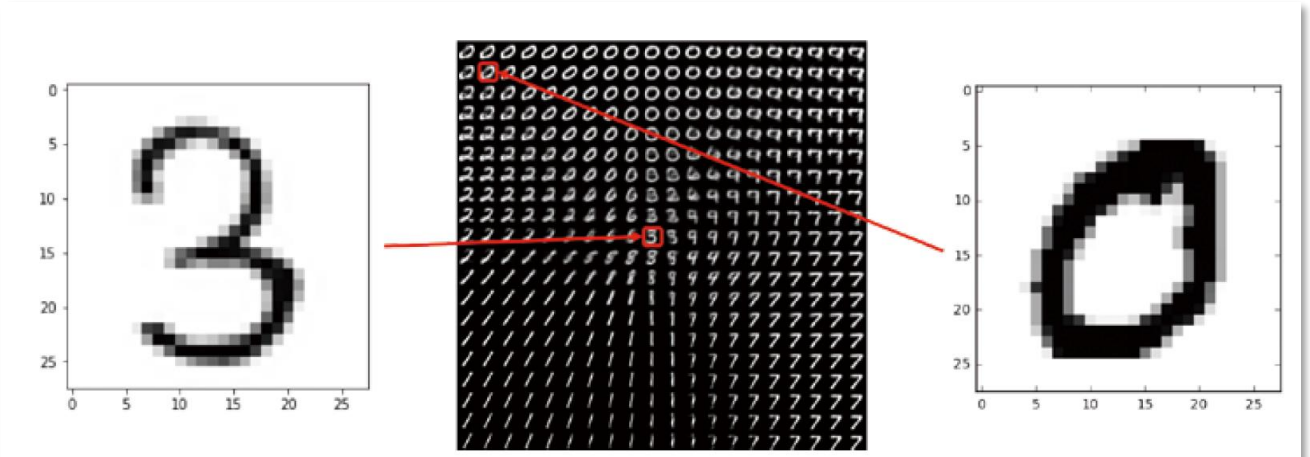
▶ 3차원 공간에 기묘한 모양으로 분포한 샘플들이 2차원 매니폴드에 속하는 모습



▶ 3차원 공간상의 2차원 매니폴드를 2차원 공간에서 표현했을 때 각 점 사이의 최단 경로. 공간에 따라 최단 경로가 바뀌는 것을 볼 수 있습니다.

6.2.2 매니폴드 가설

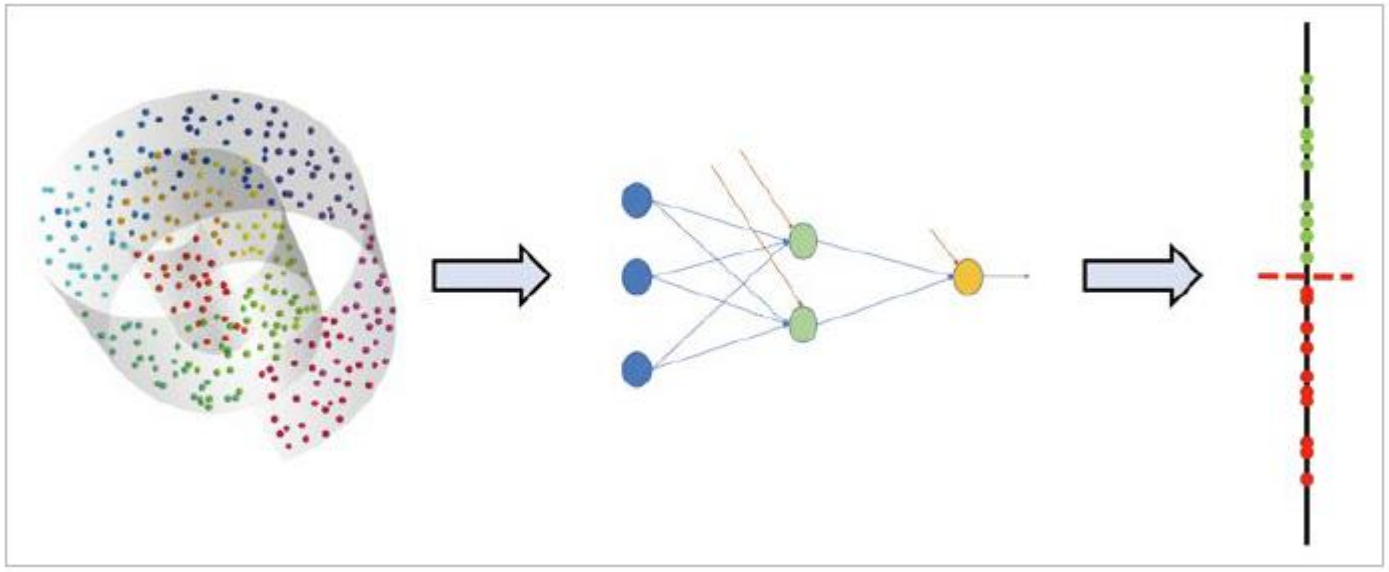
- MNIST 데이터를 2차원의 숨겨진 저차원에 표현한다고 가정
- 빨간색 표시 - 2차원 공간에서 사람이 인지하는 특징과 비슷한 특징을 갖는 위치와 관계
- 원래의 데이터차원인 784차원의 고차원 공간에서는 전혀 다른 거리와 관계를 지닐 것



▶ MNIST 데이터를 2차원 공간에 표현했을 때 두 샘플의 위치

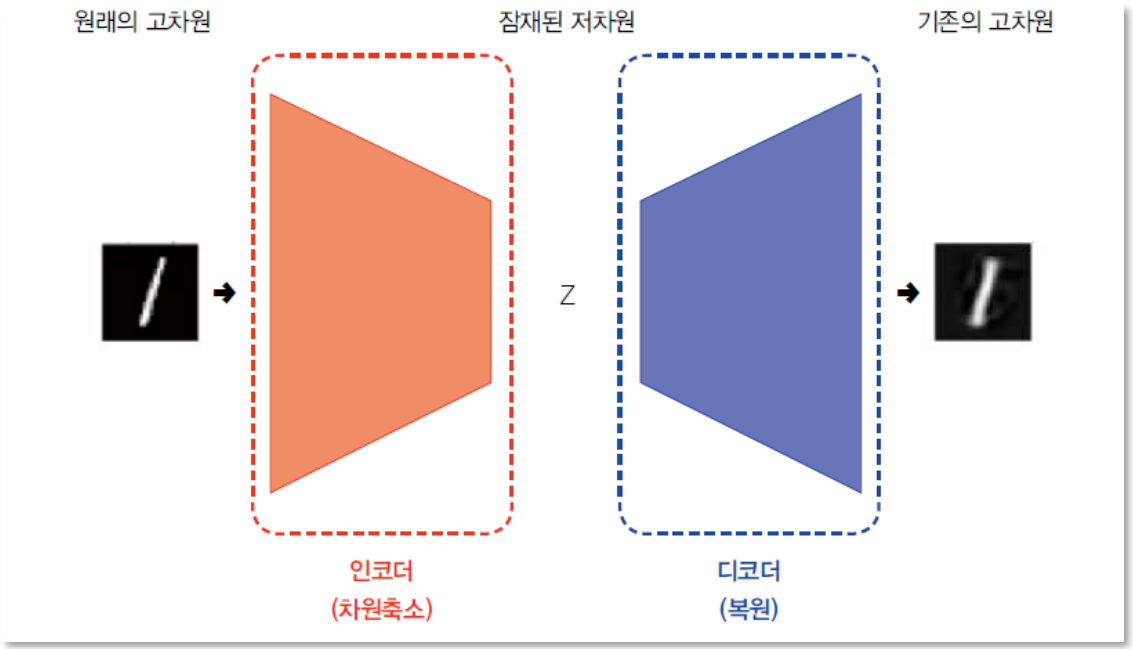
6.2.3 딥러닝이 잘 동작한 이유

- 대부분의 경우 딥러닝이 문제를 풀기 위해 차원 축소를 수행하는 과정은 고차원상에서 매니폴드를 찾는 과정
- 주성분 분석(PCA)과 같이 선형적인 방식에 비해 딥러닝은 비선형적인 방식으로 차원 축소를 수행하며, 가장 잘 해결하기 위한 매니폴드를 잘 찾는다.



▶ 3차원 데이터를 입력으로 받아 1차원의 이진 분류를 수행할 때

6.2.4 오토인코더 (Auto Encoder)



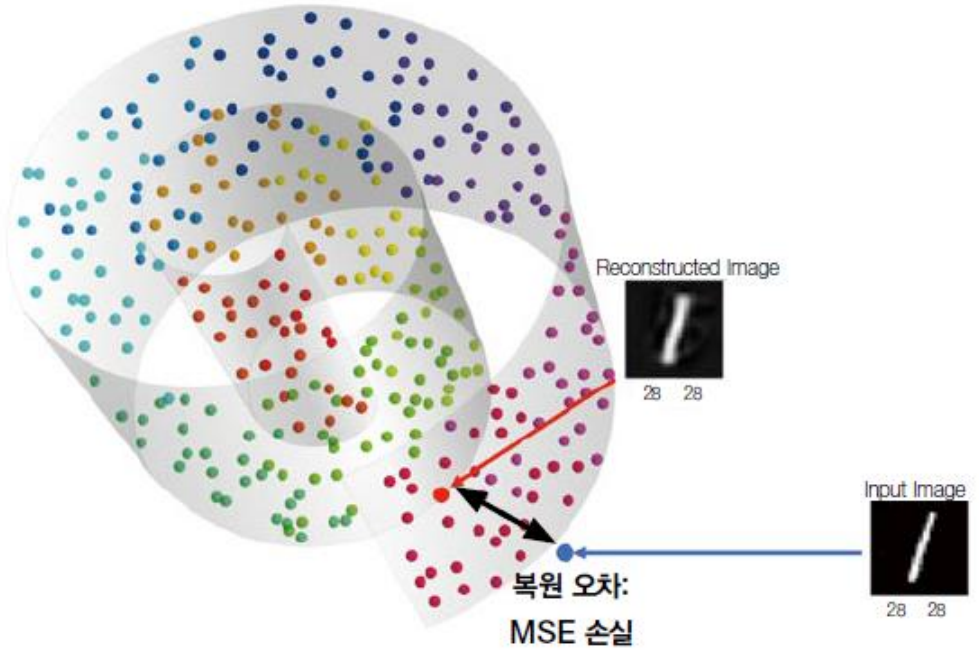
- 인코더
- 고차원의 샘플 벡터를 입력으로 받아 매니폴드를 찾고, 저차원으로 축소하는 인코더를 거쳐 Hidden 벡터로 표현.

- 디코더
- Hidden 벡터를 받아 다시 원래 입력 샘플에 존재하던 고차원으로 데이터를 복원하는 작업.

- 복원 된 데이터는 고차원 상의 매니폴드 위에 위치하게 됨.

6.2.4 오토인코더 (Auto Encoder)

- 손실 함수
  - 복원된 데이터와 실제 입력 데이터 사이의 차이를 최소화
  - 복원에 필요한 정보만 남김
  - 정보 손실은 불가피
- 단어 임베딩
  - 희소 단어 특징 벡터로 훈련
  - 병목 계층 결괏값을 활용 가능



▶ 고차원(3차원)에서 저차원(2차원)으로 투사할 때의 정보 손실(복원 오류)

6.3 흔한 오해

- Skip-gram 또는 GloVe를 사용해 저차원의 덴스벡터로 표현하는 방법
- 임베딩 벡터를 사전 훈련된 임베딩 벡터라고 부름.
- Word2Vec을 통해 얻은 단어 임베딩 벡터가 훌륭하게 단어의 특징을 잘 반영하고 있지만 모든 TASK에서 문제를 해결할 수 있는 최적의 임베딩 벡터는 아님.
- TASK의 목적에 따른 임베딩 벡터를 형성해야 TASK에 맞는 최적의 임베딩 벡터를 형성할 수 있음.

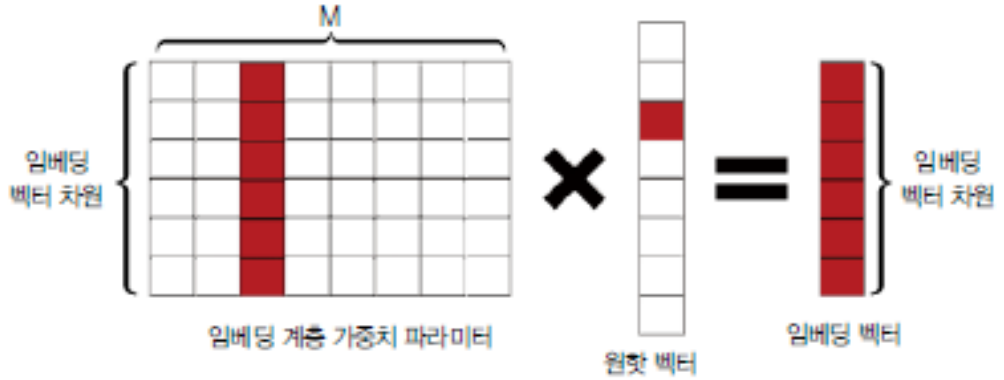
6.3.1 Word2vec 없이 신경망 훈련하기

- 임베딩 계층 (embedding layer) 제공

$$y = \text{emb}(x) = Wx,$$

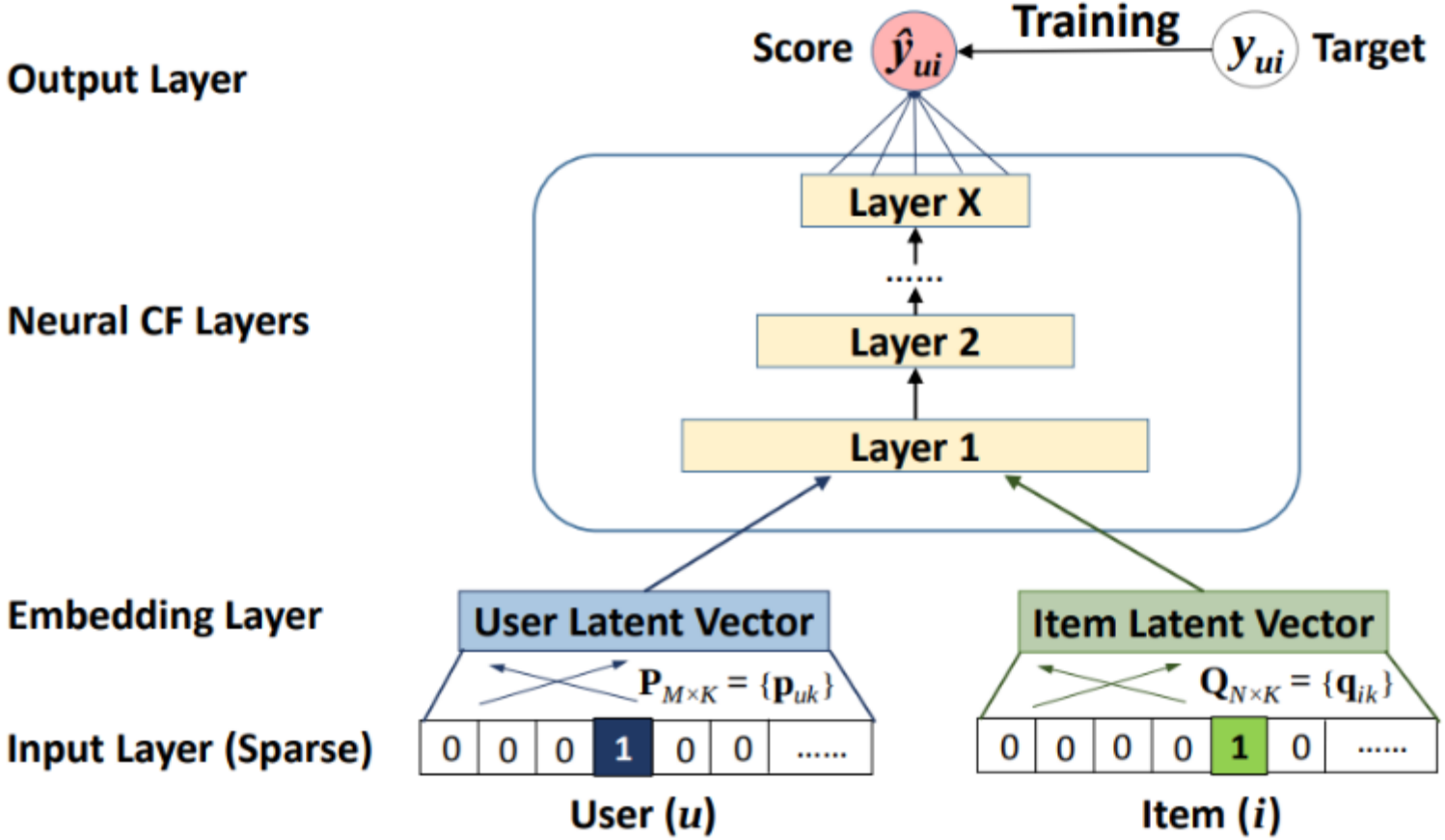
where $W \in \mathbb{R}^{d \times |V|}$ and $|V|$ is size of vocabulary.

  - W: d * |V| 크기의 2차원 행렬



▶ 임베딩 계층의 동작 개념 : 0과 곱해지는 부분은 무시될 것입니다.

실제로는 단어의 인덱스 정숫값만 필요


Output Layer
Score
$\hat{y}_{ui}$
Training
y_{ui}
Target
Neural CF Layers
Layer X
.....
Layer 2
Layer 1
Embedding Layer
User Latent Vector
$\mathbf{P}_{M\times K} = \{\mathbf{p}_{uk}\}$
Item Latent Vector
$\mathbf{Q}_{N\times K} = \{\mathbf{q}_{ik}\}$
Input Layer (Sparse)
0 0 0 1 0 0 .....
0 0 0 0 1 0 .....
User (u)
Item (i)

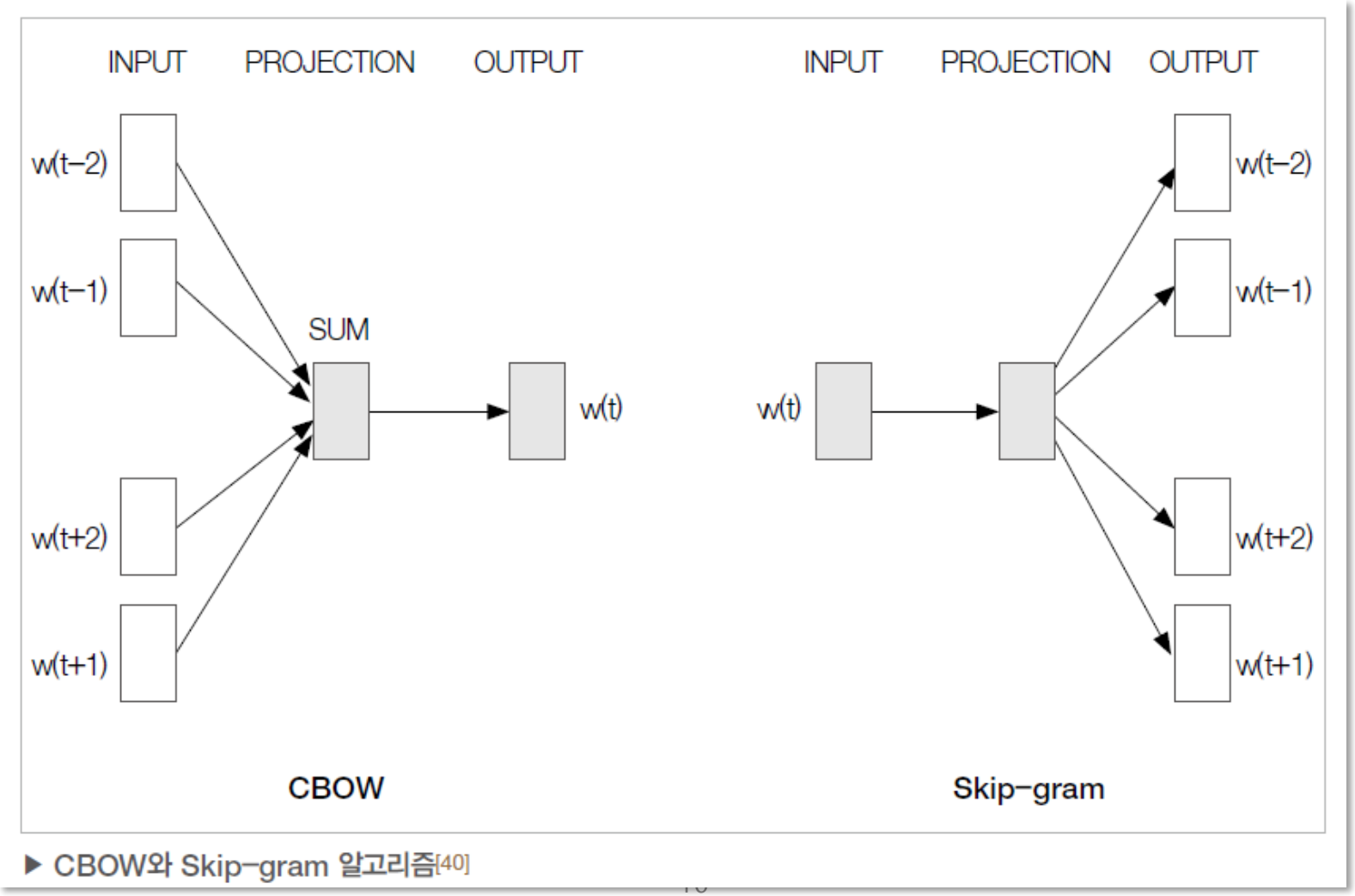
6.3.2 그래도 word2vec 적용하는 경우

- 사전 훈련된 단어 임베딩 벡터를 적용하는 경우
  - 준비된 코퍼스의 양이 너무 적은 경우
  - 외부로부터 많은 양의 말뭉치를 통해 미리 훈련된 단어 임베딩을 구할 수 있는 경우
  - 베이스라인을 만들고 성능을 끌어올리는 경우 (여러 임베딩을 사용후 비교)
  - 전이학습 방식을 적용하는 경우
  - 언어 모델을 통해 사전 훈련

6.4 Word2Vec

- 2013년 Tomáš Mikolov 가 word2vec이라는 방법을 제시
- 복잡한 신경망 네트워크를 사용해 임베딩 벡터를 형성하는 데 의문을 가짐.
- 빠르고 쉬우면서 효율적으로 임베딩하는 word2vec 을 제시
- 두 가지 방법 : CBOW , Skip-gram
  - 두 방법 모두 함께 등장하는 단어가 비슷할수록 비슷한 벡터 값을 가질 것이라는 공통된 가정을 전제.
- 두 방법 모두 윈도우 크기가 주어지면 특정 단어를 기준으로 윈도우 내의 주변단어를 사용하여 단어 임베딩을 학습.

6.4 Word2Vec

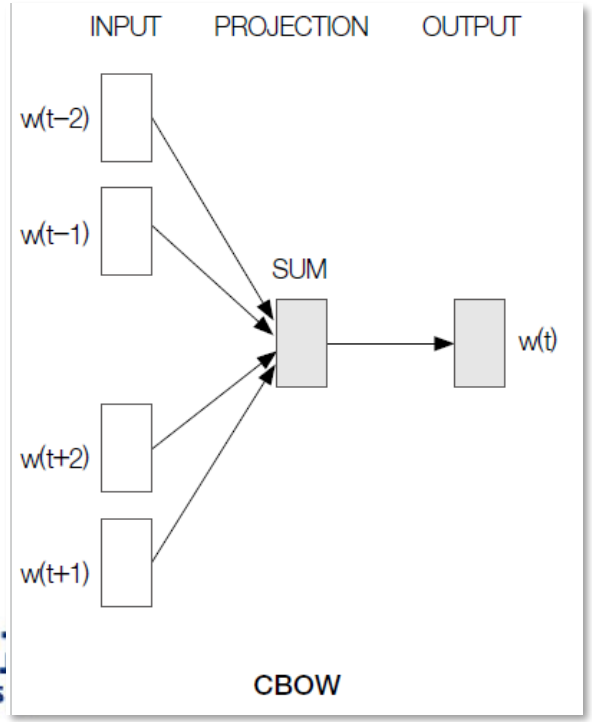


▶ CBOW와 Skip−gram 알고리즘[40]

6.4.1 알고리즘 개요 : CBOW 방식과 skip-gram 방식

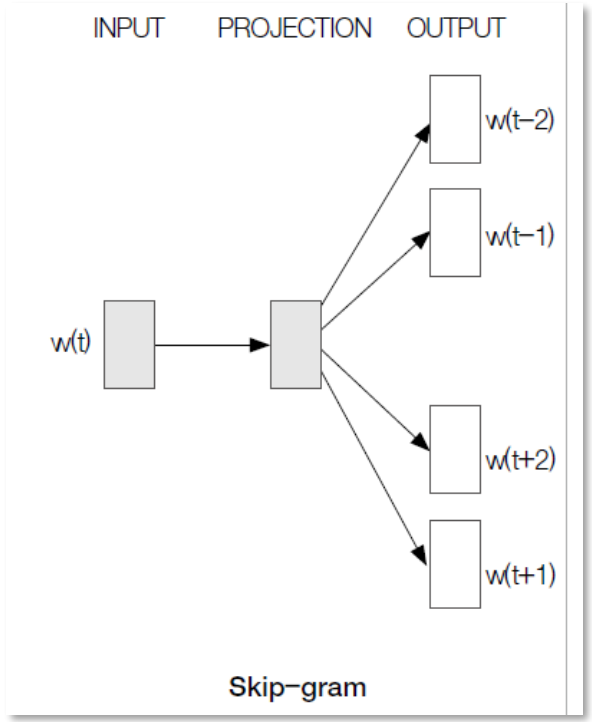
CBOW

- 주변에 나타나는 단어들을 원핫 인코딩된 벡터로 입력 받아 대상 단어를 예측



Skip-gram

- 대상 단어를 원핫 인코딩된 벡터로 입력받아 주변에 나타나는 단어를 예측.



6.4.2 상세 훈련 방식

- MLE를 통해 argmax 내의 수식을 최대로 하는 θ를 찾는다.

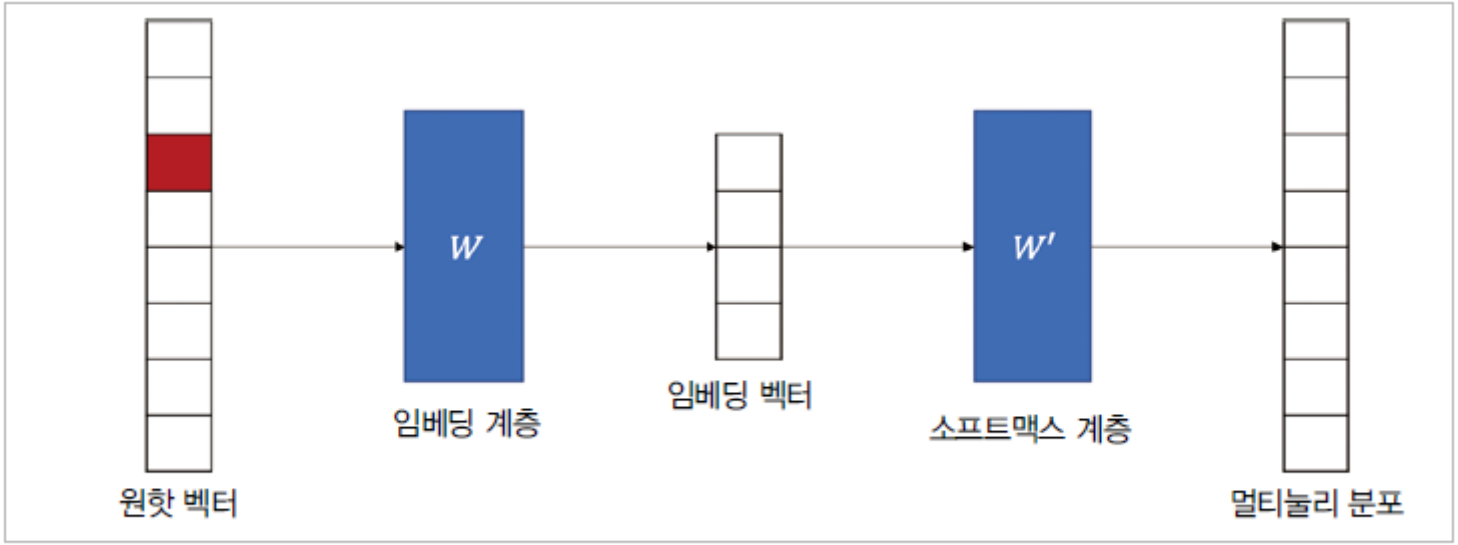
- W_t가 주어졌을 때, 앞 뒤 n 개의 단어$(w_{t-\frac{n}{2}}, \ldots, w_{t+\frac{n}{2}})$를 예측하도록 훈련. (window size n)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{t=1}^{T} \left(\sum_{i=1}^{n} \log P(w_{t-i} \mid w_t; \theta) + \sum_{i=1}^{n} \log P(w_{t+i} \mid w_t; \theta) \right)$$

6.4.2 상세 훈련 방식

$$\hat{y} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \ \operatorname{softmax}(W'Wx)$$

where $W' \in \mathbb{R}^{|V| \times d}, W \in \mathbb{R}^{d \times |V|}$ and $x \in \{0,1\}^{|V|}$

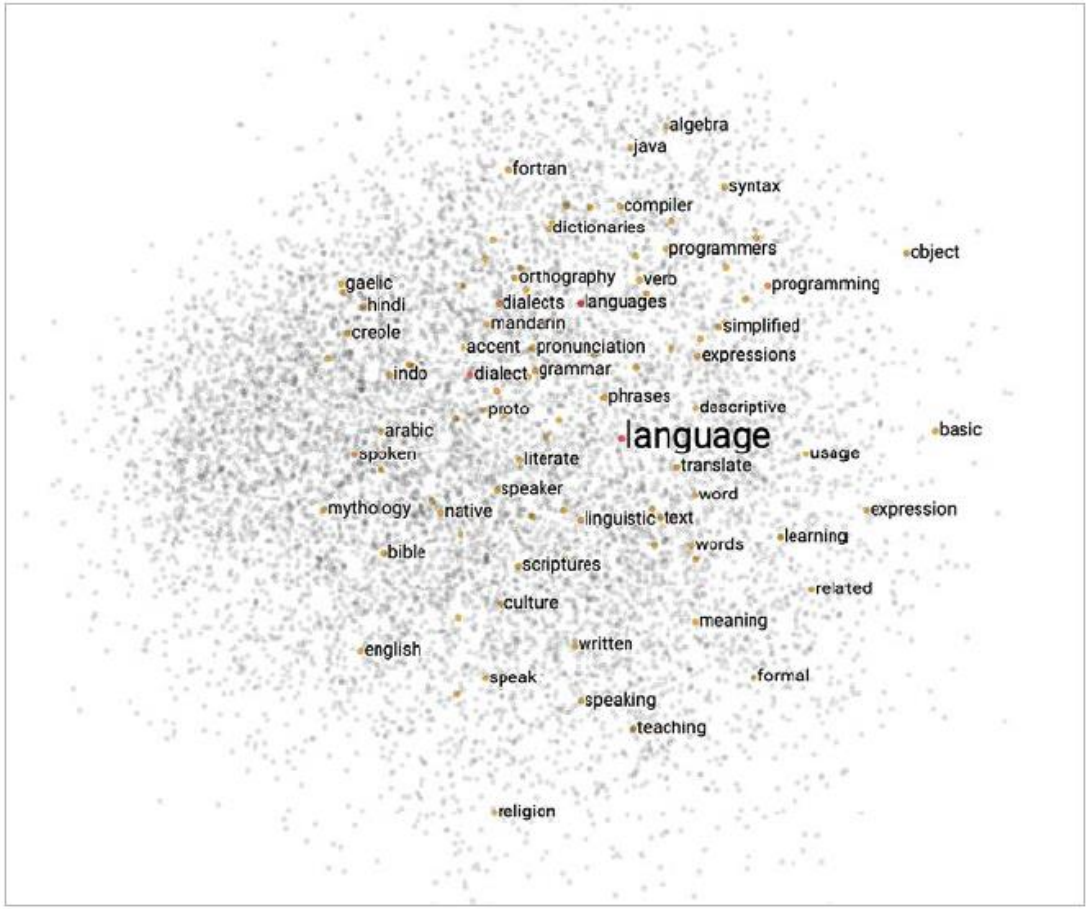


▶ Skip-gram의 구조

W의 각 행 -> 단어 임베딩 벡터

6.4.2 상세 훈련 방식

- 단어 임베딩 벡터들을 3차원 공간에 표현



▶ Skip-gram을 통해 얻은 단어 임베딩 벡터를 PCA를 통해 시각화한 예제

6.5 Glove

- Global Vectors for word representation
- Skip-gram은 대상 단어를 통해 주변 단어를 예측하도록 네트워크를 구성하여 단어 임베딩 벡터를 학습.
- Glove는 대상 단어에 대해서 코퍼스에 함께 나타난 단어별 출현 빈도를 예측.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{x \in \mathcal{X}} f(x) \times | W'Wx - \log C_x |_2$$

where C_x is vector of cooccurences with x

Also, $x \in \{0,1\}^{|V|}, W \in \mathbb{R}^{d \times |V|}$ and $W' \in \mathbb{R}^{|V| \times d}$

- Skip-gram을 위한 네트워크와 유사.
- 분류 문제가 아닌 출현 빈도를 근사하는 회귀 문제이기 때문에 MSE를 사용하여 최적화

Glove

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{x \in \mathcal{X}} f(x) \times | W'Wx - \log C_x |_2$$

where C_x is vector of cooccurences with x

Also, $x \in \{0,1\}^{|V|}, W \in \mathbb{R}^{d \times |V|}$ and $W' \in \mathbb{R}^{|V| \times d}$

$$f(x) = \begin{cases} (\text{Count}(x) / \text{thres})^{\alpha} & \text{Count}(x) < \text{thres} \\ 1 & \text{otherwise} \end{cases}$$

- 단어 x 자체의 출현빈도 혹은 사전확률에 따라 MSE 손실 함수 값이 매우 달라질 것임.
- 따라서 $f(x)$는 단어의 빈도에 따라 손실 함수에 가중치를 부여한다.
- GloVe 논문에서는 thres 100, α 3/4 일 때 가장 좋은 결과가 나왔다고 함.

6.5.2 장점

- Skip-gram과 달리 GloVe는 처음 코퍼스를 통해 단어별 동시 출현 빈도를 조사하여 그에 대한 출현 빈도 행렬을 만들고, 이후엔 해당 행렬을 통해 동시 출현 빈도를 근사하려 함

- 따라서 코퍼스 전체를 훑으며 대상 단어와 주변 단어를 가져와 학습하는 skip-gram과 달리 훨씬 학습이 빠름.

- 코퍼스를 훑으며 학습하는skip-gram 특성상 출현 빈도가 적은 단어에 대해서 학습기회가 적음. 따라서 출현 빈도가 적은 단어들은 비교적 부정확한 단어 임베딩 벡터를 학습한다.

6.5.3 정리

- GloVe vs. skip-gram
  - GloVe 논문에서는 GloVe가 가장 뛰어난 단어 임베딩 방식이라 주장
  - Skip-gram도 hyper-parameter 를 조정하면 거의 비슷함
  - 시스템 구성에 대한 제약에 따라 적절한 방식을 선택

마무리

- 기존의 선형적인 차원 축소 방법에 비해 신경망은 비선형적인 차원 축소를 통해 특징을 효율적으로 추출. → 딥러닝이 기존의 머신러닝 알고리즘에 비해 월등한 성과를 내는 이유
- 이런 딥러닝의 비선형적인 차원 축소는 계산 비용이 비싸고 최적화가 어려운 단점.
- Word2Vec, GloVe은 비선형적인 방법을 사용하지 않고도 매우 좋은 단어 임베딩을 구현.
- 하지만 뒷 장에서 소개하는 텍스트 분류, 텍스트 생성 같은 기법에서는 해당 장에서 설명한 단어 임베딩 알고리즘을 쓰는 것 보다 단순 임베딩 계층을 사용하는 것이 더 정확하고 효율적인 방법임.
- 이후 장은 문장 단위로 자연어 처리를 하는 방법들을 배웁니다.