ML Session

Word embedding

word embedding??

우리가 사용하는 모델은 수치형 데이터만 처리할 수 있다.

텍스트 데이터를 모델에 넣으려면 어떤 방법을 써야 할까?

텍스트 데이터를 수치형으로 변환 하는 것을 word embedding이라고 한다.

one-hot encoding

단어 하나를 벡터 형태로 표현하는 가장 기본적인 방법

벡터의 길이는 서로 다른 단어의 개수가 된다.

문장
나는 어제 밥을 먹었다.

단어 벡터	
나는	[1, 0, 0, 0]
어제	[0, 1, 0, 0]
밥을	[0, 0, 1, 0]
먹다	[0, 0, 0, 1]

문장 벡터	
나는 어제 밥을 먹었다.	[1, 0, 0, 0]
	[0, 1, 0, 0]
	[0, 0, 1, 0]
	[0, 0, 0, 1]

Bag of words

단어의 개수 만큼의 길이를 가지는 벡터에 그 단어가 나온 숫자만큼 횟수만큼
인덱스의 값을 증가시키는 방법

문장
어제도 밥 오늘도 밥
나는 어제 밥을 먹었다

인덱스
[어제, 밥, 오늘, 먹다, 나는]

문장 벡터	
어제도 밥 오늘도 밥	[1, 2, 1, 0, 0]
나는 어제 밥을 먹었다	[1, 1, 0, 1, 1]

Bag of words

위의 방법(hard coding)만 사용하면 단어 간의 의미를 찾기 힘들다.

데이터가 너무 spares해지는 것도 문제가 된다.

word embedding으로 단어 간의 의미나 유사도를 찾아야한다.

word embedding을 하면 벡터 공간에 텍스트를 뿌리게 된다.

단어들 사이의 연관성을 기하학적으로 나타낼 수 있다.

word2vec

다양한 기법들이 있지만 우리가 사용할 방법은 word2vec이다.

word2vec의 기본 개념은 '같이 많이 등장하는 단어는 비슷한 의미를 가진다.' 이다.

word2vec은 크게 두가지(CBOW와 Skip-gram)로 나뉜다.

신경망에 대한 이해가 조금 필요하지만 이번 강의에서 다루지 않을 예정이다.

CBOW

나는 ___을(를) 하다가 인대가 파열되었다.

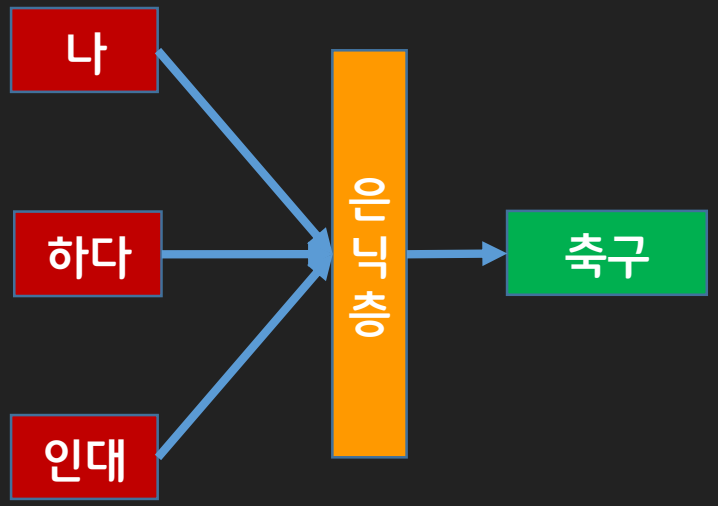
빈칸에 무슨 단어가 들어갈까??

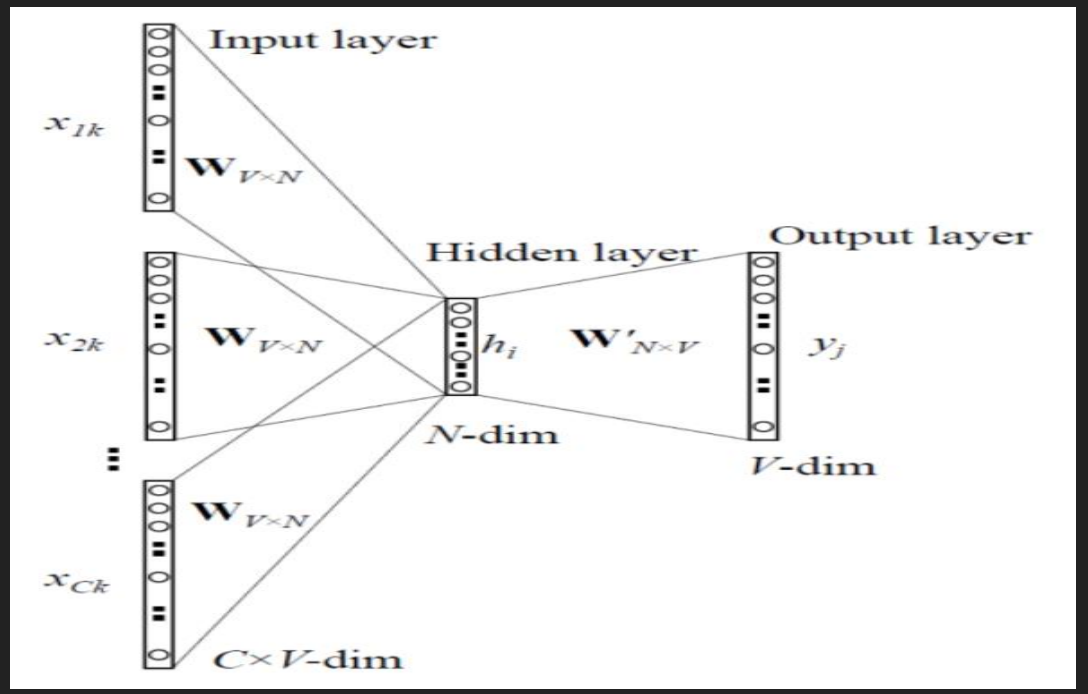
CBOW

위의 사례처럼 주변 단어를 통해 주어진 단어를 유추하는 방법이 바로 CBOW다.

사용할 단어를 one-hot encoding 형태로 변환해서 신경망에 집어 넣는다.

학습이 다 끝나면 은닉층은 행렬 형태로 나오게 되는데 각행이 그 단어의 embedding 값이 된다.
ex) 첫번째 단어의 embedding은 은닉층 W의 첫번째 행이다.





CBOW

__ 축구를 하다가 인대가 파열되었다.

나는 ___ 하다가 인대가 파열되었다.

나는 축구를 ___ 인대가 파열되었다.

나는 축구를 하다가 ___ 파열되었다.

나는 축구를 하다가 인대가 _____.

window size = 2

window size라는 개념이 있는데
중심 단어 앞뒤로 몇 개의 단어를
볼지 결정하는 하이퍼파라미터다.

Skip-gram

__ 축구을(를) ___ ___ _____.

이번에는 무슨 단어가 들어갈까?

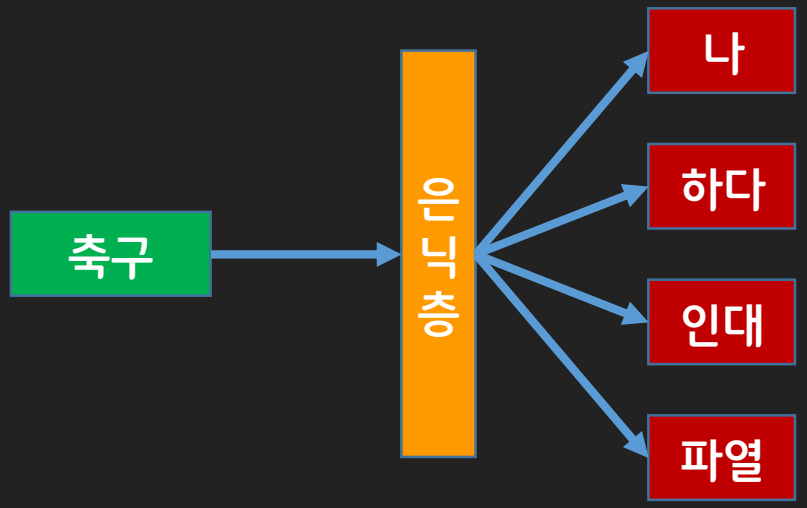
Skip-gram

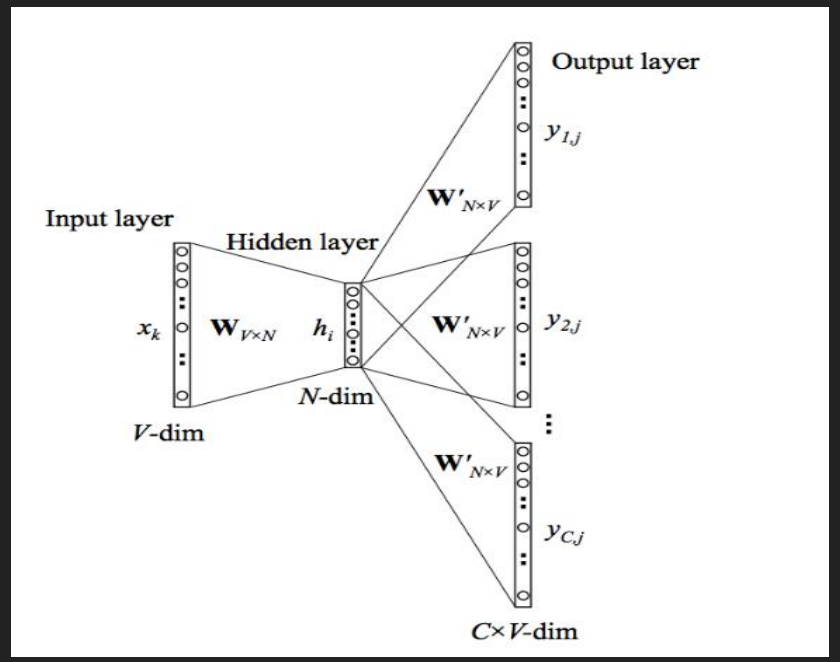
Skip-gram은 CBOW와는 반대로 중심단어를 통해 주변 단어를 예측하는 방법이다.

CBOW에 비해 중심단어의 업데이트 기회가 많다.

softmax(다중 클래스 분류 때 쓰는 함수) 함수를 사용해서 단어가 나올 확률을 계산

CBOW와 동일하게 은닉층 행렬에서 각 행이 각 단어의 embedding 벡터가 된다.





Skip-gram

일반적으로 CBOW 보다 Skip gram이 성능이 좋은 편이기 때문에
Skip gram을 많이 쓴다.

Skip-gram

Skip gram의 특징

1. subsampling frequent words: 많이 나오는 단어를 확률적으로 제외하는 방법

영어의 a, the 같은 단어들이나 한국어의 은/는 같은 글자들은 많이 등장하지만 특별한 의미가 없다.

따라서 많이 나오면 나올수록 자주 제외하고 적게 나오는 단어는 빼지 않고 학습하는 방법

단어를 제외하기 때문에 학습량이 획기적으로 감소한다.

Skip-gram

Skip gram의 특징

2. negative sampling: 학습할 때 들어가지 않았던 단어를 학습에 사용하는 방법

기존 -> 단어 A를 넣었을 때 나오는 단어를 학습
negative sampling -> 단어 A를 넣었을 때 나와야 하는 단어(positive)
나와야 하지 말아야 하는 단어(negative)를 학습

모든 단어에 대해 확률 값을 구하지 않고 positive와 sampling한 negative의 확률 값만 계산하기 때문에 계산량이 줄어든다.

Skip-gram

Skip gram의 특징

3. Hierarchical Softmax: softmax 함수의 계산량을 줄이는 방법

이번 시간 안에 이해하기 어려워서 자세한 설명은 생략

거리 계산 방법

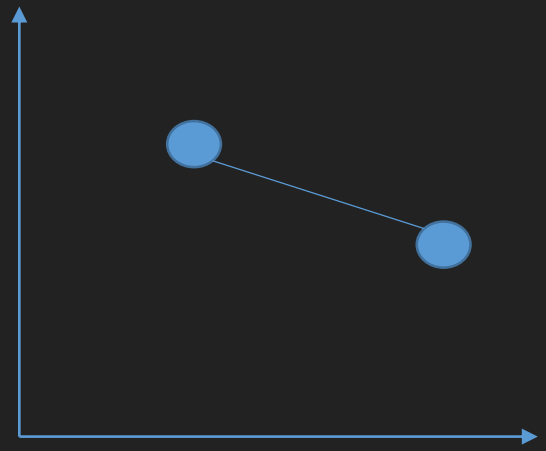
열심히 word embedding을 통해 word vector를 만들었다면
벡터와 벡터 사이의 거리를 구해보자

거리 계산 방법

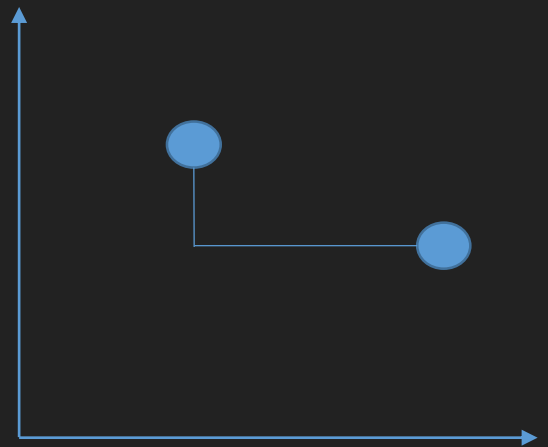
벡터 간의 거리(유사도)를 구하는 알고리즘

1. 유클리디안 거리: (흔히 아는) L2 distance
2. 맨해튼 거리: L1 distance
3. 코사인 유사도: 벡터의 크기는 고려하지 않고 방향성을 보는 방법 두 벡터가 이루는 각이 작을 수록 유사도가 높다.
4. 자카드 계수: 집합 사이의 유사도를 판단하는 방법
5. 피어슨 상관계수

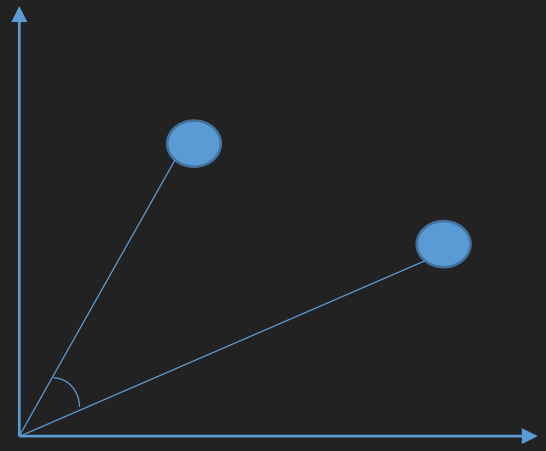
거리 계산 방법



유클리디안 거리



맨해튼거리



코사인 유사도

Question