CS20SI

Lec4. Structure your TensorFlow model

2017.07.17

장예훈

들어가기 전에...

- 지난 lecture에서 화재와 도난사고의 상관 관계를 예측한 Linear Regression Model과

  MNIST를 이용한 Logistic Regression Model을 살펴봄

- 우리는 위의 Model을 작성하였던 방법과 같은 방법으로도 조금 더 **복잡한 Model**을 작성 할 수 있음

  → **자세한 Planning**이 필요

  그렇지 않으면, Code가 messy해지고, debug하기도 어려움

- 이번 Lecture에서는 Model을 보다 **효율적인 구성**으로 작성하는 방법을 소개

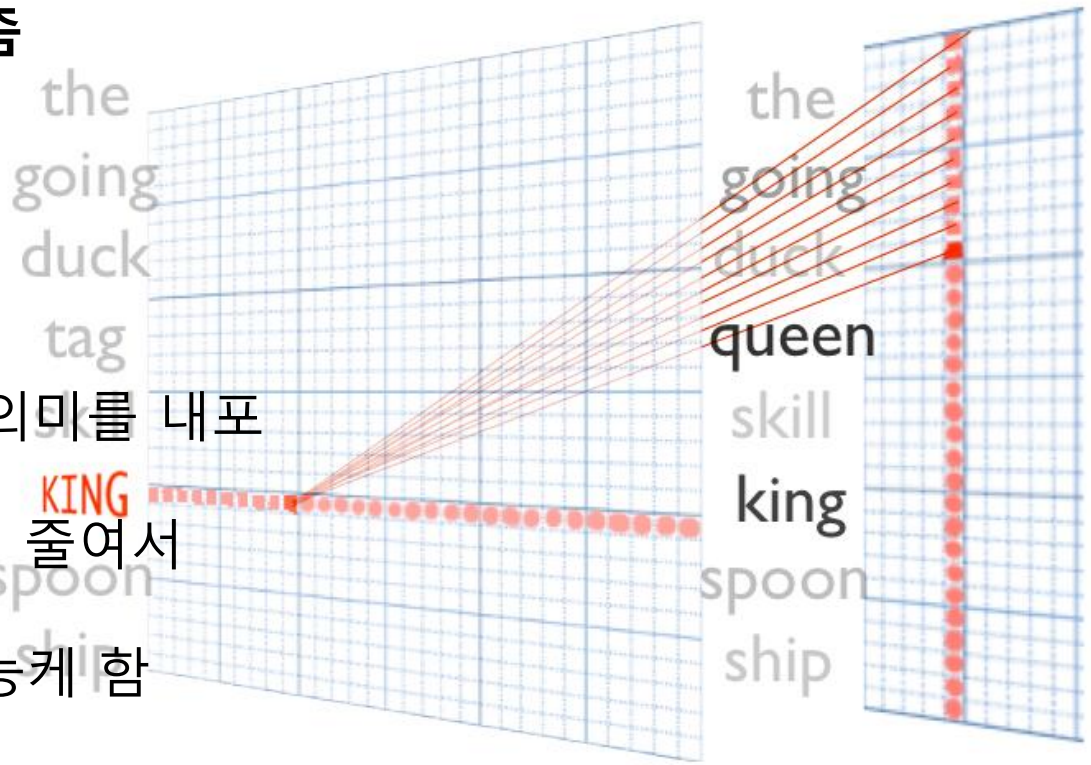
- 예시 모델은 word2vec을 사용

Word2vec ?

- word2vec : 구글의 연구원인 Tomas Mikolov와 Kai Chen, Greg Corrado, Jeffrey Dean의 논문인 "Efficient Estimation of Word Representations in Vector Space **(벡터 공간상에서 단어 의미의 효율적인 추정)**"에서 제안된 방법을 구현한 **알고리즘**
- 단어의 의미를 벡터형태로 표현하는 기법

  ▷ 각 단어를 200차원 정도의 공간에 표현
- 두 단어 간의 거리는 관계성을, 방향은 문맥상의 의미를 내포
- 기존의 모델(NNLM, RNNLM)보다 계산량을 엄청나게 줄여서

  기존의 방법에 비해 **몇 배 이상 빠른 학습**을 가능케 함
- 가장 많이 사용하는 Word Embedding Model



Word2vec의 예시

- thisplusthat.me : 데이터 과학자 Christopher Moody가

  만든 word2vec을 이용한 검색엔진 형태의 질의응답 사이트

- France - Paris + Seoul = Korea
- King – Men + Women = Queen
- Basketball – Michael Jordan + Golf = Tiger Woods

cf) 한국어 버전 : http://w.elnn.kr/search/

Word2vec의 동작원리

- **Concept** "같은 맥락을 지닌 단어는 **가까운 의미**를 지니고 있다."
- 텍스트 문서를 통해 학습을 진행
- 한 단어에 대해 근처(전후 5~10 단어 정도)에 출현하는 다른 단어들을 관련 단어로서 뉴럴넷에 학습
- **연관된 의미**의 단어들은 문서상에서 **가까운 곳**에 출현할 가능성이 높음

→ 학습을 반복해 나가면 두 단어는 점차 **가까운 벡터**를 지니게 됨

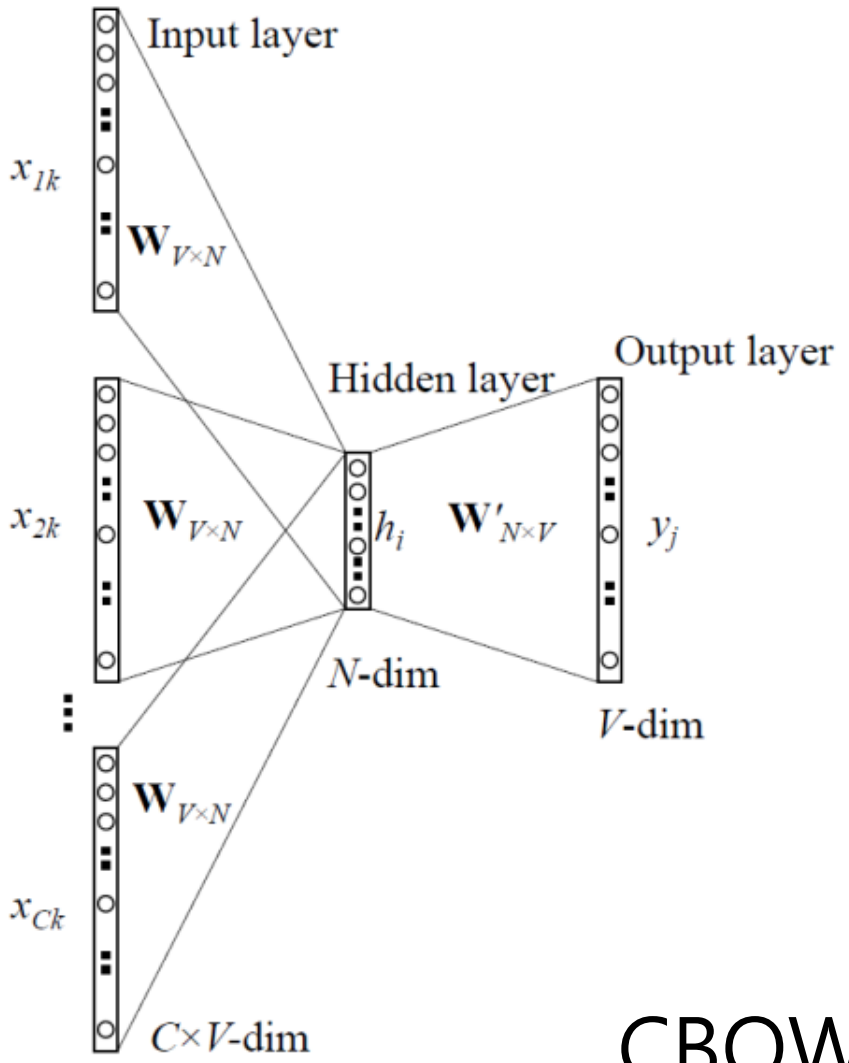
Word2vec Model

The Quick Brown Fox Jumps

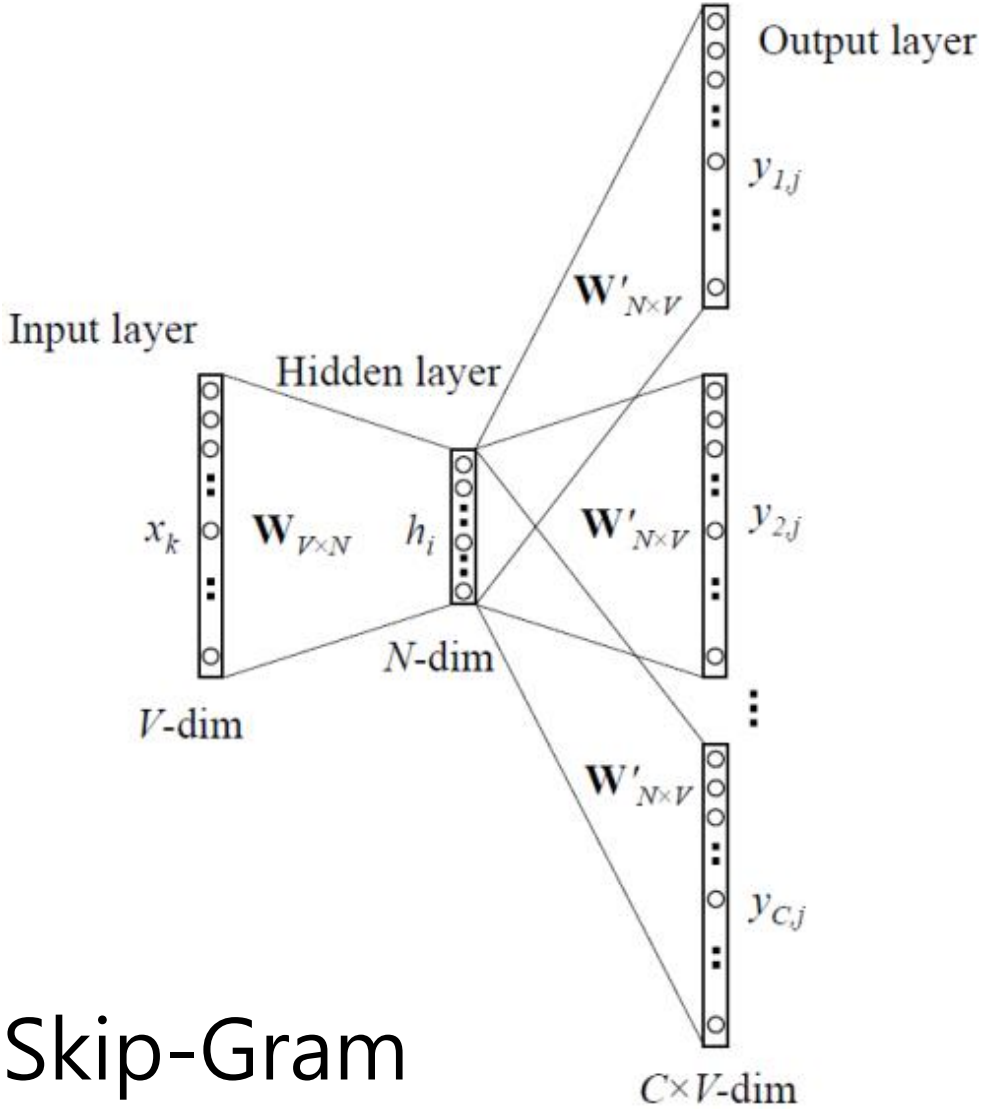
Context word | Center word | Context word

CBOW	Skip Gram
Context word로 Center word를 예측 ex) Input : 'The', 'Quick', 'Fox', 'Jumps' → Output : 'Brown'	Center word로 Context word를 예측 ex) Input : 'Brown' → Output :'The', 'Quick', 'Fox', 'Jumps'
전체 상황을 보고 예측 보다 매끄러우며, 적은 Data에서 유용	Context – Target 쌍을 취급 큰 Data에서 유용

Word2vec Model



CBOW



Skip-Gram

Complexity Reduction

- Word2vec의 두 모델의 많은 학습량 (영어 단어의 총 개수는 백만 개 이상)
- Output Layer에서 각 단어에 대해 전부 softmax 계산 후 normalization을 해야 함
- **방지책** : Hierarchical Softmax, Negative Sampling
- Negative Sampling : NCE를 단순화한 모델

How to structure your TensorFlow model

1. Assemble you Graph

- Define placeholders for input and output
- Define the weights
- Define the inference model
- Define loss function
- Define optimizer

2. Execute the computation