

Lesson 8-14

≡ Files	
≡ Tags	

Lesson 8. Introduction to Neural Networks

- Classification
- Perceptron, , XOR (NAND), neurons, VC dimension
- Discrete, continuous
- MLE
 - log, product, sum
- Cross entropy, RMSE
- Cost function vs Accuracy
- Gradient descent, chain rule, backprop

Lesson 9. Training Neural Networks

- training optimization
- overfitting vs underfitting
- regularization
- dropout
- batch normalization
- local minima
- random restart
- early stopping
- vanishing gradient

- activation functions
- batch vs stochastic gradient descent
- learning rate decay
- momentum
- error functions

Lesson 10

(생략)

Lesson 12. Embeddings & Word2vec

1. 단어 임베딩(Word Embedding)이란 텍스트를 구성하는 하나의 단어를 수치화하는 방법

word → embedded vector : 이 벡터를 embeddings 이라 함

text data의 차원 축소 + 단어 간 연관관계 파악(ex. 시제,성별,...)

문맥 속 단어 파악하기 때문에, 편향되거나 잘못된 정보가 있으면 잘못 임베딩 됨

2. one-hot encoding의 비효율성을 개선(행렬 연산 복잡)

M4L52 HSA Embedding Weight Matrix V2 RENDER V2

전체화면을 종료하려면 esc 을(를) 누르세요.

The diagram illustrates a neural network layer structure. On the left, a vertical column labeled 'inputs' contains five circles with values 0, 1, 0, 0, and 0. An orange arrow points to the second circle (value 1) with the text 'input corresponding to "possibility"'. From each input circle, a line connects to one of three green circles in a vertical column labeled 'hidden layer'. These connections represent the dot product of the input vector with the rows of a weight matrix. An orange arrow points from the hidden layer to an orange square labeled 'matrix of values', which is also labeled 'layer outputs' below it.

and this is really computationally inefficient.

1:04 / 2:38 스크롤해서 자세히 알아보기

**word → integer token(input) → embedding layer → embedding weight matrix
→ embedding vector(output)**

one-hot의 weight matrix를 곱하면 output은 한 개의 행

이때 embedding weight matrix를 lookup table 이라고도 함. lookup table 은 훈련 과정에서 학습된 columns 개수가 embedding dimension(보통 몇백개)

M4L52 HSA Embedding Weight Matrix V2 RENDER V2

전체화면을 종료하려면 esc 을(를) 누르세요.

inputs

0

1

0

0

0

second index

embedding layer

embedding weights matrix

1 8 5

2 4 1

0 6 3

9 2 7

7 3 0

second row

=

[2 4 1]

layer outputs

the index of the one or the on input unit.

1:45 / 2:38

스크롤해서 자세히 알아보기

center word context words

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

center word	context words
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0]
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,1,0,0,0]
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0] [0,1,0,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,1,0,0]
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,1,0,0]
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,1,0]
[0,0,0,0,0,1,0]	[1,0,0,1,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,1,0]
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0] [0,0,0,0,0,1,0]

그림 20.5.4 : CBOW 임베딩 예

<https://datascienceschool.net/view-notebook/6927b0906f884a67b0da9310d3a581ee/>

3. Word2Vec

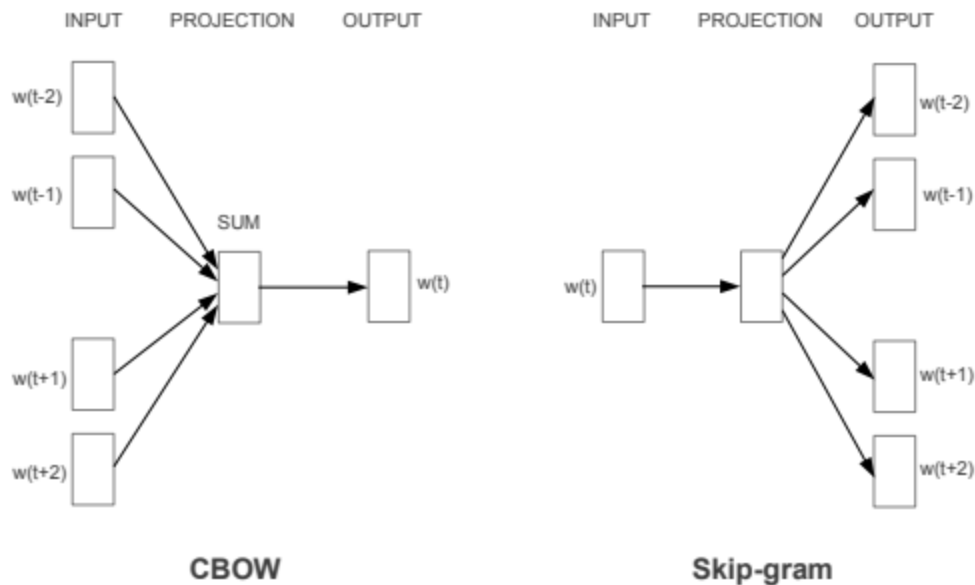
같은 문장 안에 등장한 단어는 비슷한 벡터 값을 가짐

1) CBOW(Continuous Bag Of Words)

문맥 속 '주변 단어'로 '현재 단어'를 예측

2) Skip-gram

'현재 단어'로 '주변 단어'를 예측 (k : window size)



3) word2vec

기본적인 임베딩 모형에 subsampling, negative sampling 등의 기법을 추가된 형태

(1) Loading Data

(2) Pre-processing : util 패키지 내 preprocess 함수 사용(부호를 토큰화 <PERIOD>)

(3) Dictionaries : lookup table 만들기 vocab \leftrightarrow int (가장 많이 등장:0 ~ 덜 등장:n) list로 저장

(4) Subsampling : 많이 등장한 common words 삭제 ($f(w_i)$ 는 해당 단어 등장한 비율, t 는 사용자가 지정 값 0.00001 권장)

(5) Making batches : 크기 C의 윈도우

Lesson 13

1. Model Architecture

- Embedding layer
 - 단어를 벡터로
- LSTM Layer
 - 각 LSTM cell의 입력:
 - 현재 t의 입력 벡터(=1층짜리 LSTM같은 경우, t 번째 단어 임베딩)
 - 이전 셀에서 가지고 있는 정보
 - 각 LSTM cell의 출력:
 - 벡터
 - 이 벡터들을 fully-connect
- Sigmoid output layer
 - 각 LSTM cell의 출력 벡터들을 fully-connect한 후 softmax에 태움

2. Data Preprocessing

- 필요없는 단어 제거, 줄바꿈/구두점 이런것들 다 띄어쓰기로 대체, 정규화
- Word → integer, integer → vector
- 너무 크기가 크거나 작은 텍스트는 훈련에서 배제
- 긍정의견은 1, 부정의견은 0으로 변경
- sequence padding:
 - 길이가 다른 문장들을 고속병렬처리하기 위함

3. Training

- torch 문법
- 나중에 쓰게될 프레임워크 따라서 배웁시다. 저는 tensorflow 파라서...
- AutoML 같은 거 쓸거면 이 조차도 필요없음

- 지금 배우는 RNN 감정분석은 꽤나 널리 알려진거라 거의 제품화 되어있을거예요
- Train, Validation, Test
 - Train vs Validation: parameter(=알고리즘이 결정하는 값)의 검증
 - Train/Validation vs Test: hyperparameter(=아직까지는 사람이 결정하는 값)의 검증

Lesson 14

(생략)