Natural Language Processing with DeepLearning

week 5

The course

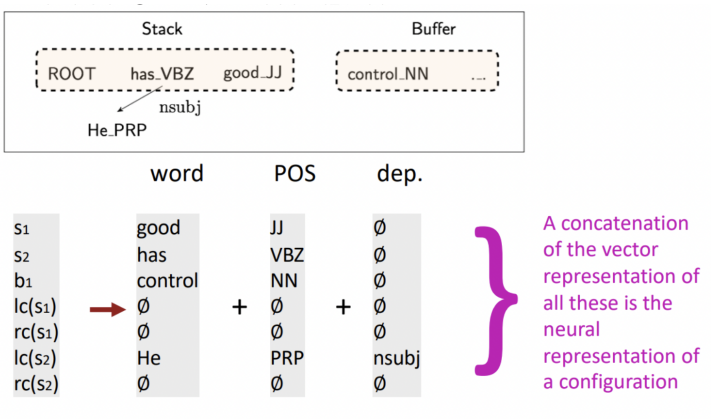
- Neural dependency parsing
- A bit more about neural networks
- Language modeling + RNNs
  - A new NLP task: Language Modeling (motivates)
  - A new family of neural networks: RNN

1. Neural dependency parsing

< 성능**,** 속도 향상 이유 **>**

1) Distributed Representation

- word embedding 과 같은 방식을 통한 각 단어의 d차원의 dense 벡터로 표현
- 품사, dependency labelseh d-차원의 벡터로 표현
  : 적은수이지만 벡터로 표현하면 유사한 품사는 인접하게 됨!

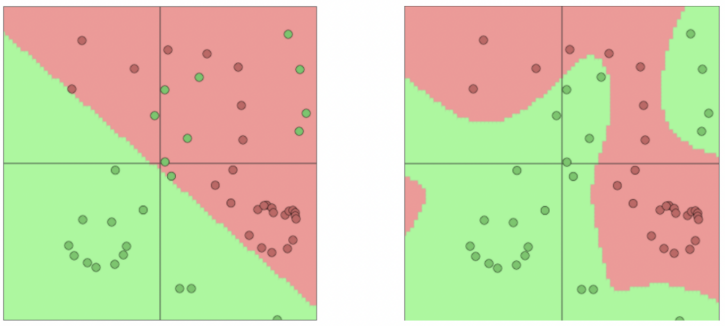


2) 비선형 딥러닝 분류기

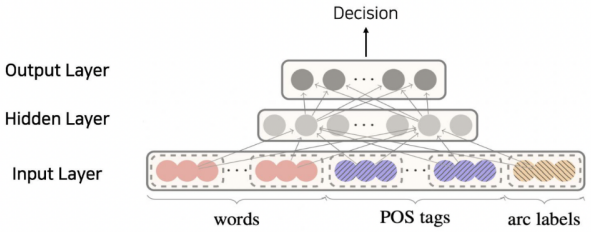
- 고전적인 ML 분류기들(Naive Bayes, SVMs, logistic regression 그리고 softmax 포함)은 선형 decision boundaries를 제공하기에 그렇게 강력한 분류기는 아님.

$W \in \mathbb{R}^{C \times d}$의 가중치 행렬을 손실함수를 최소화하는 방향으로 학습!

$$P(y|x) = \frac{exp(W_y \cdot x)}{\sum_{c=1}^{C} exp(W_c \cdot x)}, y \in C, x \in \mathbb{R}^{\mathrm{d}}$$

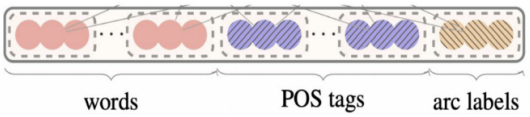


#4 Neural Dependency Parser



Neural dependency parser는 모델 구조: 기본적인 feed forward network와 닮았음
그러나, feature representation하는 과정과 hidden layer의 활성화 함수에서 차별점

Feature Selection



- STACK과 BUFFER의 Top 3 단어 (6개)
  $[s_1, s_2, s_3, b_1, b_2, b_3]$
- STACK Top 1, 2 단어의 첫번째, 두번째 left & right child 단어 (8개)
  $[lc_1(s_1), rc_1(s_1), lc_2(s_1), rc_2(s_1), lc_1(s_2), rc_1(s_2), lc_2(s_2), rc_2(s_2)]$
- STACK Top 1, 2 단어의 left of left & right of right child 단어 (4개)
  $[lc_1(lc_1(s_1)), rc_1(rc_1(s_1)), lc_1(lc_1(s_2)), rc_1(rc_1(s_2))]$

word features의 해당 POS tag (18개)

word features의 해당 arc-label (12개)

Input으로 입력되는 feature: words, POS tag, arc labels

1) words feature 데이터: 총 18개로 구성
STACK과 BUFFER의 TOP 3 words (6개)

STACK TOP 1, 2 words의 첫번째,
두번째 left & right child word (8개)

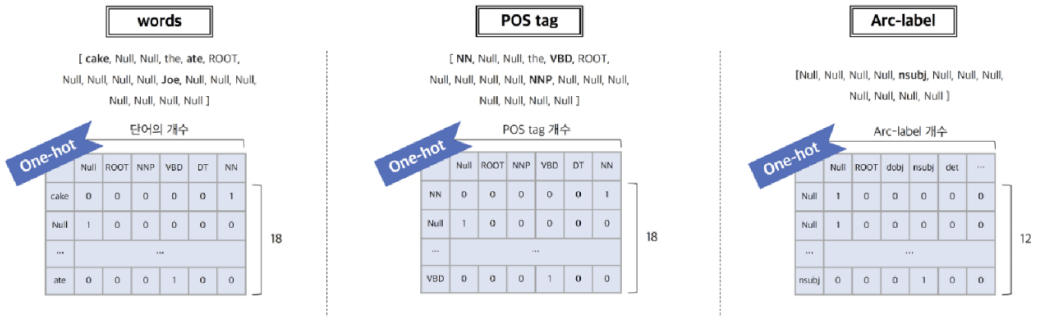
STACK TOP 1,2 words의 left of left & right of right child word (4개)

2) POS tags feature데이터:
words feature에서 들어가는 데이터의 태그를 의미 (18개)

3) Arc labels 데이터:
STACK과 BUFFER의 TOP 3 words 6개를 제외한 12개의 label(dependent 관계 표시)

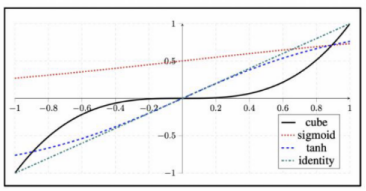
#4 Neural Dependency Parser

One hot Representation



#4 Neural Dependency Parser

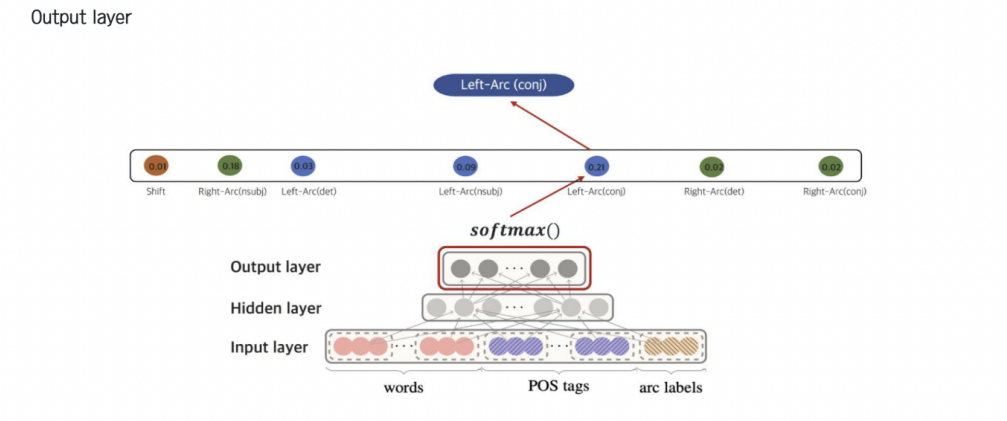
Hidden layer



$$g(x) = x^3$$

$$g(w_1x_1 + \ldots + w_mx_m + b) = \sum_{i,j,k}(w_iw_jw_k)x_ix_jx_k + \sum_{i,j} b(w_iw_j)x_ix_j \ldots$$

Words, POS tag, Arc-label간
상호작용을 반영할 수 있는 조합

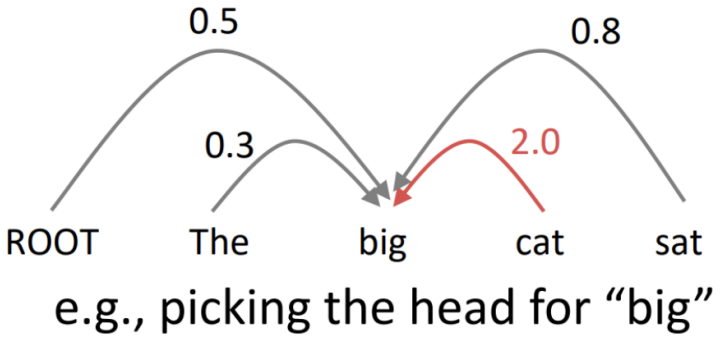


○ **Transition-based neural dependency parsing** 의 발전

Method	UAS	LAS (PTB WSJ SD 3.3)
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79

○ **Graph-based dependency parser**

: 각 단어의 모든 가능한 head의 확률을 계산하는 방식으로 단순하게 두단어의 관계 뿐만 아니라 모든 단어 간의 관계를 알 수 있음

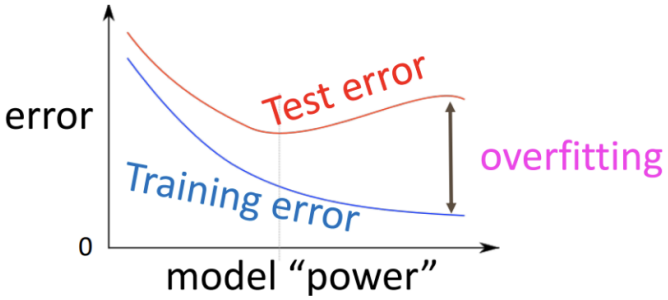


2. A bit more about neural networks

1) Regularization

모델을 학습할 때 매개변수가 정말로 유용한 경우에만 매개변수를 0이 아닌값으로 만들려고 하는데, 정규화 항을 추가하면 매개변수가 큰 도움이 되지 않는

범위에서는 0이 아닌 값으로 함으로써 불이익을 받게 됨. 과적합 규제에 사용.



2) DropOut
Preventing Feature Co-adaptation = 좋은 정규화 방법
여기서 Co-adaptation이란 뉴런들 간 상호의존적이게 되는 것으로 서로 같은 특징에 대해 학습하기 때문에 중복적인 역할을 하게된다. 이를 막기 위해서 드롭아웃(Dropout)을 사용한다.

Train: 평가할 때 일부 뉴런에 대한 입력값을 0으로 한다(비활성화)
Test: 모든 모델 가중치를 이용하여 평가

3) Vectorization

4) Non linearities, Old and New

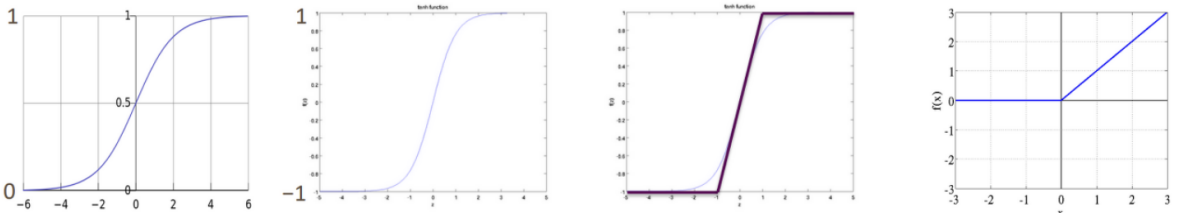
logistic ("sigmoid") tanh hard tanh ReLU (Rectified Linear Unit)

$$f(z) = \frac{1}{1+\exp(-z)}$$

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$

$$\text{rect}(z) = \max(z, 0)$$



5) Parameter Initialization
가중치: Uniform (-r, r) 범위에서 초기화를 진행하며, 편차의 경우는 0으로 초기화,
대표적 방법 : Xavier initialization

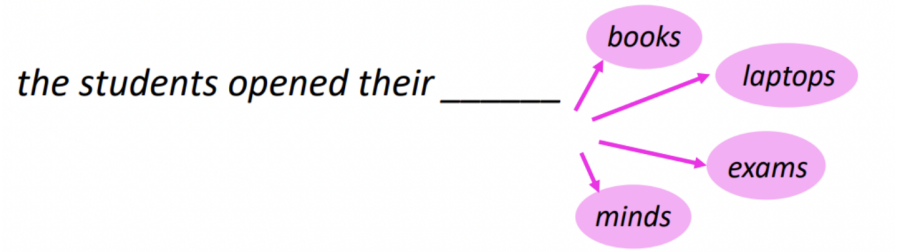
6) Optimizers
Adagrad, RMSprop, Adam(일반적), SparseAdam

7) Learning Rate
학습률은 학습 시 기울기를 기반으로 얼마만큼 매개변수를 변화시킬지를 결정하는 상수이며 일반적으로 10^-3 이용
학습률이 큰 경우: 모델이 발산하거나 수렴에 실패
학습률이 작은 경우: 모델이 수렴하는데 오랜 시간이 소요

3. Language modeling + RNNs

1) Language Modeling

: 문맥을 보고 다음에 오는 단어가 무엇인지를 예측하는 작업 (다음 단어의 확률 분포 계산)



$$P(x^{(t+1)}|x^{(t)},\dots,x^{(1)})$$

문서의 일부를 확률로 변환 -> 아래식 : 위의 수식예시에서 해당 문서의 확률

$$P(x^{(1)},x^{(2)},\dots,x^{(t)}) = P(x^{(1)}) \times P(x^{(2)}|x^{(1)}) \times \dots \times P(x^{(t)}|x^{(t-1)},\dots,x^{(1)})$$

$$= \prod_{i=1}^{t} P(x^{(i)}|x^{(i-1)},\dots,x^{(1)})$$

2) N-gram language models in practice
언어모델 학습방법? => n-gram 언어모델 학습!

- unigrams: “the”, “students”, “opened”, ”their”
- bigrams: “the students”, “students opened”, “opened their”
- trigrams: “the students opened”, “students opened their”
- 4-grams: “the students opened their”

ex) 4-gram language model

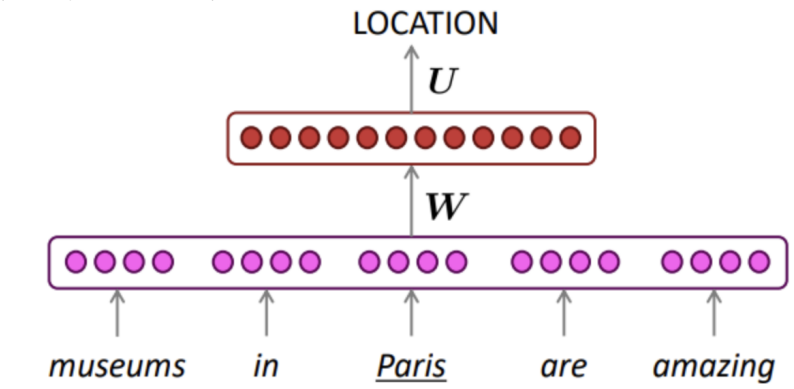
~~as the proctor started the clock, the~~ students opened their ____

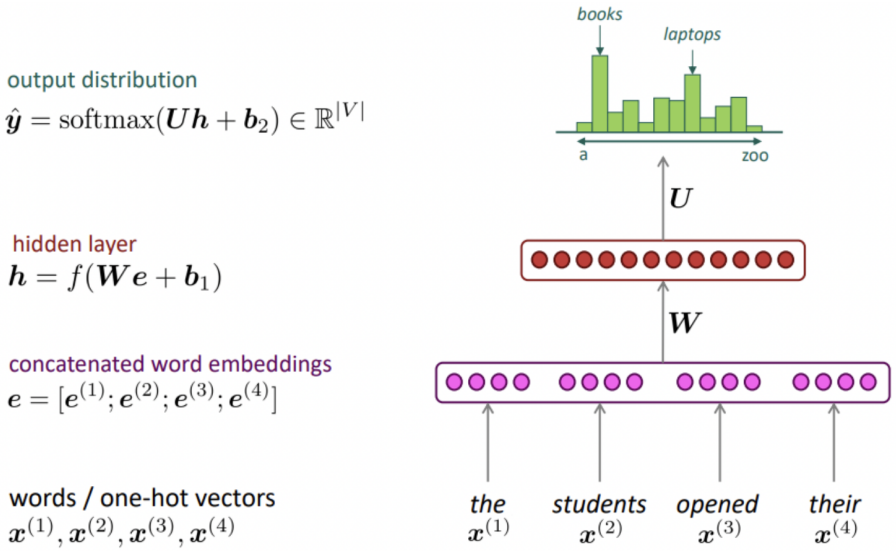
$$P(w|\texttt{students opened thier}) = \frac{count(\texttt{students opened thier}\ w)}{count(\texttt{students opened thier})}$$

만약 "students opened thier"가 1000개 있을 시,

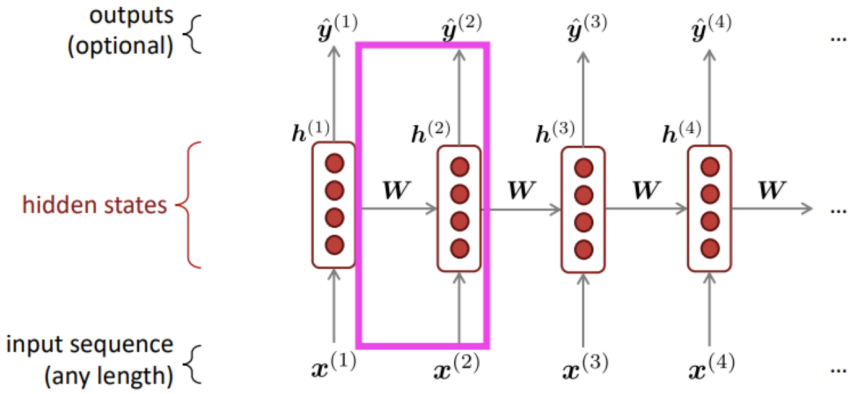
- "students opened thier books"가 400개 있으면 → $P(\texttt{books}|\texttt{students opened thier}) = 0.4$
- "students opened thier exams"가 100개 있으면 → $P(\texttt{exams}|\texttt{students opened thier}) = 0.1$

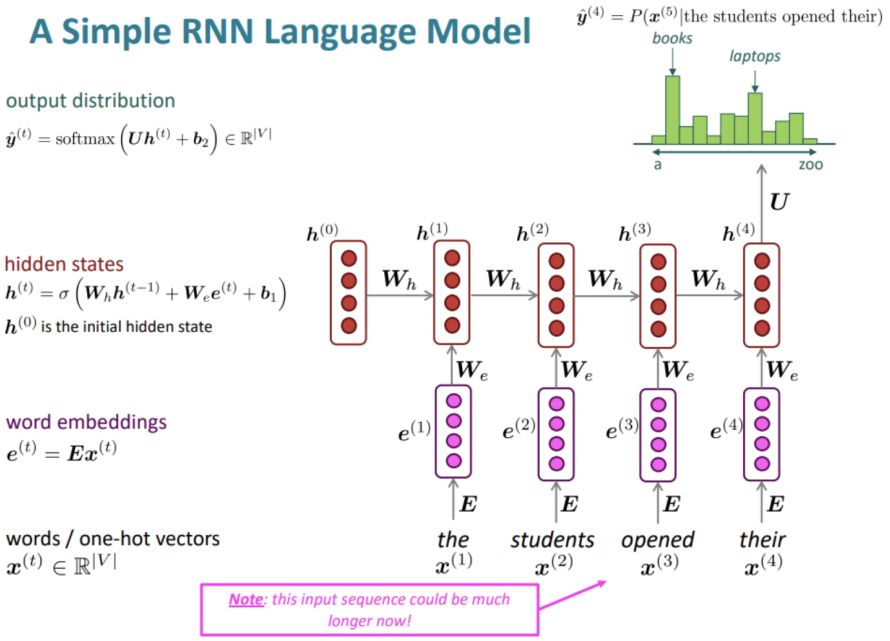
3) How to build a Neural Language model





4) RNN(Recurrent Neural Networks)





- RNN 장점
  1) 임의의 입력값을 처리 가능하다
  2) (이론상) 여러 timestep 전의 정보를 이용하여 계산이 가능하다
  3) 모델의 크기가 입력 크기에 영향을 받지 않는다
  4) 같은 가중치가 모든 timestep에 적용된다. 즉, 입력이 어떻게 처리되는지에 대칭성이 존재한다

- RNN 단점
  1) 은닉층 밖에서 반복 구문을 수행해야하기 때문에 연산이 느리다
  2) 실 사용시 많은 timestep 전의 정보를 잘 기억하지 못한다