220128 CS224N Lecture 3,4

[Neural Network in NLP]

1. NER (Named Entity Recognition)

- 이름을 가진 개체를 인식하는 것 → 어떤 단어가 사람인지, 장소인지 파악하는 것
- 사람(PER), 장소(LOC), 날짜(DATE) 등 **고유명사를 분류**하는 방법론

  → 문장은 고유명사를 기준으로 의미 형성 (NLP에서 문맥을 파악하고 글을 해석하는 데 있어 중요한 역할을 함)

Last night , Paris Hilton wowed in a sequin gown .
PER PER

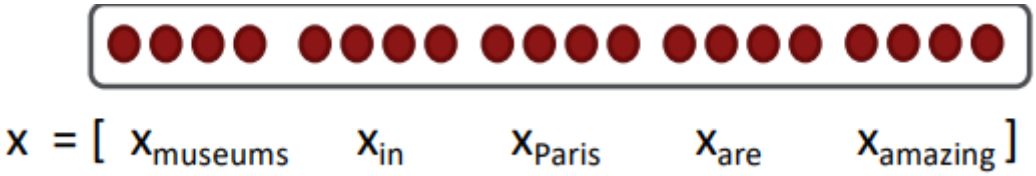
Samuel Quinn was arrested in the Hilton Hotel in Paris in April 1989 .
PER PER LOC LOC LOC DATE DATE

- 'paris' 라는 단어가 장소인지 사람인지 파악하기 위해서는 그 단어만 바라보는 것이 아닌 **context** 필요
- NLTK 내 NER chunker 통해 간단하게 구현가능함

WIndow classification using binary logistic classfier

> 💡 중심 단어와 주변 단어들을 함께 분류 문제에 활용하는 방법

- center word와 주변 context word를 함께 분류문제에 활용하는 방법 (window size= context word의 개수)



  - 'paris'가 사람 이름인지 장소인지 분류하기 위해 주변 단어 활용 (window =2)

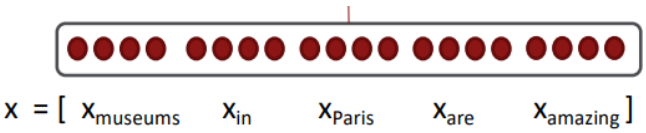
- 5D 크기의 벡터 생성
- averge (window 내 word vector를 평균내고 평균 vector를 분류하는 방법)→ 단순 평균이기 때문에 위치 정보를 잃어버릴 수 있음

- **binary logistic classfier**

  : input으로 x 대신 x_window가 들어갔다고 생각

  1) input (X) → 5D vector

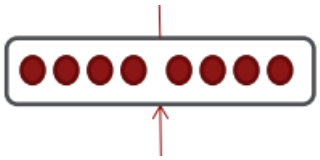
$$\boldsymbol{x} \quad (\text{input})$$



  2) hidden layer(input보다 크기 small) → activate function

  각 x에 가중치 곱한 weighted sum에 bias합한 layer

$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$



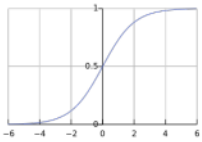
  3) dot product→ single vlaue

$$s = \boldsymbol{u}^T \boldsymbol{h}$$

  4) ‘Paris’ 가 장소면 high score 출력하도록 train

  → sigmoid function 활용

$$J_t(\theta) = \sigma(s) = \frac{1}{1 + e^{-s}}$$



- softmax classifier도 많이 활용

Stochastic Gradient Descent

- Backpropagation(역전파)를 활용하여 손실함수(J(theta)) 최소화해야함
- 파라미터가 손실값에 얼마나 많은 기여를 했는지 확인하고 이에 따라 파라미터 조정 과정 필요

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

α = *step size or learning rate*

- 계산 방법: By hand / 역전파 알고리즘 활용

2. Matrix calculus

- SGD 과정을 수식으로 이해하기 위해 여러 matrix 계산법을 알아보자!

Chain rule

- one-varible → multiply derivatives

$$z = 3y$$
$$y = x^2$$
$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = (3)(2x) = 6x$$

- multiple variable → multiply Jacobians

$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \dots$$

Jacobian Matrix

- n input . m output → m*n matrix
- partial derivatives w/ respect to each input

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \qquad \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{ij} = \frac{\partial f_i}{\partial x_j}$$

- examples

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \qquad \text{definition of Jacobian}$$

$$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases} \qquad \text{regular 1-variable derivative}$$

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} = \begin{pmatrix} f'(z_1) & & 0 \\ & \ddots & \\ 0 & & f'(z_n) \end{pmatrix} = \text{diag}(\boldsymbol{f}'(\boldsymbol{z}))$$

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{W}$$

$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I} \quad \text{(Identity matrix)}$$

$$\frac{\partial}{\partial \boldsymbol{u}}(\boldsymbol{u}^T \boldsymbol{h}) = \boldsymbol{h}^{\boldsymbol{T}}$$

loss function의 gradient 계산

1) Break up equations into simple pieces

Wx+b를 z로 치환하여 매개변수 하나 생성

$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \quad \Rightarrow \quad \boldsymbol{h} = f(\boldsymbol{z}), \quad \boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

2) Apply the chain rule

chain rule활용하여 s를 b로 미분한 값 도출

$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad (\text{input})$$

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

3) Write out the Jacobians

미분한 계산 과정 대입하여 최종 식 도출

$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad (\text{input})$$

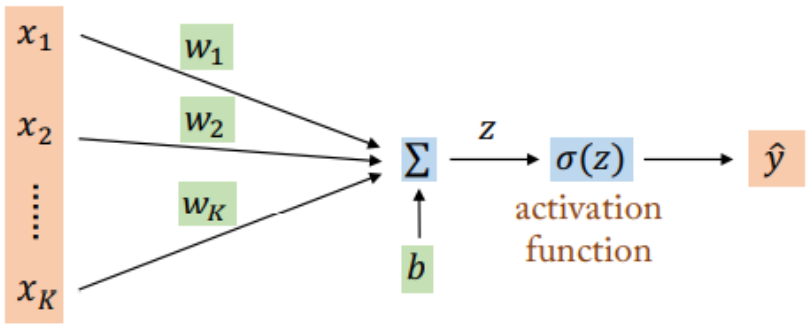
$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$
$$= \boldsymbol{u}^T \operatorname{diag}(f'(\boldsymbol{z}))\boldsymbol{I}$$
$$= \boldsymbol{u}^T \circ f'(\boldsymbol{z})$$

Useful Jacobians from previous slide

$$\frac{\partial}{\partial \boldsymbol{u}}(\boldsymbol{u}^T\boldsymbol{h}) = \boldsymbol{h}^T$$
$$\frac{\partial}{\partial \boldsymbol{z}}(f(\boldsymbol{z})) = \operatorname{diag}(f'(\boldsymbol{z}))$$
$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I}$$

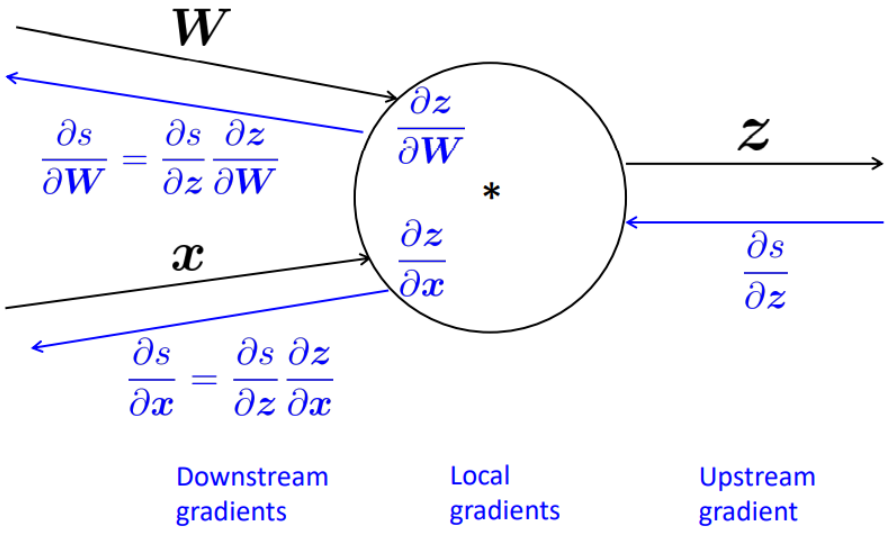
3. Backpropagation (역전파)

- layer가 많아질 때의 학습 방법으로, gradient를 효과적으로 update하기 위한 계산법
- Goal: loss(cost)를 최소화시키는 parameter 찾기
- cost C를 줄여나가도록 parameter w를 update해나감
- Single layer
  - Forward Pass (FeedForward): weighted sum에 bias 더하고 활성화 함수 사용하여 output vector 출력
    - input: X vector

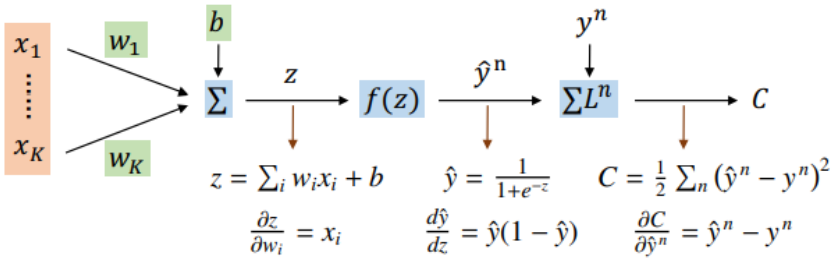


$$\hat{y} = f(z) = f(\sum w_i x_i + b) = f(\boldsymbol{w}^T \boldsymbol{x} + b)$$ Forward Pass

    - Backpropagation
      - output vector를 반대로 미분하면서 cost가 최소가 되도록 가중치를 업데이트 하는 과정
      - chain rule 활용



$$w_i(t+1) = w_i(t) - \eta \frac{\partial C}{\partial w_i}$$

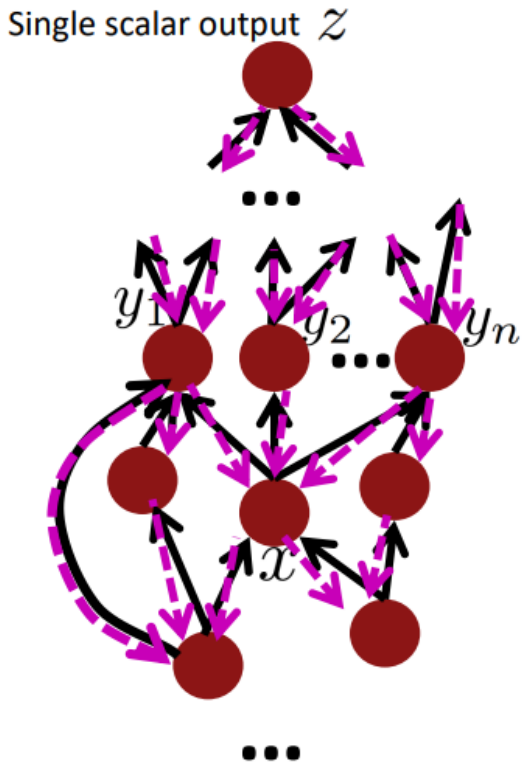


$$\frac{\partial C}{\partial w_i} = \sum_{n=1}^{N} \frac{\partial z^n}{\partial w_i}\frac{d\hat{y}^n}{dz^n}\frac{\partial C}{\partial \hat{y}^n} = \sum_{n=1}^{N} x_i^n \hat{y}^n(1-\hat{y}^n)(\hat{y}^n - y^n)$$

Chain rule

computation Graph

- 값을 그래프 노드 위에 위치시키는 것
- Forward : topological sort로 정렬한 뒤 노드 지남
- Backward : output node에서 1로 gradient 시작, topological sort의 반대 순서로 노드 지남
- 순전파 과정과 역전파 과정에 시간복잡도는 동일



- 딥러닝 프레임워크들은 그래프 기반으로 연산 진행 → 그래프에 위치시켜 연산을 진행하면 효율적 연산 진행 가능

Numeric Gradient

- 아주 작은 값을 대입하여 순간 기울기를 구하는 것→ gradient를 잘 구했는지 체크 가능
- NN에서는 해당 방법으로 연산 진행하지는 않음 → 계산 과정이 너무 많아서 비효율적으로 연산 진행

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

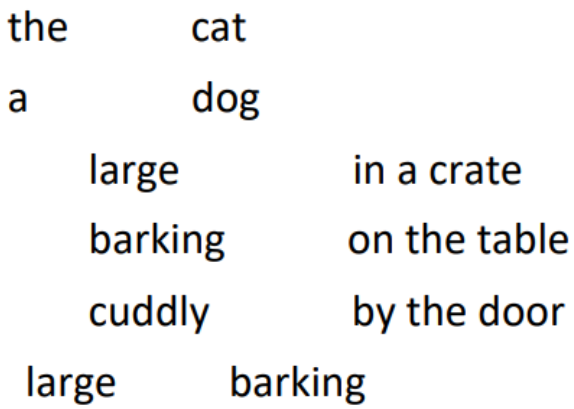
[Dependency Parsing]

1. Syntactic Structure: Consitituency and Dependency

📎 Pharse (Constituency)

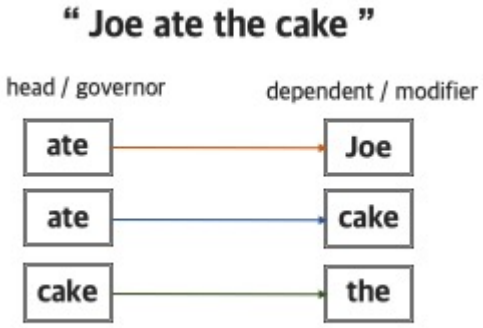
- 각 문장의 문법적인 구성, 구문을 분석하는 과정
- 문장의 의미를 보다 정확하게 파악하기 위해서 필수적인 과정
- 언어는 복잡한 단어의 구성을 통해 이루어진→ 문장 연결 구조에 대한 이해 필요
- 목적에 따라 **Consitituency parsing**과 **Dependency parsing**로 구분 가능
- **Consitituency parsing(=** structure grammar = context-free grammars (CFGs)**)**

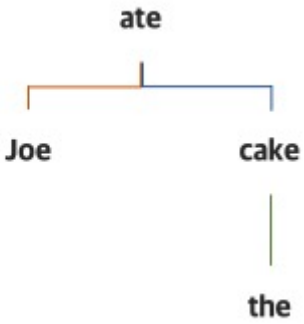
  : 문장의 구성요소를 파악하여 구조 분석
  - 관형사 ,명사 등등 문장의 구성요소 파악 → 단어- 단어 / 구- 구 → 문장을 이룸



  - 영어처럼 어순이 비교적 고정적인 언어에서 주로 사용됨
- **Dependency parsing**: 단어 간 의존 관계 파악하여 구조 분석

  : 문장에 존재하는 단어 간의 의존 또는 수식 방향으로 관계를 파악하여 문장 구조 분석
  - 한국어처럼 자유 어순을 가지거나 문장 성분 생략 가능한 언어에서 선호, 최근 영어에 적용하는 연구 활발히 진행
  - 수식하는 단어 : head or gorverner / 수식 받는 단어 : dependent or modifier





- **Ambiguity (모호성)**
  - PP(Prepositional Parse) attachment Ambiguity : 형용사구, 동사구, 전치사구 등이 어떤 단어를 수식하는지에 따라 의미가 달라지는 모호성

Scientists count whales from space

Scientists count whales from space

과학자들이 우주에 고래를 세다 vs 과학자들이 우주에서 온 고래를 세다

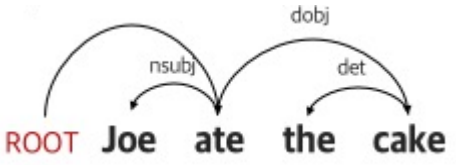
  - coordination scope ambiguity: 특정 단어가 수식하는 대상의 범위가 달라짐에 따라 의미가 변하는 모호성

  - VP(Verb Pharse) attachment Ambiguity: 하나의 동사에 많은 뜻을 가지고 있음에 따른 모호성

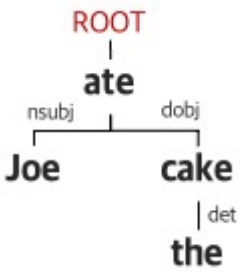
⇒ 문장은 많은 단어가 복잡한 관계로 얽혀있으므로 문장 구성요소를 명확히 알고 분석하는 것이 중요

2. Dependency Grammar and Treebanks

- 두가지 형태로 표현하는
- sequence



- tree



- 두가지 형태의 output은 동일해야함
- ROOT: 가장의 노드, 문자 맨 처음에 추가 → head로 설정

  → 모든 단어가 최소 한개의 node에 의존될 수 있도록 함
- 화살표는 순환하지 않으면 중복 관계 X, head에서 dependent하도록 방향 진행