Natural Language Processing with DeepLearning

week 3

NER(Named Entity Recognition

: 개체명 인식 (자연어 처리 태스크)- 텍스트에서 name을 찾고 분류하는 것

The task: find and classify names in text, for example:

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

=> 많은 텍스트에서 자동으로 지식 기반 구축을 시작하려는 경우 일반적으로 하고 싶은 것은 명명된 entity를 가져와서 이들간의 관계를 파악하는 것

Possible Purpose

- 문서에서 특정entity에 대한 언급 추적
- 질문에 대한 답변의 경우, 답변은 보통 인명
- 요구되는 많은 정보들은 인명간의 관계에 관한 것인 경우가 많음
- 동일한 기술들이 다른 slot-filling classification으로 확장될 수 있음

Matrix Calculus

<Jacobian Matrix>

- Given a function with 1 output and *n* inputs

$$f(\boldsymbol{x}) = f(x_1, x_2, ..., x_n)$$

$$\frac{\partial f}{\partial \boldsymbol{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right]$$

- Given a function with ***m* outputs** and *n* inputs

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(x_1, x_2, ..., x_n), ..., f_m(x_1, x_2, ..., x_n)]$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$\boxed{\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{ij} = \frac{\partial f_i}{\partial x_j}}$$

- 왼쪽(함수 한 개의 미분): n개의 input을 넣으면 한 개의 output 산출. 이를 미분하면 n개의 각각의 input으로 미분하여 하나의 벡터가 됨.
- 오른쪽(matrix에서의 미분): m개의 함수 각각에 n개의 input이 들어간다고 할 때, 이를 미분하면 input과 output의 모든 조합을 고려하여 m*n matrix가 되고 이를 Jacobian Matrix라 함

<Chain Rule>

- For composition of one-variable functions: **multiply derivatives**

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

- For multiple variables at once: **multiply Jacobians**

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

Stanf

<Example Jacobian: Elementwise activation Function>

Example Jacobian : Elementwise activation Function

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$
$$h_i = f(z_i)$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Function has *n* outputs and *n* inputs → *n* by *n* Jacobian

H는 z를 input으로 집어넣어 활성화함수를 적용해 나온 식임. 이를 z로 미분한다면 우리가 아까 봤던 매트릭스와 비슷한 형태로 nxn의 매트릭스 형태가 나옴

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \qquad \text{definition of Jacobian}$$
$$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases} \qquad \text{regular 1-variable derivative}$$

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

z(i) 와 z(j) 가 같을 때 미분이 됨
다르면 0으로 없어짐

미분한 식을 알아보면 hi는 f(zi)로 나타낼 수 있고, zi 와 zj가 I = j 가 같을 때 즉, 변수가 같을 때 미분했을 때 값이 나오게 되고, 다른 경우는 0으로 값이 나오게 됨
이를 행렬로 표현하면 그림과 같이 대각 행렬이 나옴. 이 계산은 우리가 window classification에서 손실 함수를 최소화하기 위한 미분 과정에 부분으로 들어감. 이를 후의 계산에 대입할 것

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{W}$$
$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I} \ \text{(Identity matrix)}$$
$$\frac{\partial}{\partial \boldsymbol{u}}(\boldsymbol{u}^T\boldsymbol{h}) = \boldsymbol{h}^{\boldsymbol{T}}$$

$$+$$

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

첫 번째는 wx + b를 x로 미분한 값 가중치가 나옴, 두 번째는 b로 미분한 값 항등행렬이 나옴, 세 번째는 u로 미분한 값 h를 전치한 값이 나옴.

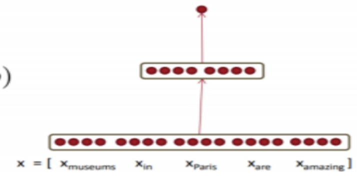
=> 이 세 가지 식과 전에서 봤던 식 모두 우리가 아까 보았던 window classification의 손실 함수를 최소화하기 위해 미분하는 과정에서 사용되는 식들임

Back to our Neural Net!

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

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

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



Let's find $\frac{\partial s}{\partial \boldsymbol{b}}$

손실함수의 gradient를 계산해야 하지만,
쉽게 score의 gradient를 먼저 계산해보자!

Back to Neural Net! => 손실함수의 gradient를 계산하자 (score함수의 gradient를 계산)
이 score 함수의 gradient를 계산하는 설명은 지금까지의 chain rule과 우리가 계산했던 4가지 식들을 단순하게 조합하면 됨.

1. Break up equations into simple pieces

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

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

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

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

2. Apply the chain rule

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

$$\boxed{\boldsymbol{h} = f(\boldsymbol{z})}$$

$$\boxed{\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}}$$

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

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

3. Write out the Jacobians

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

$$\boldsymbol{h} = f(\boldsymbol{z})$$

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

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

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

$$= \boldsymbol{u}^T \text{diag}(f'(\boldsymbol{z}))\boldsymbol{I}$$

$$= \boldsymbol{u}^T \circ f'(\boldsymbol{z})$$

Useful Jacobians from previous slide

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

$$\boxed{\frac{\partial}{\partial \boldsymbol{z}}(f(\boldsymbol{z}))} = \text{diag}(f'(\boldsymbol{z}))$$

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

Re – using Computation

Using the chain rule again:

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

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

파란색 부분의 계산과정이 같다.
계산을 줄여주는 장점!

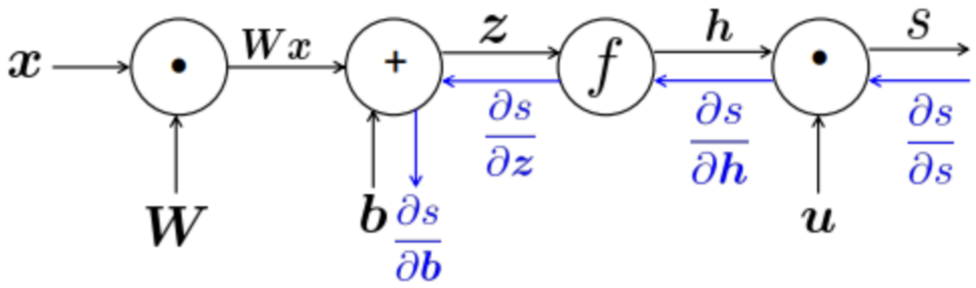
$$\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}}$$

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

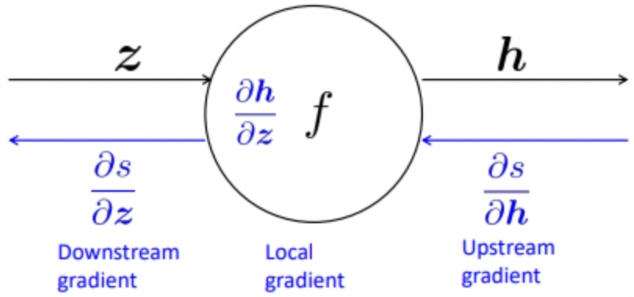
$$\boldsymbol{\delta} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} = \boldsymbol{u}^T \circ f'(\boldsymbol{z})$$

δ is local error signal

Back Propagation



- Backpropagation
  Forward Propagation을 통해 얻어진 결과와 실제값을 비교해 계산된 오차를 미분하며 가중치를 업데이트 및 학습.
- Backpropagation 과정에서 중요한 요소
  (1) Local Gradient
  (2) Downstream Gradient

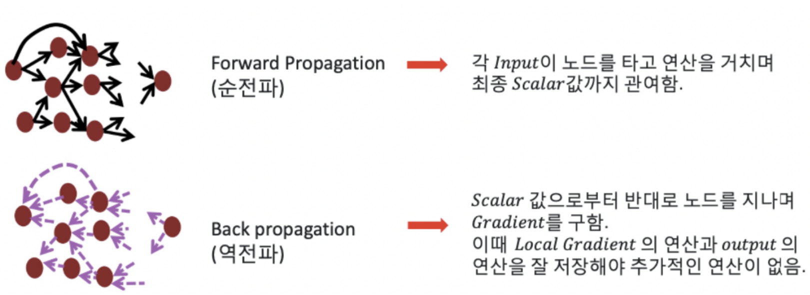


하나의 input과 하나의 output으로 이루어진 single node 그림
Back propagation 과정을 위해 진행 과정에서 Upstream gradient와 Downstream gradient의 순서로 미분값을 계산함

1. Local gradient는 Forward Propagation 수행 시의 output을 input으로 미분
2. 그 다음 Downstream gradient는 Local gradient에 Upstream gradient을 곱하여 도출

결론적으로 Chain Rule을 이용한 것과 같은 결과이다.

<Computation Graph>



검정색 화살표는 순전파가 진행되는 과정
보라색 화살표는 순전파를 통해 나온 최종 값을 바탕으로 오류를 전파하며 거꾸로 값을 업데이트 해가는 역전파과정