3. Backprop and Neural Networks

1. Named Entity Recognition(NER)

- 텍스트(문장) 내에서 단어를 찾고 분류하는 작업

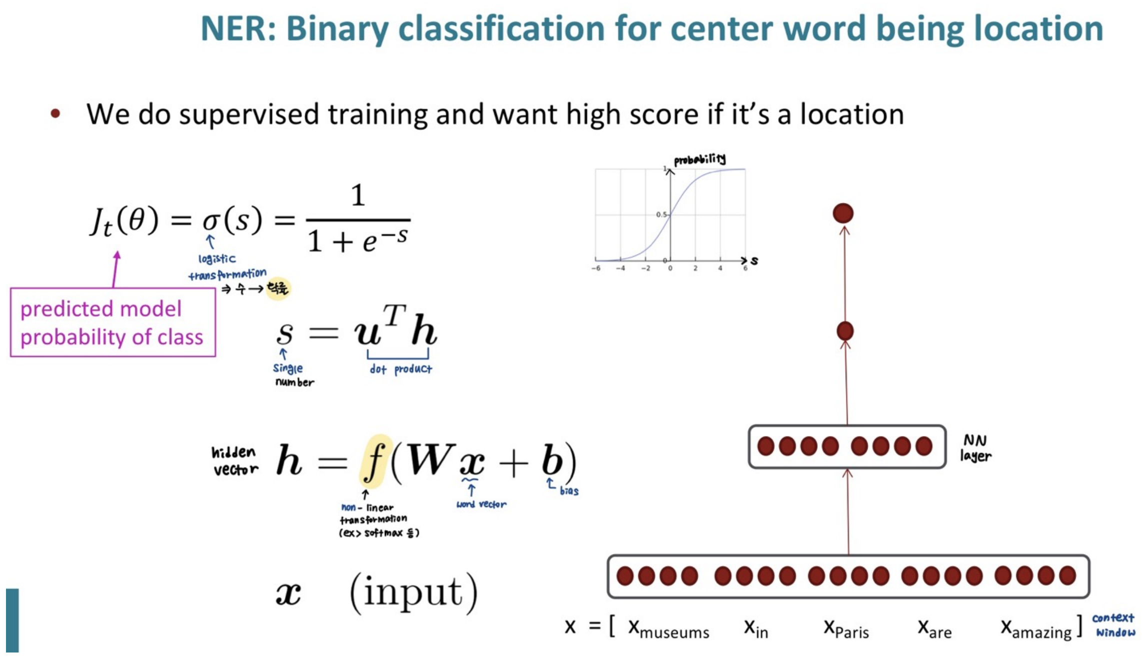
Last night , Paris Hilton wowed in a sequin gown .
PER PER
Samuel Quinn was arrested in the Hilton Hotel in Paris in April 1989 .
PER PER LOC LOC LOC DATE DATE

- **context window**를 활용하여 간단히 구현 가능

Example: Classify "Paris" as +/– location in context of sentence with window length 2:

the museums in Paris are amazing to see .

$X_{window} = [\ x_{museums} \quad x_{in} \quad x_{Paris} \quad x_{are} \quad x_{amazing}\ ]^T$



2. Gradients

- input data가 변화 했을 때 output이 얼마만큼 변화하는지를 나타내는 값
  - 각 vector에 대한 편미분 값(partial derivatives)
- multi-input, one-output case
  - 각 input에 대한 partial derivative를 모은 vector가 gradient

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

- multi-input, multi-output case
  - gradient를 matrix 형태로 표현 → Jacobian
  - jacobian matrix는 gradients의 **일반화**된 형태

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

Chain Rule

- 합성함수의 derivative는 각각의 derivative의 곱으로 얻을 수 있음

- chain rule을 multi-variable 함수에 적용하면 jacobian의 곱으로 확장 가능

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

- chain rule을 사용하면 계산을 재사용할 수 있게 됨

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

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

The same! Let's avoid duplicated computation ...

Neural Net에 적용하기

1. 각각을 부분으로 나누기

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

$$\Rightarrow$$

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

linear transformation

2. Chain Rule 적용
3. Jacobian 구하기

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

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

Useful Jacobians from previous slide

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

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

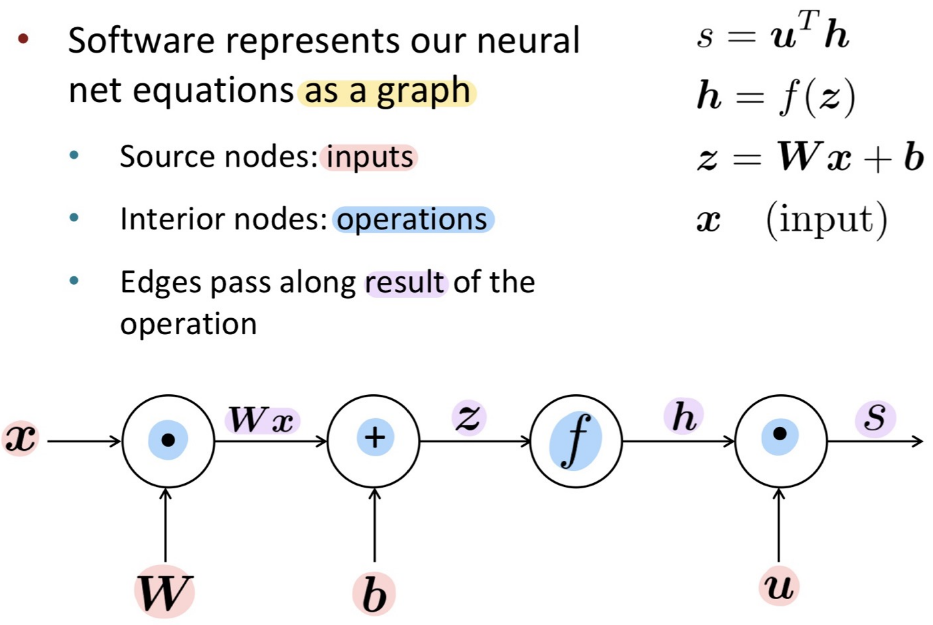
$$
\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I}
$$

Derivatives는 어떤 형태여야 할까?

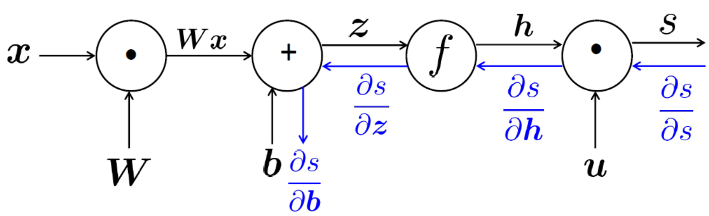
1. Jacobian 형태를 최대한 유지하고, 맨 마지막에 형태 맞추기
2. 형태만을 계속 생각하며 연산하기

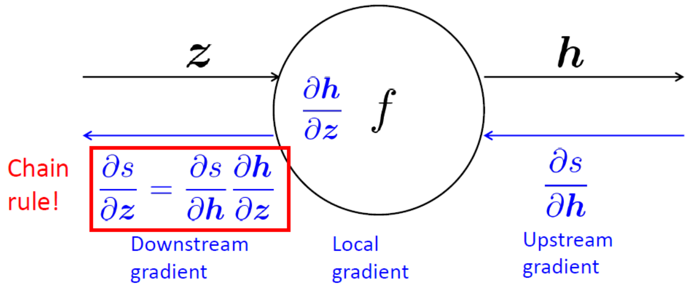
3. Backpropagation

- Neural Network 연산은 아래와 같은 그래프 형태로 표현할 수 있음



  - 앞에서부터 차례대로 그래프 연산을 진행하는 것을 **forward propagation**이라 함
- output부터 시작하여 역으로 gradients를 계산할 수 있음 ⇒ **back propagation**
  - for parameter update
  - 이때 chain rule을 활용 → 연산된 gradients를 재사용 할 수 있음


$\boldsymbol{x}$
$\boldsymbol{W}$
$\boldsymbol{Wx}$
$\boldsymbol{b}$
$\frac{\partial s}{\partial \boldsymbol{b}}$
$\boldsymbol{z}$
$\frac{\partial s}{\partial \boldsymbol{z}}$
f
$\boldsymbol{h}$
$\frac{\partial s}{\partial \boldsymbol{h}}$
$\boldsymbol{u}$
s
$\frac{\partial s}{\partial s}$


$\boldsymbol{z}$
$\boldsymbol{h}$
$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}$
f
Chain rule!
$\frac{\partial s}{\partial \boldsymbol{z}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}$
Downstream gradient
Local gradient
$\frac{\partial s}{\partial \boldsymbol{h}}$
Upstream gradient

- 예시) ppt p.57~