Lecture4: Backpropagation and Computation Graphs

목차

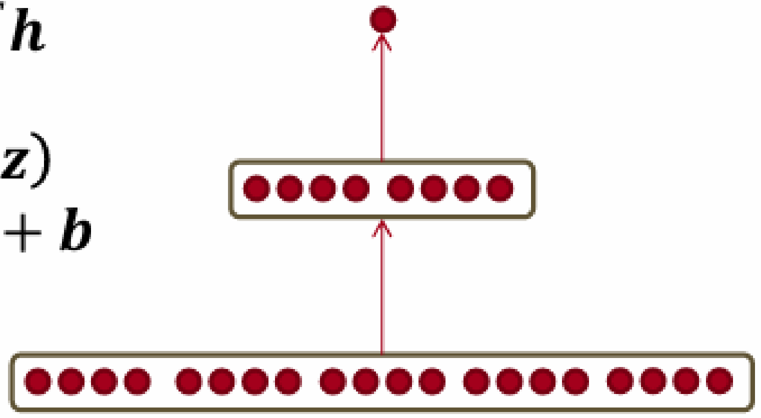
1. Matrix Gradients for Neural Net
2. Computation Graphs and Backpropagation
3. Stuff you should know

1. Matrix Gradients for Neural Net

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

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

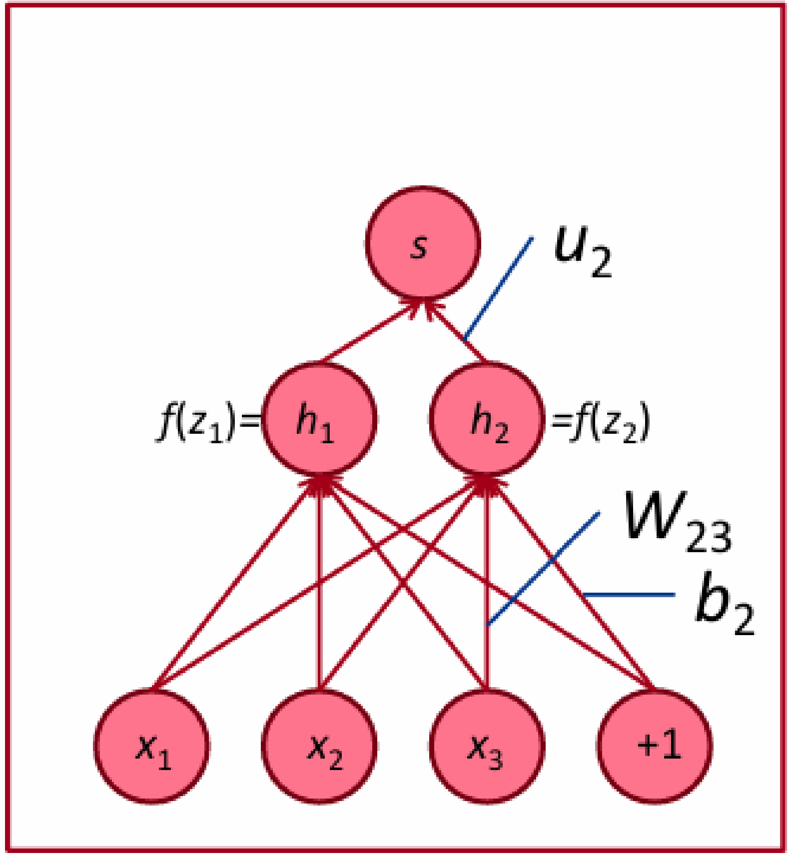
$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$



- Let's look carefully at computing $\frac{\partial s}{\partial \boldsymbol{W}}$
  - Using the chain rule again:

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

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



- Let's consider the derivative of a single weight W_{ij}
- W_{ij} only contributes to z_i
  - For example: W_{23} is only used to compute z_2 not z_1

$$\frac{\partial z_i}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \boldsymbol{W}_{i\cdot}\boldsymbol{x} + b_i$$
$$= \frac{\partial}{\partial W_{ij}} \sum_{k=1}^{d} W_{ik} x_k = x_j$$

- So for derivative of single W_{ij} :

$$\frac{\partial s}{\partial W_{ij}} = \underbrace{\delta_i}_{\substack{\text{Error signal} \\ \text{from above}}} \underbrace{x_j}_{\substack{\text{Local gradient} \\ \text{signal}}}$$

- We want gradient for full ***W*** – but each case is the same
- Overall answer: Outer product:

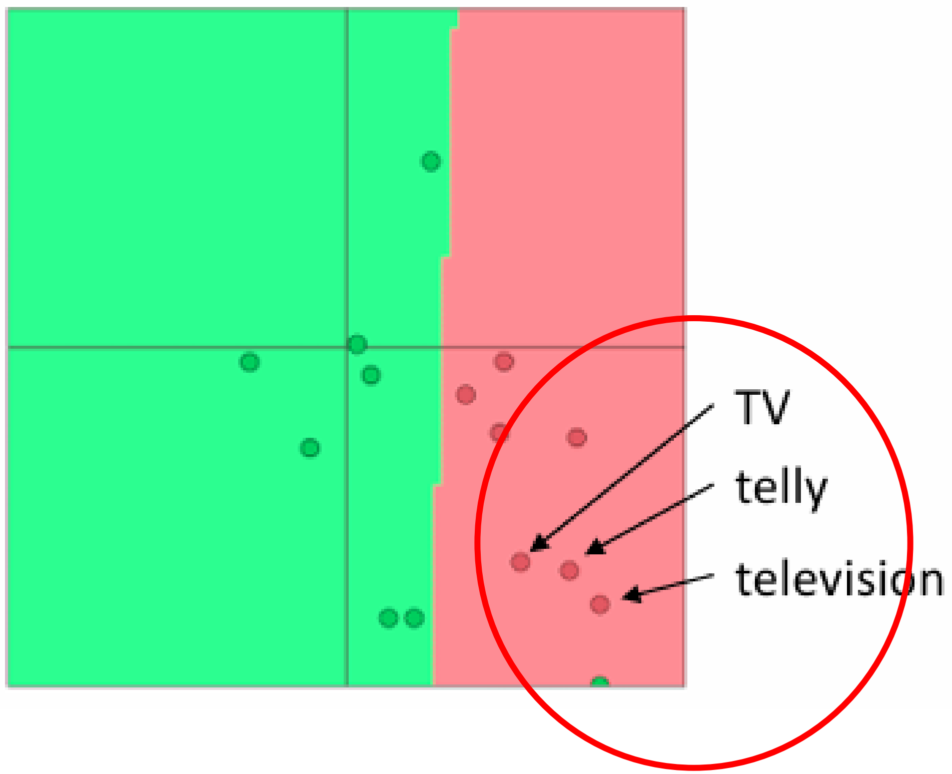
$$\underset{[n \times m]}{\frac{\partial s}{\partial \boldsymbol{W}}} = \underset{[n \times 1]}{\boldsymbol{\delta}^T} \; \underset{[1 \times m]}{\boldsymbol{x}^T}$$

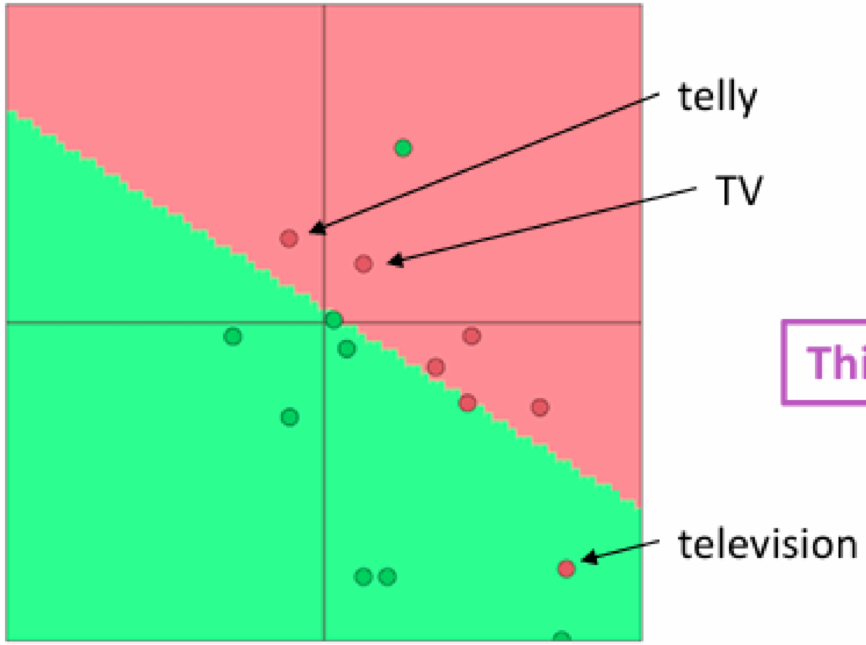
- The gradient that arrives at and updates the word vectors can simply be split up for each word vector:
- Let $\nabla_x J = W^T \delta = \delta_{x_{window}}$
- With x_{window} = [$x_{museums}$ x_{in} x_{Paris} x_{are} $x_{amazing}$]
- We have

$$\delta_{window} = \begin{bmatrix} \nabla_{x_{museums}} \\ \nabla_{x_{in}} \\ \nabla_{x_{Paris}} \\ \nabla_{x_{are}} \\ \nabla_{x_{amazing}} \end{bmatrix} \in \mathbb{R}^{5d}$$

상황: 단일 단어를 사용해서 영화 리뷰의 positive/negative 분류를 수행하는 모델 학습

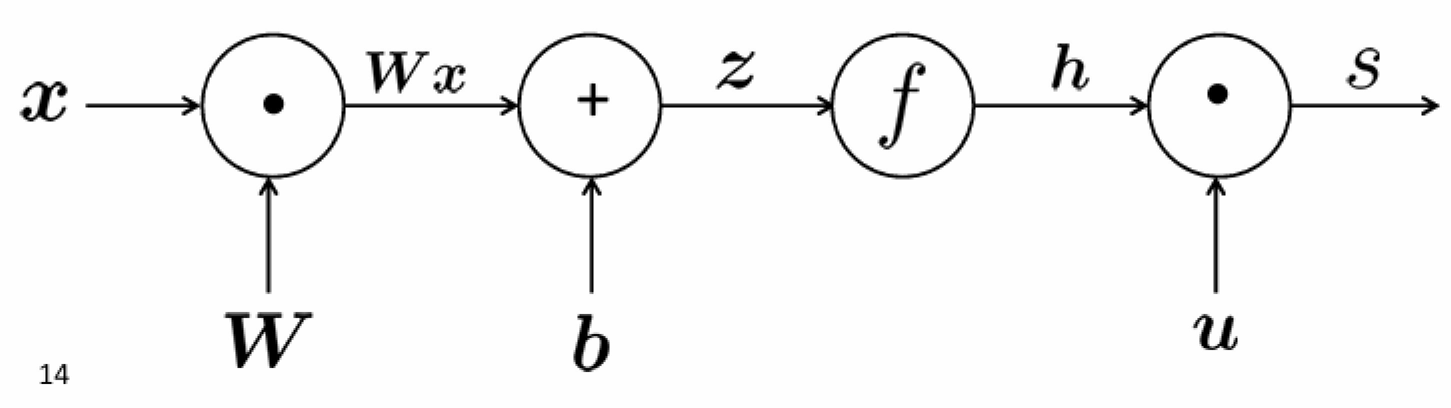
- 학습 데이터: TV, telly
- 테스트 데이터: television



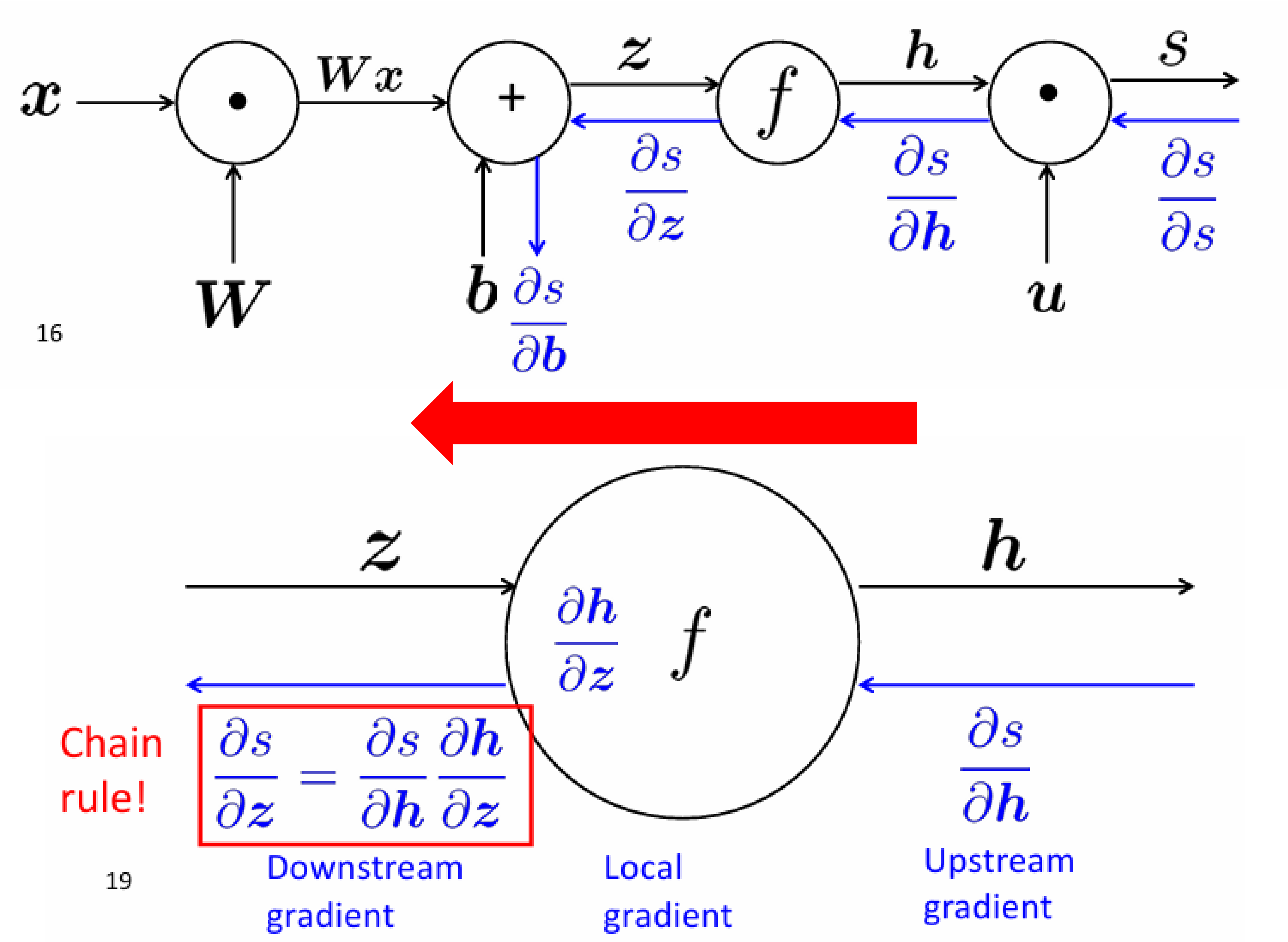


This can be bad!

2. Computation Graphs and Backpropagation

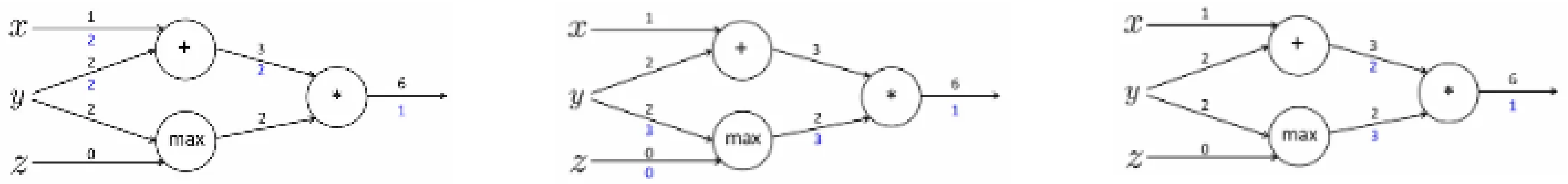


“Forward Propagation”



x
Wx
z
h
s
W
b
u
$\frac{\partial s}{\partial z}$
$\frac{\partial s}{\partial h}$
$\frac{\partial s}{\partial s}$
$\frac{\partial s}{\partial b}$
16
z
h
f
$\frac{\partial h}{\partial z}$
Chain rule!
$\frac{\partial s}{\partial z} = \frac{\partial s}{\partial h} \frac{\partial h}{\partial z}$
$\frac{\partial s}{\partial h}$
19
Downstream gradient
Local gradient
Upstream gradient

3) 연산별 역전파의 특징 이해



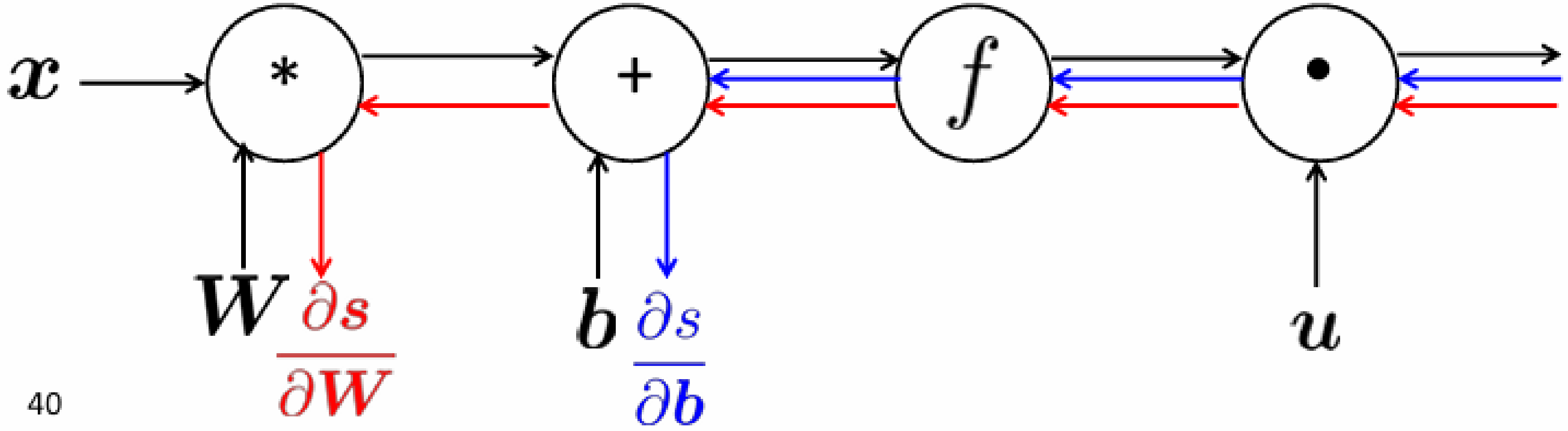
- **더하기 연산**$(+)$의 경우, upstream gradient가 그대로 downstream에 복사된다.
- **최댓값 연산**(max)의 경우, 값이 큰 쪽으로만 gradient를 보내준다(1). 이는 더 큰 변수에 대해서만 max 결과값이 영향을 받기 때문이다.
- **곱하기 연산**$(*)$의 경우, 순방향 전파할 때 온 값을 서로 바꿔서 전달한다.

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

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

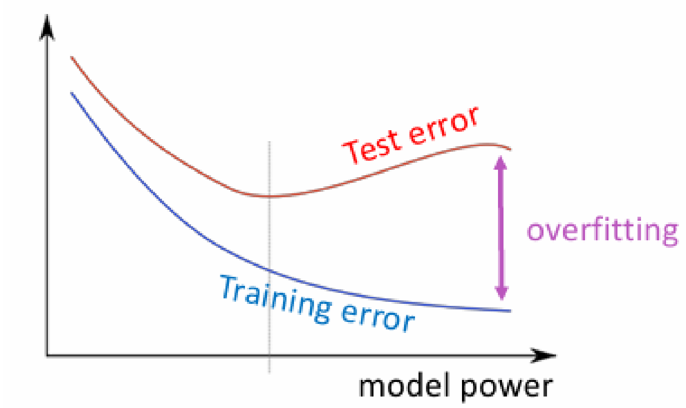
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

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



3. Stuff you should know

1) Regularization



regularization

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{e^{f_{y_i}}}{\sum_{c=1}^{C} e^{f_c}}\right) \boxed{+ \lambda \sum_{k} \theta_k^2}$$

2) Vectorization

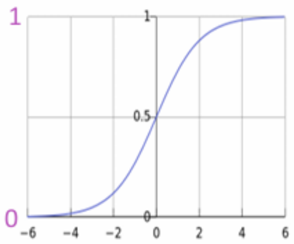
```
from numpy import random
N = 500 # number of windows to classify
d = 300 # dimensionality of each window
C = 5 # number of classes
W = random.rand(C,d)
wordvectors_list = [random.rand(d,1) for i in range(N)]
wordvectors_one_matrix = random.rand(d,N)

%timeit [W.dot(wordvectors_list[i]) for i in range(N)]
%timeit W.dot(wordvectors_one_matrix)
```

3. Stuff you should know

3) Non-linearities

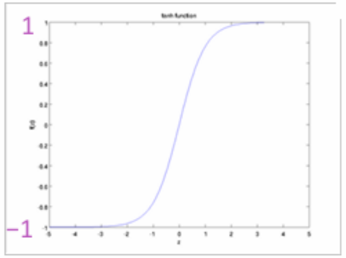
logistic (“sigmoid”)

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



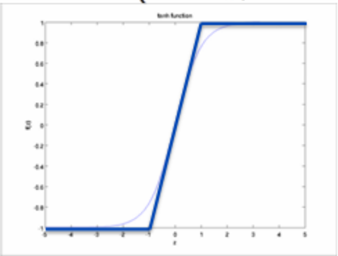
tanh

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



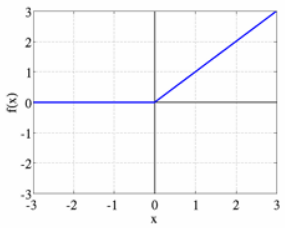
hard tanh

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



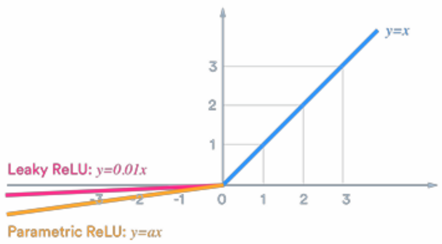
ReLU (rectified linear unit) hard tanh

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



Leaky ReLU

Parametric ReLU



4) Initialization

5) Optimization

6) Learning Rate