Week 2 CS224n NLP with Deeping Learning

Lecture 3 : Neural Networks

softmax classifier: $P(y|x) = \frac{\exp(W_y \cdot x)}{\sum_{c=1}^{C} \exp(W_c \cdot x)}$

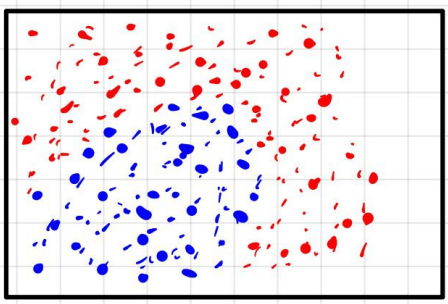
- take y^{th} row $W_y \cdot x = \sum_{i=1}^{d} W_{yi} x_i = f_y$
- objective : maximize the probability of the correct class y
  $\Longleftrightarrow$ minimize negative log function
  $-\log P(y|x) = -\log\left(\frac{\exp(f_y)}{\sum_{c=1}^{C} \exp(f_c)}\right)$
- Alternative : cross-entropy loss

$$H(p,q) = -\sum_{c=1}^{C} p(c) \log q(c)$$

Assume the ground truth probability is 1 at the correct class,
the only term left is the negative log probability of the true class.

Neural Network Classifier

- notice basic classifier only gives linear boundary



← Linear classifier won't work

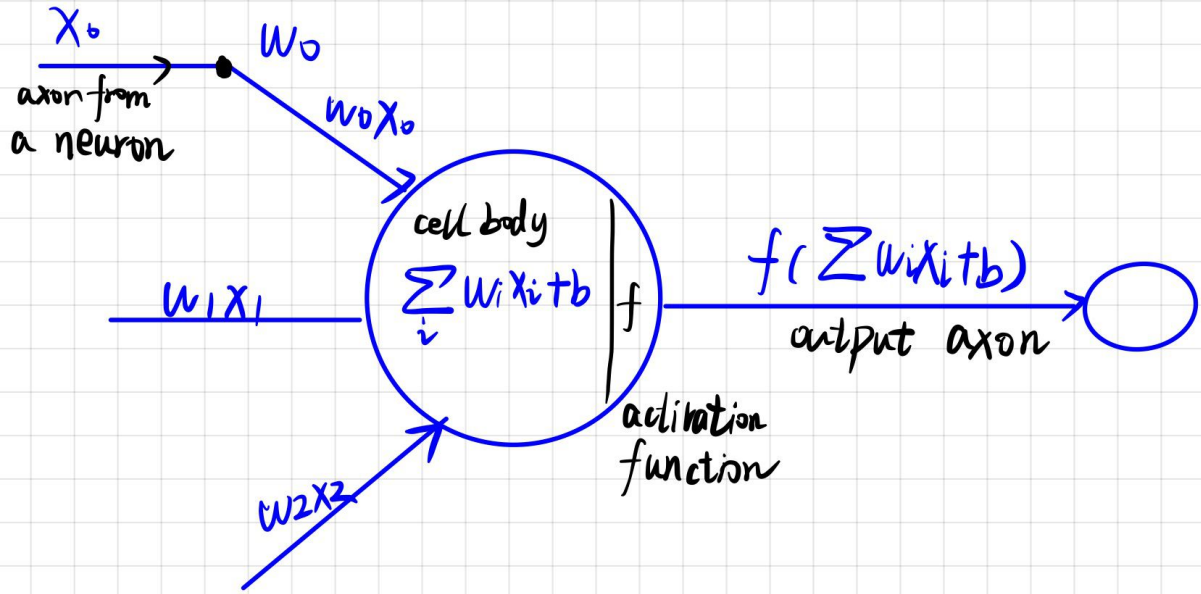
- neural network can learn much more complex patterns and nonlinear decision boundary

- Common in NLP deep learning :
  - learn both W and word vector x
  - learn both parameters and representations
  - The word vectors re-represent on-hot encoding — move them around in an intermediate layer vector space

$$\nabla_\theta J(\theta) = \begin{bmatrix} \nabla_{W_1} \\ \vdots \\ \nabla_{W_d} \\ \nabla_{x_{\text{aardvark}}} \\ \vdots \\ \nabla_{x_{\text{zebra}}} \end{bmatrix} \begin{matrix} \} C \\ \} V \end{matrix} \in \mathbb{R}^{Cd + Vd}$$

very large number of parameters

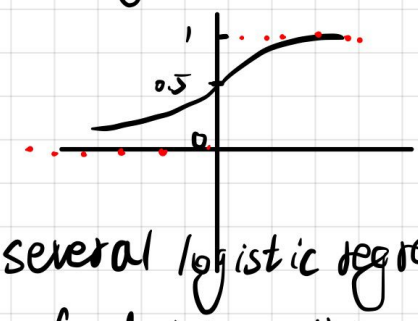
an artificial neuron



a neuron can be a binary logistic regression unit

$$h_{w,b}(x) = f(w^T x + b)$$

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



a neural network = running several logistic regressions at the same time.

.... which we can feed into another logistic regression

Matrix notation: $\begin{cases} z = Wx + b \\ a = f(z) \end{cases}$

activation f is applied element-wise: $f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$

✡ non-linearity is most important for deep neural network

(axiom: affine transformation's composition is still affine)

4. Named Entity Recognition (NER)

- Task: find and classify names in Texts

why difficult:
- boundary of entity
- if something is an entity
- hard to know class of novel entity
- entity depends on context

- Binary word window classification

Idea: classify a word in its context window

⚠: unnormalized scores

use neural activation "a" simple to give unnormalized score

Score $(x) = U^T a \in \mathbb{R}$

add extra lager for non-linearity information.

How to compute $\nabla_\theta J(\theta)$? By hand or by back propogation

Gradients: 1. $f(x) = x^3$, $\frac{df}{dx} = 3x^2$

2. $f(\mathbf{x}) = f(x_1, x_2, \dots, x_n)$, $\frac{\partial f}{\partial \mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right]$

3. $f(x) = [f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)]$

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

4. Chain Rule:

for multiple variables at once: multiply Jacobians

$h = f(z)$

$z = Wx + b$

$\frac{\partial h}{\partial x} = \frac{\partial h}{\partial z} \cdot \frac{\partial z}{\partial x} = \dots$

5. $h = f(z)$, element-wise activation function f

$h_i = f(z_i)$

What is $\frac{\partial h}{\partial z}$?

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

$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$

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

other Jacobian

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

$$\frac{\partial}{\partial b}(Wx+b) = I \quad (\text{identity matrix})$$

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

Back to neural Network:

$s = u^T h$

$h = f(Wx+b)$

x (input)

let's find $\frac{\partial s}{\partial b}$

$$\Longrightarrow \begin{cases} s = u^T h \\ h = f(z) \\ z = Wx+b \\ x \text{ (input)} \end{cases}$$

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

$$= u^T \cdot f'(z) \cdot I$$

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

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

may use "local error signal". δ

Instead of following convention: shape of the gradient is the shape of the parameters

7 Matrix Calculus Identity

(1) Matrix times column vector with respect to the column vector

$z = Wx$, then $\frac{\partial z}{\partial x} = W$

(2) Row vector times matrix with respect to the row vector

$z = xW$, then $\frac{\partial z}{\partial x} = W^T$

(3) A vector with itself

$z = x$, then $\frac{\partial z}{\partial x} = I$

(4) An element-wise function applied to a vector

$z_i = f(x_i)$ $z = f(x)$, then $\frac{\partial z}{\partial x} = \text{diag}(f'(x))$

(5) Matrix times column vector with respect to the matrix

$z = Wx$, $\delta = \frac{\partial J}{\partial z}$, what is $\frac{\partial J}{\partial W} = \frac{\partial J}{\partial z}\frac{\partial z}{\partial W} = \delta\frac{\partial z}{\partial W}$

for keep the shape, $\frac{\partial J}{\partial W} = \delta^T x^T$

(6) Row vector times matrix with respect to the matrix

$z = xW$, $\delta = \frac{\partial J}{\partial z}$, what is $\frac{\partial J}{\partial W} = \delta\frac{\partial z}{\partial W}$

for keep the shape, $\frac{\partial J}{\partial W} = x^T\delta$

(7) Cross-entropy loss with respect to logits

$\hat{y} = \text{softmax}(\theta)$, $J = CE(\hat{y}, y)$

$\frac{\partial J}{\partial \theta} = \hat{y} - y$

Lecture 4: Backpropagation

- The gradient that arrives at and updates the word vectors can be simplit up for each word vector

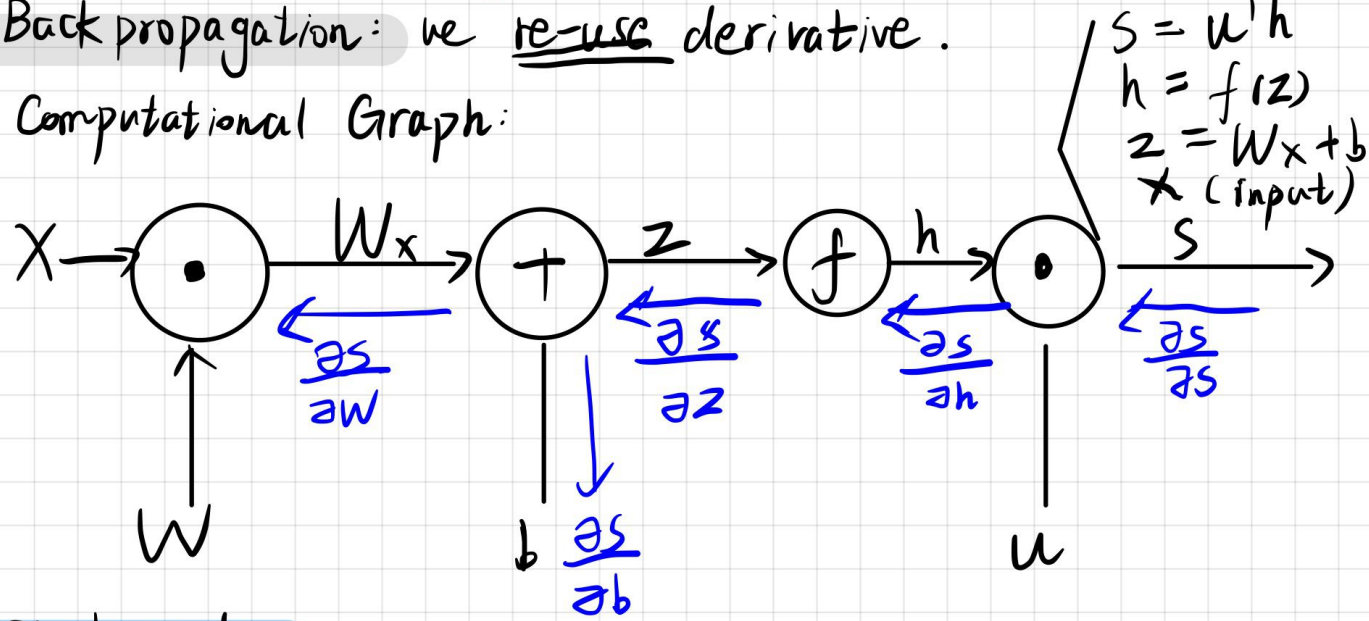
with $X_{window} = [X_{museum} \quad X_{in} \quad X_{paris} \quad X_{are} \quad X_{amazing}]$

we have $\nabla_{window} = \begin{bmatrix} \nabla X_{mesuesm} \\ \nabla X_{in} \\ \nabla X_{pais} \\ \nabla X_{are} \\ \nabla X_{amazing} \end{bmatrix} \in \mathbb{R}^{5d}$

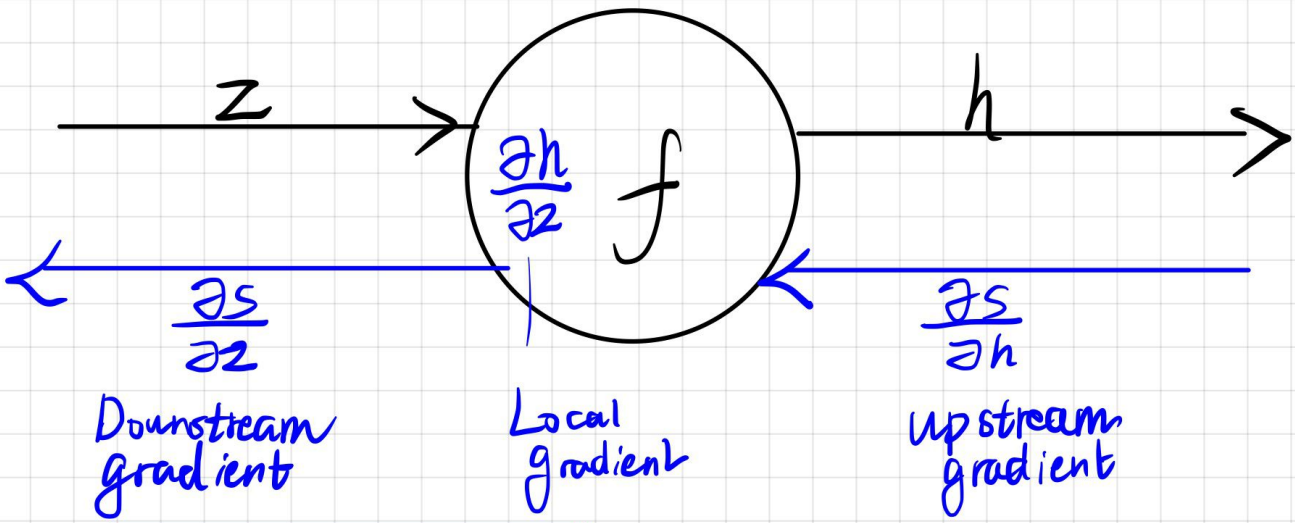
⟹ This push word vectors around, might be helpful in determing named entity recognition.

⟹ pitfall: similar words may not be moved together if not appear in training data.

- Backpropagation: we re-use derivative.
- Computational Graph:



Single node

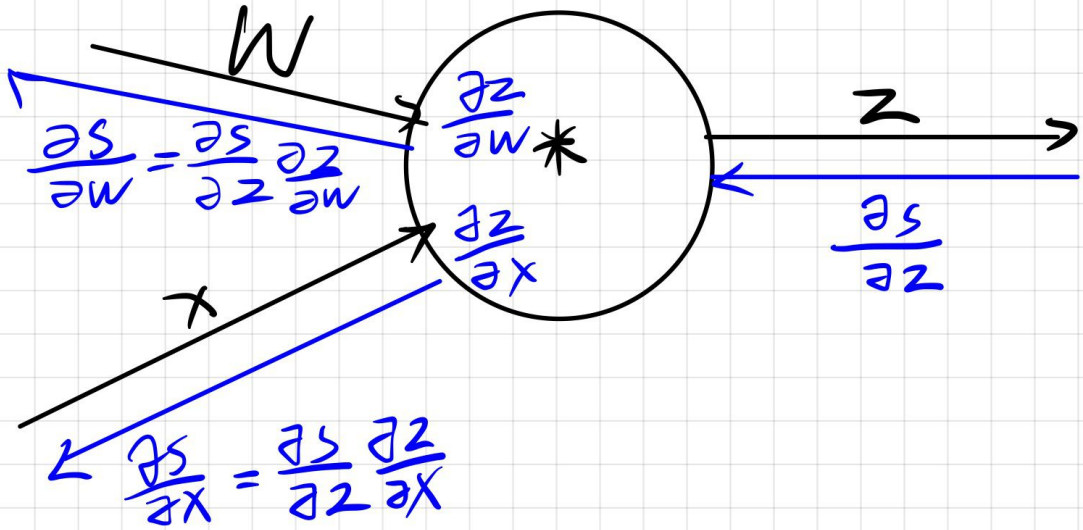


$$\frac{\partial s}{\partial z} = \frac{\partial s}{\partial h}\frac{\partial h}{\partial z}$$ Chain rule

[downstream gradient] = [upstream gradient] × [Local gradient]

multiple inputs

$$z = Wx$$



example:

Forward prop steps

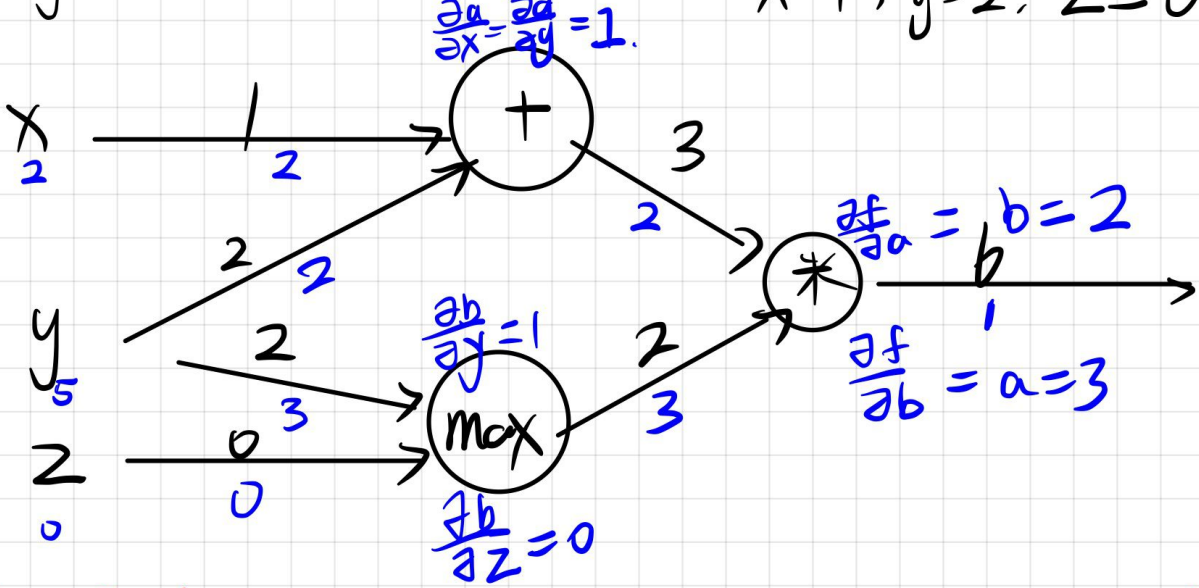
$a = x + y$

$b = \max(y, z)$

$f = ab$

$f(x,y,z) = (x+y)\max(y,z)$

$x=1, y=2, z=0$



*: Gradients sum at outward branches (加法，非乘法)

"+" distribute

"*" switch — the upstream gradient

"max" route

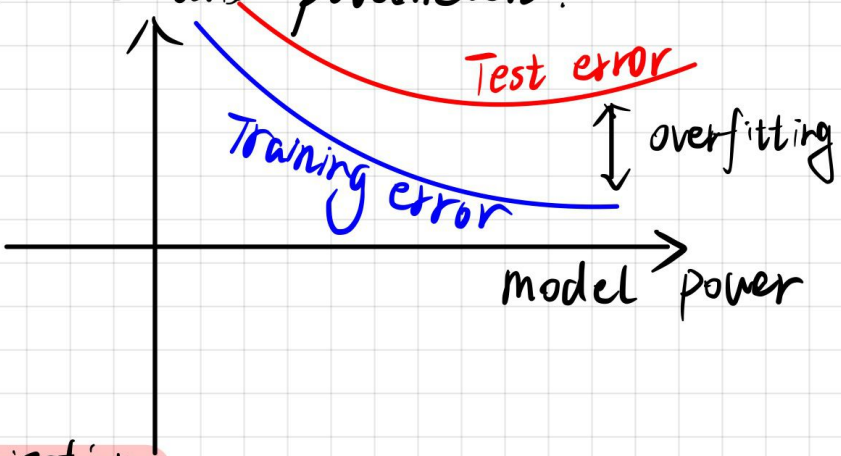
correct way: avoid repeated computation.

compute all gradient (local) at once and save it for later usage.

Gradient checking: $f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$, for $h \approx 1e-4$

- easy to implement correctly
- But approximate and very **slow**
  - have to recompute f for **every parameter**

- **Regularization**
  - In practice loss function usually includes **regularization** over all parameter θ, e.g. L2-regularization
  - Regularization (largely) prevents overfitting when have a lot of features and parameters.

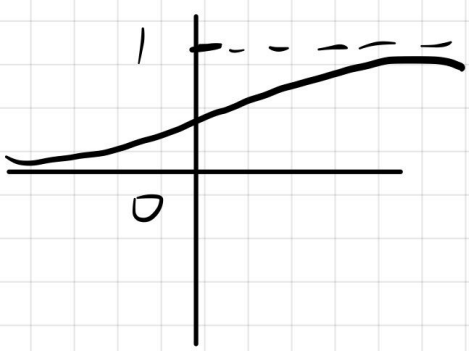


- **Vectorization**

- **Non-Linearity**

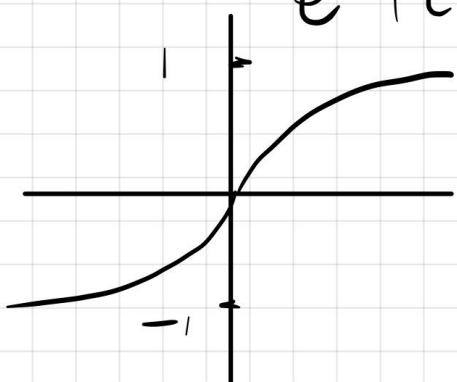
logistic ("sigmoid")

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

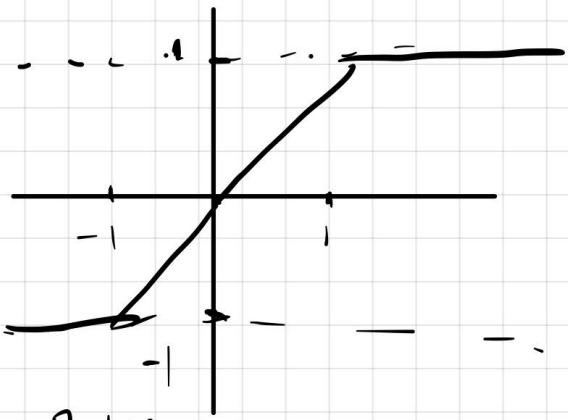


tanh

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

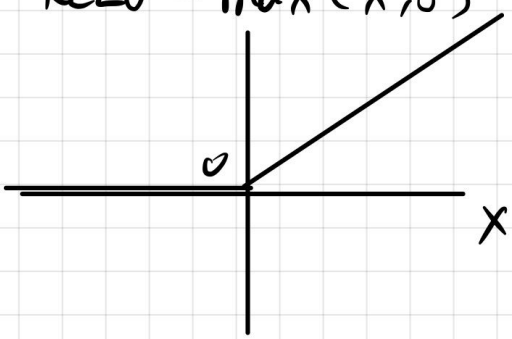


hard tanh



$$\text{hard tanh} = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } [-1, 1] \\ 1 & \text{if } x > 1 \end{cases}$$

ReLU = max(x, 0)



- Parameter Initialization
  - must initialize weights to small values
    - to avoid symmetries that prevent learning
  - Xavier

$$Var(W_i) = \frac{2}{n_{in} + n_{out}}$$