



NUS

National University
of Singapore

| **Computing**

Eric Han

eric_han@nus.edu.sg

<https://eric-han.com>

Computer Science

T08 – 29 Oct 2024

Week 11

CS2109s Makeup Wed 1-2pm, 7-8pm

- 1 Backpropagation
- 2 Backpropagation for a Deep(er) Network
- 3 Potential Issues with Training Deep Neural Networks
- 4 Dying ReLU Problem



Section 1: **Backpropagation**

$$f^{[1]} = W^{[1]T}X, \quad \hat{Y} = g^{[1]}(f^{[1]}), \quad \mathcal{E} = -\frac{1}{n} \sum_{i=0}^{n-1} \left\{ [Y_{0i} \cdot \log(\hat{Y}_{0i})] + [(1 - Y_{0i}) \log(1 - \hat{Y}_{0i})] \right\}$$

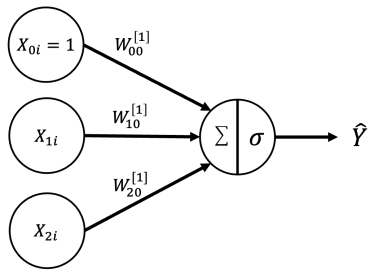


Figure 1: Simple Neural Network

Question [G]

When $n = 1$:

a. $\frac{\partial \mathcal{E}}{\partial \hat{Y}} = \left[-\frac{Y_{00}}{\hat{Y}_{00}} + \frac{1-Y_{00}}{1-\hat{Y}_{00}} \right] \text{ (Given)}$

b. $\frac{\partial \mathcal{E}}{\partial f^{[1]}} = \hat{Y} - Y$

c. $\frac{\partial \mathcal{E}}{\partial W_{20}^{[1]}} = \left(\frac{\partial \mathcal{E}}{\partial f^{[1]}} \right)_{00} X_{20}$

Recap

- › What is back propagation?
- › How to perform forward propagation?
- › How to perform back propagation?

Answer b

Since $n = 1$, $\frac{\partial \mathcal{E}}{\partial f^{[1]}} = \frac{\partial \mathcal{E}}{\partial f_{00}^{[1]}} = \frac{\partial \mathcal{E}}{\partial \hat{Y}_{00}} \frac{\partial \hat{Y}_{00}}{\partial f_{00}^{[1]}}$ (chain rule)

Since $\hat{Y}_{00} = \sigma(f_{00}^{[1]}) \implies \frac{\partial \hat{Y}_{00}}{\partial f_{00}^{[1]}} = \sigma(f_{00}^{[1]}) (1 - \sigma(f_{00}^{[1]})) = \hat{Y}_{00}(1 - \hat{Y}_{00})$

From (i), $\frac{\partial \mathcal{E}}{\partial \hat{Y}_{00}} = \left[-\frac{Y_{00}}{\hat{Y}_{00}} + \frac{1-Y_{00}}{1-\hat{Y}_{00}} \right]$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial f^{[1]}} &= \left[\frac{\partial \mathcal{E}}{\partial \hat{Y}_{00}} \frac{\partial \hat{Y}_{00}}{\partial f_{00}^{[1]}} \right] \\&= \left[-\frac{Y_{00}}{\hat{Y}_{00}} + \frac{1-Y_{00}}{1-\hat{Y}_{00}} \right] [\hat{Y}_{00}(1 - \hat{Y}_{00})] \\&= \left[-Y_{00}(1 - \hat{Y}_{00}) + (1 - Y_{00})\hat{Y}_{00} \right] \\&= [\hat{Y}_{00} - Y_{00}] \\&= \hat{Y} - Y.\end{aligned}$$

Answer c

Since $n = 1$, $\frac{\partial \mathcal{E}}{\partial W_{20}^{[1]}} = \frac{\partial \mathcal{E}}{\partial f_{00}^{[1]}} \frac{\partial f_{00}^{[1]}}{\partial W_{20}^{[1]}}$ (chain rule)

$$f_{00}^{[1]} = W^{[1]T} X = \sum_{i=0}^2 (W^{[1]T})_{0i} X_{i0} = \sum_{i=0}^2 W_{i0}^{[1]} X_{i0} \implies \frac{\partial f_{00}^{[1]}}{\partial W_{20}^{[1]}} = X_{20}$$

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial W_{20}^{[1]}} &= \frac{\partial \mathcal{E}}{\partial f_{00}^{[1]}} \frac{\partial f_{00}^{[1]}}{\partial W_{20}^{[1]}} \\ &= \frac{\partial \mathcal{E}}{\partial f_{00}^{[1]}} X_{20} \\ &= \left(\frac{\partial \mathcal{E}}{\partial f^{[1]}} \right)_{00} X_{20} \end{aligned}$$

Note: $\frac{\partial \mathcal{E}}{\partial \hat{Y}}$, and $\frac{\partial \mathcal{E}}{\partial f^{[1]}}$ are matrices since \mathcal{E} is a scalar, but \hat{Y} and $f^{[1]}$ are matrices. However, $\frac{\partial \mathcal{E}}{\partial W_{20}^{[1]}}$ is a scalar since $W_{20}^{[1]}$ is a scalar.

Question 2-4 [G]

- 2 Derive an expression for $\frac{\partial \mathcal{E}}{\partial W^{[1]}}$, how does back propagation work?
- 3 Let us consider a general case where $n \in \mathbb{N}$, find $\frac{\partial \mathcal{E}}{\partial f^{[1]}}$.
- 4 Why do the hyper-parameters α and β ? How to set their values?

$$\mathcal{E} = -\frac{1}{n} \sum_{i=0}^{n-1} \left\{ \alpha [Y_{0i} \cdot \log(\hat{Y}_{0i})] + \beta [(1 - Y_{0i}) \cdot \log(1 - \hat{Y}_{0i})] \right\}$$

Answer 2

From (1c), the general form is $\frac{\partial \mathcal{E}}{\partial W_{i0}^{[1]}} = \left(\frac{\partial \mathcal{E}}{\partial f^{[1]}} \right)_{00} X_{i0}$

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial W^{[1]}} &= \left[\frac{\partial \mathcal{E}}{\partial W_{00}^{[1]}}, \frac{\partial \mathcal{E}}{\partial W_{10}^{[1]}}, \frac{\partial \mathcal{E}}{\partial W_{20}^{[1]}} \right]^T = \left(\frac{\partial \mathcal{E}}{\partial f^{[1]}} \right)_{00} [X_{00}, X_{10}, X_{20}]^T = \left(\frac{\partial \mathcal{E}}{\partial f^{[1]}} \right)_{00} X \\ &= \left(\hat{Y} - Y \right)_{00} X = \left(g^{[1]}(f^{[1]}) - Y \right)_{00} X = \left(g^{[1]}(W^{[1]T} X) - Y \right)_{00} X \end{aligned}$$

Intuition behind back propagation $W^{[1]} = W^{[1]} - \alpha \frac{\partial \mathcal{E}}{\partial W^{[1]}}$:

- Change in first layer weighted sum $f^{[1]}$
- Change in predicted value \hat{Y}
- Change of loss \mathcal{E}
- Decrease the loss by changing the weights

Directly proportional to input X

Answer 3

From (1), $\frac{\partial \mathcal{E}}{\partial f_{0i}^{[1]}} = \left[\frac{\partial \mathcal{E}}{\partial \hat{Y}_{0i}} \frac{\partial \hat{Y}_{0i}}{\partial f_{0i}^{[1]}} \right]$

$$\frac{\partial \mathcal{E}}{\partial \hat{Y}} = \left[\frac{\partial \mathcal{E}}{\partial \hat{Y}_{00}}, \frac{\partial \mathcal{E}}{\partial \hat{Y}_{01}}, \dots, \frac{\partial \mathcal{E}}{\partial \hat{Y}_{0n}} \right] = \left[\dots, \frac{1}{n} \left(-\frac{Y_{0i}}{\hat{Y}_{0i}} + \frac{1 - Y_{0i}}{1 - \hat{Y}_{0i}} \right), \dots \right]$$

$$\frac{\partial \hat{Y}_{0i}}{\partial f_{0i}^{[1]}} = \sigma(f_{0i}^{[1]}) \left(1 - \sigma(f_{0i}^{[1]}) \right) = \hat{Y}_{0i} (1 - \hat{Y}_{0i})$$

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial f^{[1]}} &= \left[\frac{\partial \mathcal{E}}{\partial f_{00}^{[1]}}, \frac{\partial \mathcal{E}}{\partial f_{01}^{[1]}}, \dots, \frac{\partial \mathcal{E}}{\partial f_{0n}^{[1]}} \right] \\ &= \frac{1}{n} \left[(\hat{Y}_{00} - Y_{00}), (\hat{Y}_{01} - Y_{01}), \dots, (\hat{Y}_{0n} - Y_{0n}) \right] \\ &= \frac{1}{n} (\hat{Y} - Y) \end{aligned}$$

Answer 4

Weighted Error:

$$\mathcal{E} = -\frac{1}{n} \sum_{i=0}^{n-1} \left\{ \alpha [Y_{0i} \cdot \log(\hat{Y}_{0i})] + \beta [(1 - Y_{0i}) \cdot \log(1 - \hat{Y}_{0i})] \right\}$$


Apply a weight to how much each class contributes to the loss function:

- Error due to Cultiva A ($p_A = 100/1100$): $Y_{0i} \cdot \log(\hat{Y}_{0i})$
- Error due to Cultiva B ($p_B = 1000/1100$): $(1 - Y_{0i}) \cdot \log(1 - \hat{Y}_{0i})$

Since we have unbalanced dataset, we can weight using the ratio $\frac{\alpha}{\beta} = \frac{1/100}{1/1000}$:

- $\alpha = 1/100$
- $\beta = 1/1000$

We punish the model more heavily if it misclassifies A, so the model won't be biased towards predicting all samples as B.



Section 2: **Backpropagation for a Deep(er) Network**

When $n = 1$, compute $\frac{\partial \mathcal{E}}{\partial W_{11}^{[1]}}$, where

$$f^{[1]} = W^{[1]T} X, \quad a^{[1]} = g^{[1]}(f^{[1]}), \quad f^{[2]} = W^{[2]T} a^{[1]}, \quad \hat{Y} = g^{[2]}(f^{[2]}), \quad g^{[1]}(s) = \text{ReLU}(s), \quad g^{[2]}(s) = \sigma(s) = \frac{1}{1+e^{-s}}, \quad W^{[1]} \in \mathbb{R}^{3 \times 2}, \quad W^{[2]} \in \mathbb{R}^{2 \times 1}.$$

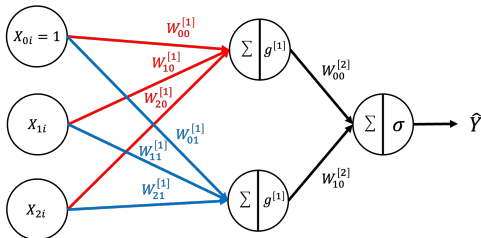


Figure 2: Complex NN

[@] Is ReLU continuous/discontinuous, not/differentiable; Can we use discontinuous activation functions?

When $n = 1$, compute $\frac{\partial \mathcal{E}}{\partial W_{11}^{[1]}}$, where

$$f^{[1]} = W^{[1]T} X, \quad a^{[1]} = g^{[1]}(f^{[1]}), \quad f^{[2]} = W^{[2]T} a^{[1]}, \quad \hat{Y} = g^{[2]}(f^{[2]}), \quad g^{[1]}(s) = \text{ReLU}(s), \quad g^{[2]}(s) = \sigma(s) = \frac{1}{1+e^{-s}}, \quad W^{[1]} \in \mathbb{R}^{3 \times 2}, \quad W^{[2]} \in \mathbb{R}^{2 \times 1}.$$

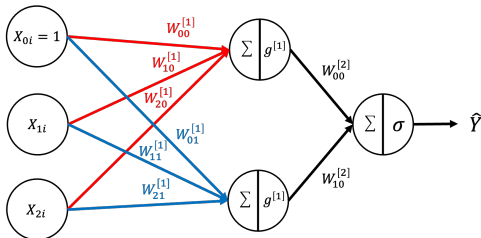


Figure 2: Complex NN

[@] Is ReLU continuous/discontinuous, not/differentiable; Can we use discontinuous activation functions? Continuous but not differentiable.

Answer

Intuition: Plot the forward path and take the derivative.

$$W_{11}^{[1]} \xrightarrow{f^{[1]}=W^{[1]T}X} f_{10}^{[1]} \xrightarrow{a^{[1]}=g^{[1]}(f^{[1]})} a_{10}^{[1]} \xrightarrow{f^{[2]}=W^{[2]T}a^{[1]}} f_{00}^{[2]} \xrightarrow{\hat{Y}=g^{[2]}(f^{[2]})} \hat{Y}_{00} \rightarrow \mathcal{E}$$

$$f^{[1]} = \begin{bmatrix} W_{00}^{[1]} & W_{01}^{[1]} \\ W_{10}^{[1]} & W_{11}^{[1]} \\ W_{20}^{[1]} & W_{21}^{[1]} \end{bmatrix}^T \begin{bmatrix} X_{00} \\ X_{10} \\ X_{20} \end{bmatrix} = \begin{bmatrix} \sum_i W_{i0}^{[1]} X_{i0} \\ \sum_i W_{i1}^{[1]} X_{i0} \end{bmatrix}$$

$$f^{[2]} = \begin{bmatrix} W_{00}^{[2]} \\ W_{10}^{[2]} \end{bmatrix}^T \begin{bmatrix} a_{00}^{[1]} \\ a_{10}^{[1]} \end{bmatrix} = \left[\sum_i W_{i0}^{[2]} a_{i0}^{[1]} \right]$$

Expand using chain rule: $\frac{\partial \mathcal{E}}{\partial W_{11}^{[1]}} = \frac{\partial \mathcal{E}}{\partial \hat{Y}_{00}} \frac{\partial \hat{Y}_{00}}{\partial f_{00}^{[2]}} \frac{\partial f_{00}^{[2]}}{\partial a_{10}^{[1]}} \frac{\partial a_{10}^{[1]}}{\partial f_{10}^{[1]}} \frac{\partial f_{10}^{[1]}}{\partial W_{11}^{[1]}}$

Find each of the terms in $\frac{\partial \mathcal{E}}{\partial W_{11}^{[1]}} = \frac{\partial \mathcal{E}}{\partial \hat{Y}_{00}} \frac{\partial \hat{Y}_{00}}{\partial f_{00}^{[2]}} \frac{\partial f_{00}^{[2]}}{\partial a_{10}^{[1]}} \frac{\partial a_{10}^{[1]}}{\partial f_{10}^{[1]}} \frac{\partial f_{10}^{[1]}}{\partial W_{11}^{[1]}}$:

- $\frac{\partial \mathcal{E}}{\partial \hat{Y}_{00}} = -\frac{\alpha Y_{00}}{\hat{Y}_{00}} + \frac{\beta(1-Y_{00})}{1-\hat{Y}_{00}}$
- $\frac{\partial \hat{Y}_{00}}{\partial f_{00}^{[2]}} = \sigma(f_{00}^{[2]}) \left(1 - \sigma(f_{00}^{[2]})\right)$
- $\frac{\partial f_{00}^{[2]}}{\partial a_{10}^{[1]}} = W_{10}^{[2]}$
- $\frac{\partial a_{10}^{[1]}}{\partial f_{10}^{[1]}} = \begin{cases} 0, \text{ if } f_{10}^{[1]} \leq 0 \\ 1, \text{ otherwise} \end{cases} = \mathbb{1}_{f_{10}^{[1]} > 0}$
- $\frac{\partial f_{10}^{[1]}}{\partial W_{11}^{[1]}} = X_{10}$

where $\mathbb{1}_{f_{10}^{[1]} > 0}$ is an indicator function. Therefore,

$$\frac{\partial \mathcal{E}}{\partial W_{11}^{[1]}} = \left[-\frac{\alpha Y_{00}}{\hat{Y}_{00}} + \frac{\beta(1-Y_{00})}{1-\hat{Y}_{00}} \right] \sigma(f_{00}^{[2]}) \left(1 - \sigma(f_{00}^{[2]})\right) W_{10}^{[2]} \mathbb{1}_{f_{10}^{[1]} > 0} X_{10}$$

A background pattern of thin, light gray lines forming a complex, interconnected geometric mesh of triangles and polygons, resembling a network or a crystalline structure.

Section 3: **Potential Issues with Training Deep Neural Networks**

Play around with the code given in the `ipynb`, layer 0 gradient is $< 10^{-40}$

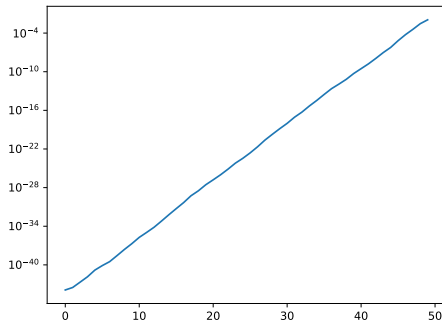


Figure 3: Layers / Max Abs Gradient, using sigmoid

- 1 Gradient magnitudes of the first few layers are extremely small, what's the problem?
- 2 Based on what we have learnt thus far, how can we mitigate this problem?
 - » [C] Other sophisticated ways to resolve the issue, and why does it work?

Answer 1

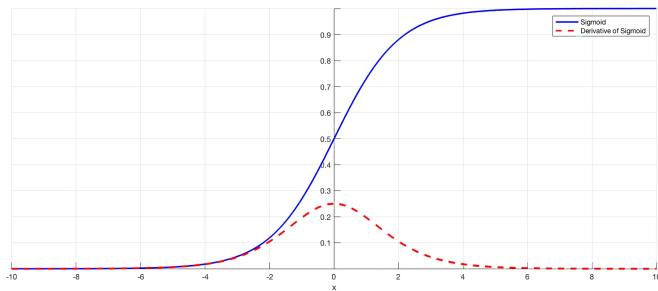


Figure 4: Sigmoid Function

Vanishing gradient problem: Earlier weights need more terms for updates.

- we need to take product of many, many derivatives
- derivatives of sigmoid is in $(0, 1/4]$
- ending up with a really small number
- causing convergence to be slow.

Answer 2

Derivative of ReLU (continuous but not differentiable at $x = 0$ – usually defined as 1):

$$\text{ReLU}(x) = \max(0, x), \quad \frac{\partial \text{ReLU}(x)}{\partial x} = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x > 0 \end{cases}$$

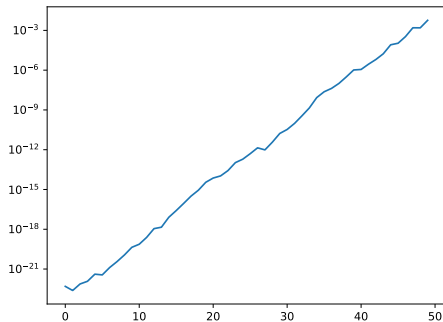


Figure 5: Layers / Max Abs Gradient, using ReLU

A background pattern of thin, light gray lines forming a complex, interconnected geometric mesh of triangles and polygons, resembling a wireframe or a low-poly mesh, covering the top half of the slide.

Section 4: **Dying ReLU Problem**

Dying ReLU Problem - majority of the activations are 0 (meaning the underlying pre-activations are mostly negative), resulting in the network dying midway.

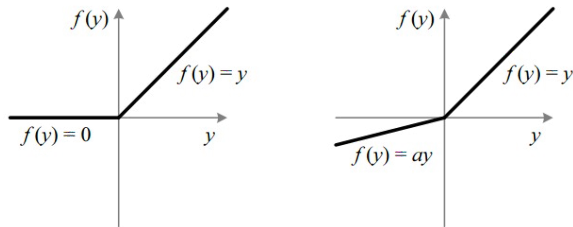


Figure 6: The Rectified Linear Unit (ReLU) (left) vs The Leaky Rectified Linear Unit (Leaky ReLU) with a as the slope when the values are negative. (right)

- How does Leaky ReLU fix this? What happens if we set $a = 1$ in the Leaky ReLU?

$$\text{ReLU}(x) = \max(0, x), \quad \frac{\partial \text{ReLU}(x)}{\partial x} = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x > 0 \end{cases}$$

- ReLU being stuck at 0 because the gradient is 0.
- Leaky ReLU get around this by creating small positive gradient a
- When $a = 1$, the activation function becomes a linear function (NN loses power)¹.

¹Last week tutorial

Investigate for exploding gradient as per the question about vanishing gradient, use the code given for tutorial as a starting point.

Tasks

- 1 Implement a neural network that exhibits exploding gradient.
- 2 Plot the magnitudes for all layers like done in vanishing gradient.
- 3 Analyse ways to mitigate the issue.



Figure 7: Attendance: <https://forms.gle/FDuQWNu52zwWfGRv9>

