National University of Singapore
School of Computing
CS2109S: Introduction to AI and Machine Learning
Semester II, 2024/2025

Tutorial 9
Backpropagation

Summary of Key Concepts

In this tutorial, we will discuss and explore the following learning points from Lecture:

1. Backpropagation
2. Matrix calculus for backpropagation
3. Potential issues with backpropagation
4. Dying ReLU problem

A Backpropagation (Warm Up)

Grace has a wine dataset which comprises of 1100 samples. Each wine sample has a label indicating which plant variety it comes from, and two features: colour intensity and alcohol level.

Now, she wants to build a classifier that can predict which plant variety a wine sample comes from using the two features. She decided to train a neural network with the architecture shown in the figure below.

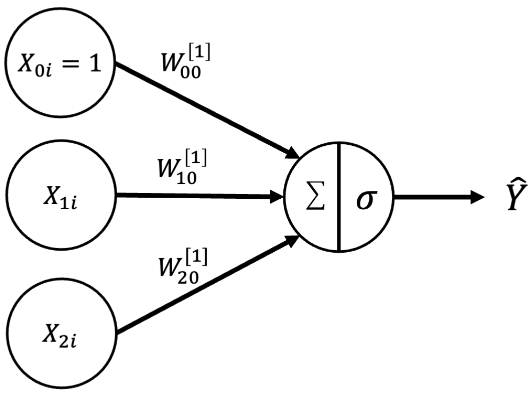

Figure 1: neural network architecture for the i-th data point (out of 1100 samples). $X_{0i} = 1$ represents its bias term, and X_{1i}, X_{2i} represent its 1st and 2nd features, i.e., the color intensity and alcohol level, respectively.

Mathematically, this is given by:

$$f^{[1]} = W^{[1]^T} X$$
$$\hat{Y} = g^{[1]}(f^{[1]})$$

where

the weight matrix $W^{[1]} = \begin{bmatrix} W_{00}^{[1]} \\ W_{10}^{[1]} \\ W_{20}^{[1]} \end{bmatrix} \in \mathbb{R}^{3\times 1}$, input data matrix $X = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \end{bmatrix} \in \mathbb{R}^{3\times n}$,

weighted sum matrix $f^{[1]} = \begin{bmatrix} f_1^{[1]} & f_2^{[1]} & \cdots & f_n^{[1]} \end{bmatrix} \in \mathbb{R}^{1\times n}$,
predicted value matrix $\hat{Y} = \begin{bmatrix} \hat{Y}_1 & \hat{Y}_2 & \cdots & \hat{Y}_n \end{bmatrix} \in \mathbb{R}^{1\times n}$.

activation function $g^{[1]}(s) = \sigma(s) = \dfrac{1}{1+e^{-s}}$, which is applied pointwise to $f^{[1]}$ to produce $\hat{Y}$.

NOTE: Each data point corresponds to a column in input data matrix X, as well as an entry in $f^{[1]}$ and $\hat{Y}$. To emphasize, you should treat $f^{[1]}$, $W^{[1]}$, $\hat{Y}$, Y and X as matrices, and refer to their scalar entries using subscripts, e.g., $f_0^{[1]}$, $W_{10}^{[1]}$, $\hat{Y}_2$.

In this classifier, she decided to use the following loss function:[1]

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

where $\hat{Y} \in (0,1)^{1\times n}$ after applying sigmoid function, $Y \in \{0,1\}^{1\times n}$ such that $Y_i = 1$ if the i-th wine sample is from plant variety A and $Y_i = 0$ if it is from plant variety B, and n is the number of wine samples (i.e., $n = 1100$ in this case). Furthermore, we take $\log(x)$ to be the natural logarithm in this tutorial.

To illustrate, we calculate the loss function for the following sample of two labels:

$$Y = [0\ 1]$$
$$\hat{Y} = [0.2\ 0.9]$$

The loss function will be calculated as follows:

$$\begin{aligned} \mathcal{E} &= -\frac{1}{2}\Big[(1-0)\cdot\log(1-0.2) + 1\cdot\log(0.9)\Big] \\ &= -\frac{1}{2}\Big[\log(0.8) + \log(0.9)\Big] \approx 0.16425 \end{aligned}$$

Note that $\log(1) = 0$ and $\log(0) \to -\infty$. We assume that $\log(x) = \ln(x)$ in this tutorial.

[1] This loss function is usually known as *log loss* or *binary cross-entropy*.

For subquestions (1) and (2), to keep things simple, let us consider the case where $n = 1$.

1. **Show that**

   (a) $\frac{\partial \mathcal{E}}{\partial \hat{Y}} = \left[-\frac{Y_0}{\hat{Y}_0} + \frac{1-Y_0}{1-\hat{Y}_0} \right]$

   We provide the answer for this part as a guiding example:

   Note that for n (all) data points, $\frac{\partial \mathcal{E}}{\partial \hat{Y}} = \left[\frac{\partial \mathcal{E}}{\partial \hat{Y}_0} \frac{\partial \mathcal{E}}{\partial \hat{Y}_1} \cdots \frac{\partial \mathcal{E}}{\partial \hat{Y}_n} \right]$.

   If we assume $n = 1$, it becomes $\frac{\partial \mathcal{E}}{\partial \hat{Y}} = \left[\frac{\partial \mathcal{E}}{\partial \hat{Y}_0} \right]$. Then, the only entry in matrix is

   $$\frac{\partial \mathcal{E}}{\partial \hat{Y}_0} = -\frac{Y_0}{\hat{Y}_0} + \frac{1-Y_0}{1-\hat{Y}_0}$$

   Therefore, the partial derivative matrix is

   $$\frac{\partial \mathcal{E}}{\partial \hat{Y}} = \left[-\frac{Y_0}{\hat{Y}_0} + \frac{1-Y_0}{1-\hat{Y}_0} \right] \text{ when } n = 1$$

   Try to plug in the case where (1) the true label $Y_0 = 0$ and (2) $Y_0 = 1$. This equation tells us how the change in predicted value $\hat{Y}$ will be able to influence the change of error/loss value.

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

   Hint: Since $n = 1$, $\frac{\partial \mathcal{E}}{\partial f^{[1]}} = \left[\frac{\partial \mathcal{E}}{\partial f_0^{[1]}} \right]$. *By chain rule,* $\frac{\partial \mathcal{E}}{\partial f_0^{[1]}} = \frac{\partial \mathcal{E}}{\partial \hat{Y}_0} \frac{\partial \hat{Y}_0}{\partial f_0^{[1]}} \ldots$

   (c) $\frac{\partial \mathcal{E}}{\partial W_{20}^{[1]}} = \left(\frac{\partial \mathcal{E}}{\partial f^{[1]}} \right)_0 X_{20}$

   Hint: Since $n = 1$, $f^{[1]} = \left[f_0^{[1]} \right]$. *Thus,* $\frac{\partial f^{[1]}}{\partial W_{20}^{[1]}} = \left[\frac{\partial f_0^{[1]}}{\partial W_{20}^{[1]}} \right] \ldots$

   NOTE: $\left(\frac{\partial \mathcal{E}}{\partial f^{[1]}} \right)_0$ refers to the first entry (at the 0-th index) of the matrix $\frac{\partial \mathcal{E}}{\partial f^{[1]}}$.

2. Using your answer in (1), derive an expression for $\frac{\partial \mathcal{E}}{\partial W^{[1]}}$. What can you observe from this expression?

3. Let us consider a general case where $n \in \mathbb{N}$. Using your answer to (1), find $\frac{\partial \mathcal{E}}{\partial f^{[1]}}$ when $n \in \mathbb{N}$. Check your answer using the Python notebook.

   Hint: Read the Python notebook.

4. Let's say that Grace has 100 samples from plant variety A and 1000 samples from plant variety B that make up the 1100 total samples. To deal with the imbalanced data set, she decided to introduce two hyper-parameters α and β in the loss function, as shown below:

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

   For example, using the same sample of two labels from part (1), the loss function can be computed as follows:

   $$\begin{aligned} \mathcal{E} &= -\frac{1}{2} \left\{ \beta[(1-0) \cdot \log(1-0.2)] + \alpha[1 \cdot \log(0.9)] \right\} \\ &= -\frac{1}{2} \left\{ \beta \cdot \log(0.8) + \alpha \cdot \log(0.9) \right\} \end{aligned}$$

Why do you think that she introduced the hyper-parameters α and β? How should she set their values?

B Backpropagation for a Deep(er) Network

After training the neural network described in question 1, Grace observed that the training error is high. She thus decided to introduce a hidden layer, resulting in the architecture shown in the figure below.

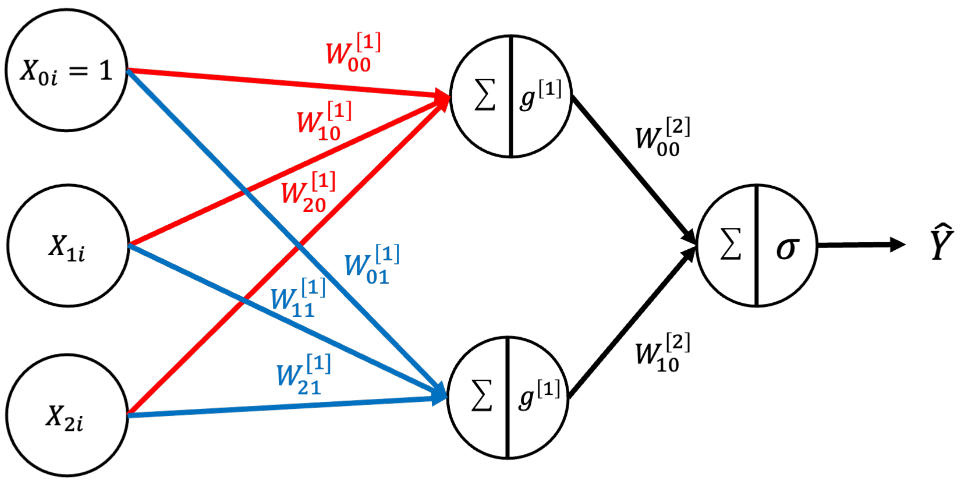

Figure 2: Figure for Question B

Mathematically, the network is given by

$$f^{[1]} = W^{[1]^T} X$$

$$a^{[1]} = g^{[1]}(f^{[1]})$$

$$f^{[2]} = W^{[2]^T} a^{[1]}$$

$$\hat{Y} = g^{[2]}(f^{[2]})$$

where $g^{[1]}(s) = ReLU(s)$, $g^{[2]}(s) = \sigma(s) = \frac{1}{1+e^{-s}}$, $W^{[1]} \in \mathbb{R}^{3\times 2}$ and $W^{[2]} \in \mathbb{R}^{2\times 1}$.

The $ReLU$ function is defined as follows:

$$ReLU(s) = \begin{cases} s & \text{if } s > 0 \\ 0 & \text{otherwise} \end{cases}$$

Note: In Question 1, we only had **one neuron** in $f^{[1]}$, making $f^{[1]}$ a matrix with a single column.

However, in this question, we have two layers of activations, denoted as $f^{[1]}$ and $f^{[2]}$.

- First layer $f^{[1]} \in R^{n\times 2}$:
  Since this layer has **two neurons**, each producing an output for every data point, $f^{[1]}$ is a matrix with **two columns**—one for each neuron.
- Second layer $f^{[2]} \in R^{n\times 1}$:
  This layer has **only one neuron**, so $f^{[2]}$ is a matrix with **single column**.

Mathematically, we represent them as follows:

$$\text{first layer } f^{[1]} = \begin{bmatrix} | & | \\ \text{entries for} & \text{entries for} \\ \text{1st neuron} & \text{2nd neuron} \\ | & | \end{bmatrix} = \begin{bmatrix} f^{[1]}_{00} & f^{[1]}_{01} \\ f^{[1]}_{10} & f^{[1]}_{11} \\ \vdots & \vdots \\ f^{[1]}_{n0} & f^{[1]}_{n1} \end{bmatrix}, \quad \text{second layer } f^{[2]} = \begin{bmatrix} f^{[2]}_{0} \\ f^{[2]}_{1} \\ \vdots \\ f^{[2]}_{n} \end{bmatrix}$$

Similar to question 1, let us keep things simple and consider what happens when $n = 1$. In addition, assume that the loss function used remains the same.

1. Compute $\frac{\partial \mathcal{E}}{\partial W^{[1]}_{11}}$.

   Your answer should not contain any partial derivatives on the right hand side (i.e., they should all be evaluated). What can you observe from this expression?

   Hint: The results found/observations made in question 1 are likely to be useful here.

C Potential Issues with Training Deep Neural Networks

Suppose we use σ (the sigmoid function) as our activation function in a neural network with 50 hidden layers, as per the code in the accompanying Python notebook.

1. Play around with the code. Notice that when performing backpropagation, the gradient magnitudes of the first few layers are extremely small. What do you think causes this problem?
2. Based on what we have learnt thus far, how can we **mitigate** this problem? Test out your solution by modifying the code and checking the gradient magnitudes.

   Hint: You don't need to change much.

D Dying ReLU Problem

This problem occurs when majority of the activations are 0 (meaning the underlying pre-activations are mostly non-positive), resulting in the network dying midway. The gradients passed back are also 0 which leads to poor gradient descent performance and hence poor learning. Refer to the figure below on the ReLU and Leaky ReLU activation functions.

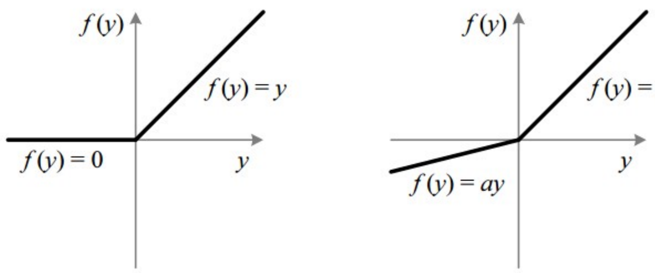

Figure 3: The Rectified Linear Unit (`ReLU`) (left) vs The Leaky Rectified Linear Unit (`Leaky ReLU`) with a as the slope when the values are negative. (right)

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

E Appendix

To learn more about matrix differentation—particularly how dimensions change throughout the calculations—refer to the following article: *https://medium.com/analytics-vidhya/matrix-calculus-for-machine-learning-e0262f0eaa8e*