Deep Learning

Programing Ex 1

Ofer Idan 038467668
Ron Ferens 037222825

למידה עמוקה
תרגיל #1

$x \in \mathbb{R}^d$

$W \in \mathbb{R}^{d \times c}$

$z = W^T x$

(1) $$\text{Softmax}(z)_{[i]} = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

(a)

$$\text{Softmax}(z_i + m) = \text{Softmax}(z+m)_{[i]} = \frac{e^{(z_i+m)}}{\sum e^{(z_j+m)}}$$

$$= \frac{e^{z_i} \cdot e^{m}}{\sum_j \left[e^{z_j} \cdot e^{m}\right]} = \frac{e^{z_i} \cdot \cancel{e^{m}}}{\cancel{e^{m}} \sum_j e^{z_j}} = \text{Softmax}(z)_{[i]}$$

(b) הגדרת פונ' Sigmoid:

$$\text{Sigmoid}(z) \triangleq \frac{1}{1+e^{-z}}$$

עבור $C=2$ מתקבל:

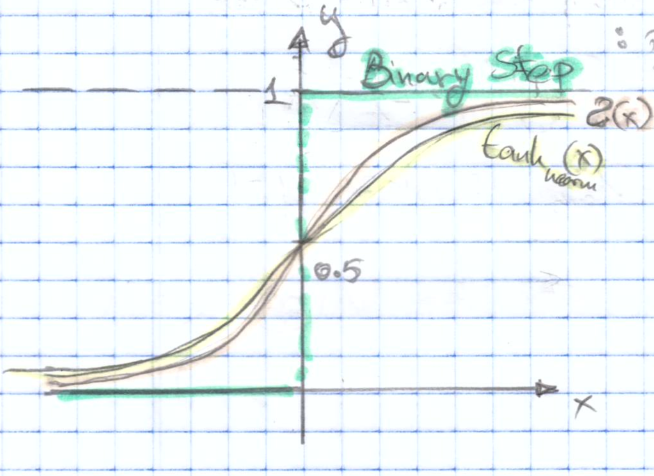
$$\text{Softmax}(z)_{[i]} = \frac{e^{z_i}}{\sum_n e^{z_n}} = \frac{e^{z_i}}{e^{z_i} + e^{z_j}} = \frac{1}{1 + \frac{e^{z_j}}{e^{z_i}}} =$$

$$= \frac{1}{1 + e^{z_j - z_i}} = \frac{1}{1 + e^{z_k}} = \text{Sigmoid}(z)_{[k]}$$

נסמן: $z_i = z_k + z_j \Rightarrow z_j - z_i = z_k$

(c) כשמשווים לפונקציית ה- Sigmoid

אנו יכולים לבחור בפונקציית מדרגה:



$$\text{Binary Step}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

$$\tanh_{norm}(x) = \frac{1}{1+e^{-2x}}$$

(הסבר והוכחה בדף 4 הבא)

ט) ב': אינטואיציה לסיגמואיד שממפה קו ממשי לאינטרבל [0,1]

* אפשר להשתמש באותה אינטואיציה בשם tanh שממפה לטווח [-1,1], ולנרמל אותה לטווח [0,1].

$$\tanh_{new} = \frac{\tanh + 1}{2}$$

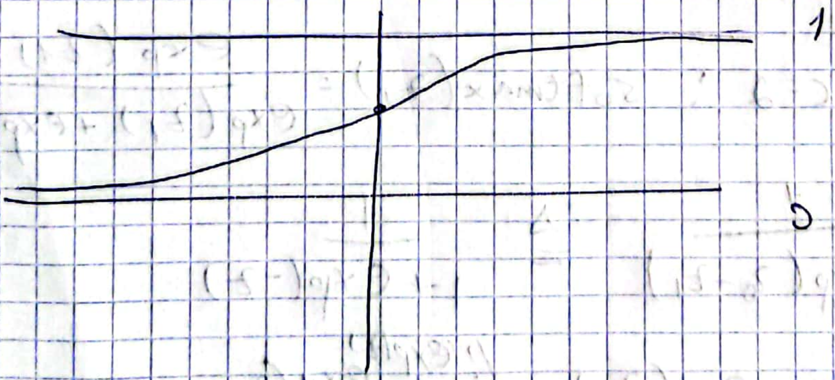
$$\tanh(x) = \frac{\frac{e^x - e^{-x}}{e^x + e^{-x}} + 1}{2} =$$

נעשה מכנה משותף.

$$= \frac{e^x - \cancel{e^{-x}} + e^x + \cancel{e^{-x}}}{2(e^x + e^{-x})} = \frac{\cancel{2} \cdot e^x}{\cancel{2}(e^x + e^{-x})} =$$

$$= \frac{1}{1 + e^{-2x}}$$

קיבלנו וריאציה של סיגמואיד.



$x \in \{0, 1\}^2$ $\qquad f(x) = w^T h + b_2$ $\qquad$ (2)

$h = \max(U^T x + b_1, 0)$

$U \in \mathbb{R}^{2\times2} \qquad w \in \mathbb{R}^2$

$b_1 \in \mathbb{R}^2 \qquad b_2 \in \mathbb{R}$

עבור המקרה הכללי:

$$h = \max\left(\begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} + \begin{bmatrix} b_{10} \\ b_{11} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

$$= \max\left(\begin{bmatrix} u_{00}x_0 + u_{01}x_1 + b_{10} \\ u_{10}x_0 + u_{11}x_1 + b_{11} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

א) עבור $x = [0\ 0]^T$ מתקבל:

$$h = \max\left(\begin{bmatrix} b_{10} \\ b_{11} \end{bmatrix}, 0\right)$$

לשם הפשטות נבחר $\boxed{b_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}}$ ונקבל:

$$f(x=(0,0)) = b_2$$

על מנת שנקבל את הפלט הנדרש, נדרוש $b_2 < 0$:

$$XOR(0,0) = Sign(f(x=(0,0))) = Sign(b_2)\Big|_{b_2<0} = 0$$

נבחר $\boxed{b_2 = -1}$.

ב) עבור $x = [1, 1]^T$ מתקבל:

$$h = \max\left(\begin{bmatrix} u_{00} + u_{01} \\ u_{10} + u_{11} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

גם עבור מקרה זה נרצה לאפס את ערכו של h ע"מ לקבל:

$$XOR(1,1) = Sign(f(x=(1,1))) = Sign(b_2) = 0$$

ולכן נדרוש:

$$\begin{cases} u_{00} + u_{01} \le 0 \Rightarrow u_{01} \overset{(*)}{\le} -u_{00} \\ u_{10} + u_{11} \le 0 \Rightarrow u_{10} \underset{(**)}{\le} -u_{11} \end{cases}$$

② עבור $x=[0,1]^T$ מתקבל:

$$h=\max\left(\begin{bmatrix}u_{01}\\u_{11}\end{bmatrix},\begin{bmatrix}0\\0\end{bmatrix}\right)$$

ע"מ לא לחזור למקרה ⓑ נדרוש כי רק מקדם אחד יהיה >0:

$$u_{01}<0 \Rightarrow f(x=(0,1)) = \underbrace{w_1 u_{11} + b_2}$$

בסוף להיות ≥ 0 ולכן נדרוש:

$$w_1 u_{11} + b_2 \geq 0$$

$$w_1 u_{11} \geq 1 \Rightarrow u_{11} \geq \frac{1}{w_1}$$

③ באותו אופן עבור $x=[1,0]^T$ נקבל:

$$u_{10}<0 \qquad u_{00}\geq \frac{1}{w_0}$$

נבחר $\boxed{w=\begin{bmatrix}w_0\\w_1\end{bmatrix}=\begin{bmatrix}2\\1\end{bmatrix}}$ ונקבל:

$$u_{00}\geq \frac{1}{2} \Rightarrow \boxed{u_{00}=\frac{1}{2}} \overset{*}{\Rightarrow} \boxed{u_{01}=-\frac{1}{2}}$$

$$u_{11}\geq \frac{1}{1} \Rightarrow \boxed{u_{11}=1} \overset{**}{\Rightarrow} \boxed{u_{10}=-1}$$

לסיכום - התקבלה הבחירה הבאה:

$$\boxed{W=\begin{bmatrix}2\\1\end{bmatrix} \quad U=\begin{bmatrix}0.5 & -0.5\\-1 & 1\end{bmatrix} \quad b_1=\begin{bmatrix}0\\0\end{bmatrix} \quad b_2=-1}$$

③ פונ' XOR היא פונקציה לא לינארית ולכן במידה ונחליף את פונ' המקס בפונ' הזהות נקבל פונ' f לינארית:

$$f(x)=w^T h+b_2=w^T(U^T x+b_1)+b_2$$

לפיכך לא נוכל לתאר את פונ' XOR משום שמתקבלת תלות לינארית:

$$f(x=(1,1))=f(x=(0,1))+f(x=(1,0))$$

```python
import numpy as np


def xor(x: np.array) -> int:
	w = np.array([2, 1])
	u = np.array([[0.5, -0.5], [-1, 1]])
	b1 = np.array([0, 0])
	b2 = -1
	h = np.zeros(2)

	for i in range(2):
		# h(x) = max(Ux + b1, 0)
		h[i] = np.maximum(u[i, 0]*x[0] + u[i, 1]*x[1] + b1[i], 0)

	# f(x) = wh(x) + b2
	f = w[0]*h[0] + w[1]*h[1] + b2

	# Returning the result of sing(f)
	return 1 if np.sign(f) >= 0 else 0

# Printing the function's output for all cases:
print(f'XOR(0, 0): {xor(np.array([0, 0]))}')
print(f'XOR(1, 0): {xor(np.array([1, 0]))}')
print(f'XOR(0, 1): {xor(np.array([0, 1]))}')
print(f'XOR(1, 1): {xor(np.array([1, 1]))}')
```

The code’s output:

```
XOR(0, 0): 0
XOR(1, 0): 1
XOR(0, 1): 1
XOR(1, 1): 0
```

$x \in \{0,1\}^2$ $\qquad$ $f(x) = w^T h(x) + b_2$ $\qquad$ (3)

$$h(x) = \max(U^T x + b_1, 0)$$

- נשתמש בקריטריון הבא לפונקצית האופטימיזציה של המודל:

$$\mathcal{L}(y, f(x)) = (y - f(x))^2$$

- נחשב את הנגזרות עבור הפרמטרים של המודל w, U, b_1, b_2:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial f} \cdot \frac{\partial L}{\partial w} = 2(y - f(x)) \cdot (-h(x)) = -2h(x) \cdot (y - f(x))$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial f} \cdot \frac{\partial L}{\partial b_2} = 2(y - f(x)) \cdot (-1) = -2(y - f(x))$$

- עבור הפונ' $h(x)$ והפרמטרים U, b_1:

$$h(x) = \max(U^T x + b_1, 0) = \begin{cases} U^T x + b_1, & U^T x + b_1 > 0 \\ 0, & U^T x + b_1 < 0 \end{cases}$$

ולכן מתקבל:

$$\frac{\partial h(x)}{\partial U} = \begin{cases} x \\ 0 \end{cases} \qquad \frac{\partial h(x)}{\partial b_1} = \begin{cases} 1 \\ 0 \end{cases}$$

- מכאן מתקבל:

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial f} \cdot \frac{\partial f}{\partial h} \cdot \frac{\partial h}{\partial U} = \begin{cases} -2(y - f(x)) \cdot w^T x \\ 0 \end{cases}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial f} \cdot \frac{\partial f}{\partial h} \cdot \frac{\partial h}{\partial b_1} = \begin{cases} -2(y - f(x)) \cdot w \\ 0 \end{cases}$$

```python
from typing import Tuple
import numpy as np
import matplotlib.pyplot as plt

def create_dataset() -> Tuple[np.array, np.array]:
	# Creating the dataset
	samples = np.array([(0, 0), (0, 1), (1, 0), (1, 1)])
	labels = np.array([-1, 1, 1, -1])
	return samples, labels

def init_model() -> Tuple[np.array, float, np.array, float]:
	# Random initializing parameters weights
	np.random.seed(101)
	w_init = np.random.rand(2)
	b2_init = np.random.rand(1)
	u_init = np.random.rand(2, 2)
	b1_init = np.random.rand(1)
	return w_init, b2_init, u_init, b1_init

def main():
	# Creating the dataset
	x, y = create_dataset()

	# Random initialization of the model's parameters
	w, b2, u, b1 = init_model()

	# Training the model
	num_epochs = 100
	learning_rate = 0.001
	loss = []

	for e in range(num_epochs):
		# Calculating the model's output
		h = np.maximum(np.matmul(u, x.T) + b1, 0)
		f = np.matmul(w, h) + b2

		# Calculating the optimization criteria
		epoch_loss = np.sum((y - f) ** 2)
		loss.append(epoch_loss)

		# Calculating the derivatives
		dl_dw = np.sum(-2 * np.matmul(h, (y - f)))
		dl_db2 = np.sum(-2 * np.matmul(np.ones_like(y), (y - f)))
		dl_du = np.sum(-2 * np.matmul((y - f), np.matmul(w, x.T)))
		dl_db1 = np.sum(-2 * np.matmul((y - f), np.matmul(w, np.ones_like(x.T))))

		# Applying Gradient Descent to optimize the model
		w = w - learning_rate * dl_dw
		b2 = b2 - learning_rate * dl_db2
		u = u - learning_rate * dl_du
		b1 = b1 - learning_rate * dl_db1

	plt.figure()
	plt.plot(loss)
	plt.title('Model Optimization\n' + r'$\mathcal{L} = (y - f(x)) ^2$')
	plt.xlabel('Epochs')
	plt.ylabel('Loss')
	plt.grid(True)
	plt.show()


	print('Final Model parameters:')
```

```python
        print(f'\tW = {w}')
        print(f'\tb2 = {b2}')
        print(f'\tU = {u}')
        print(f'\tb1 = {b1}')

if __name__ == "__main__":
        main()
```

The code's output:

```
Final Model parameters:

        W = [0.02804452 0.08231348]
        b2 = [-0.14378446]
        U = [[0.10720013 0.62095545]
             [0.76957533 0.24264469]]
        b1 = [0.78708694]
```