בס"ד

סדנאות תכנות בשפת C ו- C++ (קורס 67315)
תרגיל 4

תאריך ההגשה של התרגיל: יום שלישי, ה-20 ביוני, 2023 – עד השעה 22:00.

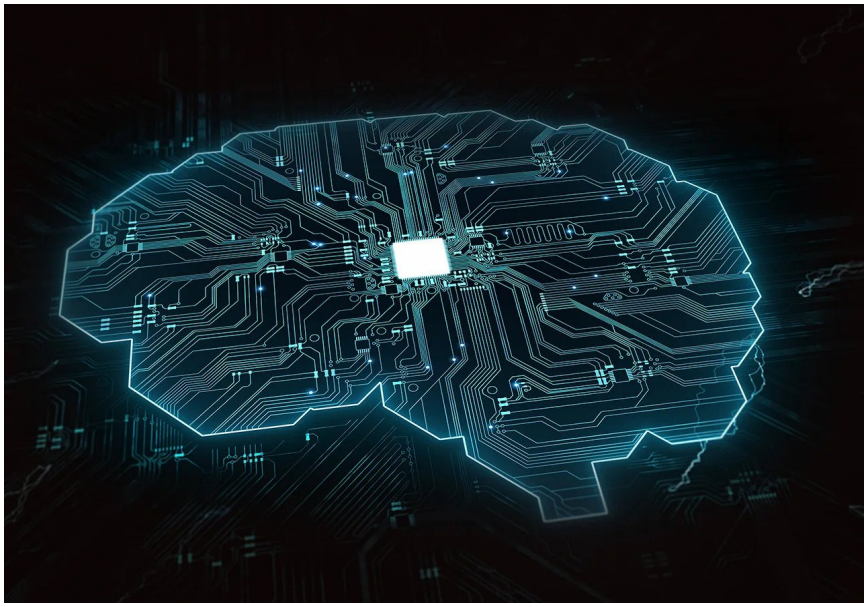
נושאי התרגיל: Exceptions, Rule of Three, References, Operator Overloading, Classes, Introduction to C++

אנא הקפידו לקרוא את כל התרגיל מתחילתו ועד סופו לפני שתתחילו לממש אותו.

1 רקע

בתרגיל זה נכתוב תוכנה לזיהוי ספרות הנכתבות בכתב יד. התוכנה שלנו תקבל כקלט תמונה של ספרה בין 0 ל- 9 ותחזיר כפלט את הספרה אשר זוהתה. נעשה זאת על ידי בניית מודל של רשת נוירונים. הרשת שנריץ תגיע לדיוק של כ- 96 אחוזים בזיהוי ספרות.

1.1 הקדמה



רשת נוירונים היא מודל בלמידת מכונה המבוסס על מבנה המוח האנושי: נוירון מקבל גירוי חשמלי מנוירונים אחרים - אם הגירוי הזה עובר סף מסוים הוא שולח בעצמו אות אל נוירונים אחרים. המוח מורכב ממספר רב של נוירונים המקושרים זה לזה ברשת מורכבת, ויחד הם מסוגלים לבצע את הפעולות הנדרשות ממנו. רשת נוירונים מלאכותית (Artificial neural network) פועלת באופן דומה. ברשתות אלו נעשה שימוש בזיהוי ומיקום של עצמים בתמונה, הבנת שפה אנושית וניתוחה, יצירת טקסט ועוד.
מוצרים רבים בחיינו משתמשים ברשתות נוירונים: עוזרים קוליים (Amazon Alexa, Apple Siri), השלמה אוטומטית לתוכן המייל ב- Gmail, זיהוי מחלות בתמונות סריקה רפואית ועוד.

שימו לב: למידת מכונה בכלל, ורשתות נוירונים בפרט, הינם נושאים רחבים ומורכבים ולכן לא יכללו בתוכן תרגיל זה. לצורך מימוש התרגיל אין צורך להבין איך ולמה עובדת רשת הנוירונים. הרקע התיאורטי הנדרש יוצג בסעיף 1.2, פרטי הרשת שנבנה יובאו בסעיף 2.2, והמחלקות למימוש יובאו בסעיף 3.
בתרגיל זה, המשקולות שאתם תקבלו בתור קלט הם משקולות של רשת מאומנת, ואתם רק עושים פרדיקציה.
מומלץ לצפות בסרטון הבא המפרט על המבנה של רשת נוירונים וכיצד ניתן לממש אותה באמצעות אלגברה לינארית: https://www.youtube.com/watch?v=aircAruvnKk

1.2 רקע תיאורטי

1.2.1 רשת נוירונים Fully Connected

- רשת בנויה משכבות (סעיף 1.2.2).
- הקלט של כל שכבה הוא וקטור, והפלט הוא וקטור אחר.
- הפלט של כל שכבה הוא הקלט של השכבה הבאה.
- קלט הרשת הוא וקטור המייצג את האובייקט שהרשת תעבד. ברשת שלנו, הוא מייצג תמונה של ספרה (סעיף 2.2.2).
- פלט הרשת הוא וקטור המייצג את המסקנה של הרשת. ברשת שלנו, הוא מייצג את הספרה שהרשת זיהתה (סעיף 2.2.3).

1.2.2 שכבה ברשת

כל שכבה ברשת מקבלת וקטור <u>קלט</u> $x \in \mathbb{R}^m$ ומחזירה וקטור <u>פלט</u> $y \in \mathbb{R}^n$ באמצעות הפעולה המתמטית הבאה:

$$y = f(W \cdot x + b)$$

כאשר:

- $W \in M_{mxn}$ מטריצה שאיבריה נקראים המשקולות של השכבה (Weights).
- $b \in \mathbb{R}^n$ וקטור שנקרא ההיסט (Bias) של השכבה.
- $f : \mathbb{R}^n \to \mathbb{R}^n$ פונקציית האקטיבציה של השכבה (סעיף 1.2.3)

כלומר, בהינתן וקטור קלט $x \in \mathbb{R}^m$, הוקטור $y = f(W \cdot x + b) \in \mathbb{R}^n$ יהיה הפלט של השכבה.

1.2.3 פונקציית אקטיבציה

במימוש פונקציות האקטיבציה , על הפונקציות לקבל ולהחזיר אובייקט מסוג מטריצה.

פונקציה $f : \mathbb{R}^n \to \mathbb{R}^n$ אשר מחזירה את התוצאה הסופית של שכבה ברשת הנוירונים. פונקציה זו אינה לינארית.

בתרגיל נממש שתי פונקציות אקטיבציה שונות:

1. פונקציית ReLu (פועלת על וקטור $x \in \mathbb{R}^n$ ומבצעת את הפעולה על כל קואורדינטה בנפרד).

$$\forall x \in \mathbb{R} \;\; \mathrm{relu(x)} \begin{cases} x & x \geqslant 0 \\ 0 & else \end{cases}$$

2. פונקציית Softmax

$$\forall x \in \mathbb{R}^n \quad soft\max(x) = \frac{1}{\sum_{k=1}^{n} e^{x_k}} \begin{bmatrix} e^{x_1} \\ e^{x_2} \\ \vdots \\ e^{x_n} \end{bmatrix} \in \mathbb{R}^n$$

x_k - הקואורדינטה ה- k של $x \in \mathbb{R}^n$.

e^t - הפונקציה המעריכית $\exp(t)$. לצורך החישוב ניתן להשתמש בפונקציה std::exp המיובאת מ- cmath.

הפונקציה מקבלת וקטור $x \in \mathbb{R}^n$ וממירה אותו לוקטור התפלגות (יש הסבר בסרטון, זה וקטור שאיבריו הם מספרים אי-שליליים שסכומם 1) באופן התואם את הפלט הסופי של הרשת שלנו. בתרגיל זה וקטור הוא מטריצה עם טור אחד.

פונקציות האקטיבציה עשויות לקבל כקלט מטריצות שאינן מהצורה nx1. במקרה זה אין לזרוק שגיאה ואין לעשות vectorize. צריך להחזיר את האקטיבציה על כל המטריצה (לא רק על העמודה הראשונה) עם מימדים זהים לקלט.

2 מימוש הרשת

2.1 שימוש ב-float ודיוק הרשת

איברי המטריצה שנממש יהיו מטיפוס $float\,(32-bit)$.

באופן המימוש של float במעבד, פעולות האריתמטיקה אינן בהכרח אסוציאטיביות, כלומר לא בהכרח יתקיים:

$$(a+b)+c \;=\; a+(b+c)$$

לכן, כדי להימנע משגיאות נומריות בעת ביצוע כפל מטריצות, אנא ממשו את סדר הפעולות לפי ההגדרה המתמטית שלמדתם בלינארית 1:

$$(A \cdot B)_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

הרשת שנריץ מגיעה לכ- 96 אחוזי דיוק. לכן, הרשת עלולה לטעות בחלק מהתמונות שתזינו לה – זוהי התנהגות תקינה של התוכנה. גם אם אחוזי ההצלחה של הרשת שלכם נמוכים במעט מ- 96 אחוזים, ייתכן כי הדבר נובע משגיאות נומריות, ולא צפויה הורדה של נקודות במקרה זה. עם זאת, אם אחוזי ההצלחה של הרשת נמוכים משמעותית מרף זה, ייתכן שהדבר נובע מטעות במימוש.

2.2 תיאור הרשת

2.2.1 שכבות הרשת

- הרשת מורכבת מ- 4 שכבות:

שכבה	משקולות - Weights	היסט - Bias	פונקציית אקטיבציה
1 (כניסה)	$W_1 \in M_{128\times784}$	$b_1 \in \mathbb{R}^{128}$	Relu
2	$W_2 \in M_{64\times128}$	$b_2 \in \mathbb{R}^{64}$	Relu
3	$W_3 \in M_{20\times64}$	$b_3 \in \mathbb{R}^{20}$	Relu
4 (מוצא)	$W_4 \in M_{10\times20}$	$b_4 \in \mathbb{R}^{10}$	Softmax

- כלומר, על מנת לממש את הרשת עלינו לשרשר את רצף הפעולות הבא:

$$\begin{aligned} r_1 &= Relu(W_1 \cdot x + b_1) \\ r_2 &= Relu(W_2 \cdot r_1 + b_2) \\ r_3 &= Relu(W_3 \cdot r_2 + b_3) \\ r_4 &= Soft\max(W_4 \cdot r_3 + b_4) \end{aligned}$$

- x – וקטור הקלט לרשת (סעיף 2.2.2).
- r_i– הפלט של השכבה ה- i , שהוא גם הקלט לשכבה ה- (i + 1) .
- r_4– הפלט של השכבה הרביעית, שהוא וקטור הפלט של הרשת (סעיף 2.2.3).

2.2.2 וקטור הקלט

- כל תמונה מקודדת בתור מטריצה A בגודל 28 × 28 של פיקסלים בגווני אפור (Grayscale), וכל מספר במטריצה הוא ערך בין 0 ל- 1, כלומר $A \in M_{28\times28}([0,1])$.
- לנוחיותכם, בקבצי העזר נמצא הקובץ plot_img.py אשר מקבל כקלט נתיב לתמונה ומציג אותה בחלון חדש.
- וקטור הקלט שישלח לרשת יהיה וקטור (מטריצה עם עמודה אחת) עם $28 \cdot 28 = 748$ שורות.

2.2.3 וקטור הפלט

- וקטור הפלט מהשכבה האחרונה הוא וקטור התפלגות

  (וקטור שאיבריו הם מספרים אי-שליליים שסכומם 1 באורך 10).
- כל אינדקס בוקטור מייצג ספרה בין 0 ל- 9.
- הערך של האינדקס מייצג את הסיכוי שזוהי הספרה בתמונה, לפי הרשת.
- התשובה שתיתן הרשת היא האינדקס עם הערך המקסימלי, כלומר הספרה הסבירה ביותר, וההסתברות של אותה ספרה.
- במקרה של שוויון, נחזיר את האינדקס הנמוך מבין השניים.

- לדוגמא:

Value	0	0.003	0.08	0	0	0	0	0.9	0.007	0.01
Index	0	1	2	3	4	5	6	7	8	9

בהינתן וקטור הפלט הזה, תשובת הרשת תהיה שהספרה בתמונה היא 7 בהסתברות 90%.

2.3 מהלך ריצת התוכנית

שימו לב: חלק זה ממומש עבורכם בקובץ main.cpp המצורף לכם עם קבצי התרגיל מלבד הפונקציה readFileToMatrix שעליכם לממש.

התוכנה תקבל בשורת הפקודה נתיבים לקבצי המשקולות וההיסטים כקבצים בינאריים.

נריץ את התוכנה עם המשקולות וההיסטים כך:

$./mlpnetwork w1 w2 w3 w4 b1 b2 b3 b4

- w_i – נתיב לקובץ המשקולות של השכבה ה- i
- b_i – נתיב לקובץ ההיסט של השכבה ה- i

כאשר התוכנה רצה היא ממתינה לקלט מהמשתמש. הקלט יהיה נתיב לקובץ של תמונה המכילה ספרה. התוכנה:

1. תפתח את הקובץ ותטען את התמונה למטריצה.
2. תכניס את המטריצה אל הרשת כקלט.
3. כאשר התקבלה תוצאה, התוכנה תדפיס את התמונה, את הספרה שהרשת זיהתה והסתברותה. לדוגמא (מתוך פתרון בית הספר):

4. התוכנה תמתין לקלט חדש.

כאשר נזין לתוכנה q – התוכנה תצא עם קוד 0.

3. המחלקות למימוש

הינכם נדרשים לממש את המחלקות הבאות בלבד. אין להגיש מחלקות נוספות.

- בטבלאות המובאות לפניכם רשומות כלל הפונקציות והאופרטורים שעליכם לממש.
- אין להרחיב את ה-API המפורט, כלומר אין להוסיף פונקציות public למחלקות (ניתן להוסיף פונקציות private).
- שימו לב, עליכם לכתוב את כל החתימות של ה API בקבצי h. ולכתוב את המימושים בקבצי cpp.
- חישבו היטב על החתימה של כל פונקציה: מהו ערך ההחזרה שלה? האם היא משנה את האובייקט הנוכחי? כיצד נגדיר את הטיפוס של הארגומנטים שלה?
- שימו לב: עליכם להחליט איפה יש להשתמש ב-const, איפה המשתנים וערכי ההחזרה צריכים להיות by value או by reference, והאם לממש כל פונקציה כחלק מהמחלקה (member function) או כפונקציה העומדת בפני עצמה (standalone, non-member function).
- איסור על שימוש ב-STL: בתרגיל זה אין להשתמש בספריית STL של C++, ובפרט לא במבני נתונים כמו std::vector.

3.1 המחלקה Matrix

מחלקה זו מייצגת אובייקט של מטריצה (גם וקטור הינו מטריצה, בעלת עמודה אחת ו- n שורות). נזכיר כי איברי המטריצה יהיו מטיפוס float.

Description	Name	Comments
		Constructors
Constructor	Matrix(int rows, int cols)	Constructs Matrix of size $rows \times cols$. Inits all elements to 0.
Default Constructor	Matrix()	Constructs Matrix of size 1×1. Inits the single element to 0.
Copy Constructor	Matrix(matrix)	Constructs a matrix from another Matrix m.
Destructor	~Matrix()	
		Methods & Functions
Getter	get_rows()	returns the amount of rows as int.
Getter	get_cols()	returns the amount of columns as int.

Description	Name	Comments

	transpose()	Transforms a matrix into its transpose matrix, for example:
		$(A.transpose())_{ij} = A_{ji}$
		Supports function call chaining/concatenation (שרשור פונקציות).
		For example: Matrix A(5,4), B(4,5);
		A.transpose(); // A has 4 rows and 5 cols
		B.transpose().transpose(); // function call concatenation
		// Matrix B remains unchanged after transposing it twice.

	vectorize()	Transforms a matrix into a column vector (see section 3.1.2).
		Supports function call chaining/concatenation. For example:
		Matrix M(5,4);
		M.vectorize(); // M has 20 rows and 1 col
		M.transpose().vectorize().transpose();
		// The above is an example of function call concatenation

	plain_print()	Prints matrix elements, no return value. Prints space after each element (including last element in row). Prints newline after each row (including last row).
	dot(matrix)	Returns a matrix which is the element-wise multiplication (Hadamard product) of this matrix with another matrix m: $\forall i,j:\ (A.dot(B))_{ij} = A_{ij} \cdot B_{ij}$
	norm()	Returns the Frobenius norm of the given matrix: $A.norm() = \sqrt{\sum_{i,j} A_{ij}^2}$
	rref()	Returns a new Matrix that is the reduced row echelon form of the original. The original Matrix should be unchanged (See section 3.1.4). Note: This function is a bonus and you do not have to implement it.
	argmax()	Returns index of the largest number in the matrix. For example: Matrix m(2,2); m(0,0) = 10; m(0,1) = 10; m(1,0)=10; m(1,1) = 20; cout<<m.argmax(); //Prints 3, since m[3] is the largest number (see brackets operator below to understand what m[3] means).

Description	Name	Comments
Operators		
+	Matrix addition	Matrix a, b; → a + b
=	Assignment	Matrix a, b; → a = b
*	Matrix multiplication	Matrix a, b; → a * b
*	Scalar multiplication on the right	Matrix m; float c; → m * c
*	Scalar multiplication on the left	Matrix m; float c; → c * m
()	Parenthesis indexing	For cell (i,j) in matrix m: m(i,j) will return the element in cell (i,j) in m. For example: Matrix m(5,4); m(1,3) = 10; float x = m(1,3); // x equals 10.0
[]	Brackets indexing	Let m be a matrix with i rows and j columns. Then m has i*j cells. For every 0 <= k < i*j : m[k] will return the k'th element as if m is represented as a vector (See section 3.1.2). For example: Matrix m(5,4); m[3] = 10; float x = m[3]; // x equals 10.0
<<	Output stream	Pretty print of matrix as (See section 3.1.1)
>>	Input stream	Fills matrix elements. The stream must be big enough to fill the entire matrix (don't trust the user to validate it). Note: We don't care if the stream is too big for the matrix, but on the other hand the stream must have enough data to fill the matrix all the way. If it’s too short, then throw an error.

הערה: לא כל הפונקציות שאתם צריכים לממש במחלקת המטריצה הן נדרשות למימוש רשת הניורונים שלכם! לדוגמא, הפונקציה rref לא נדרשת לרשת.

3.1.1 הדפסת תמונה של מטריצה

כדי להדפיס את התמונה המיוצגת במטריצה A באמצעות אופרטור << נשתמש בפסאודו-קוד הבא:

```
for i = 1 to A.rows:
    for j = 1 to A.cols:
        if A(i, j) > 0.1:
            print "**" (double asterisk)
        else:
            print "  " (double space)
    print newline
```

3.1.2 אינדקס יחיד לזכרון דו-מימדי

מטריצה A הינה אובייקט דו-מימדי, ולכן אנו זקוקים לשני אינדקסים על מנת לגשת לאיבר בה. פעמים רבות, נוח יותר לגשת לכל איבר במטריצה באמצעות אינדקס יחיד. נבצע את המיפוי מזוג אינדקסים לאינדקס יחיד באופן הבא:

$$A(i,j) \iff A[i \cdot row\_size + j]$$

i - אינדקס השורה

j - אינדקס העמודה

row_size - אורך השורה (מספר עמודות)

3.1.3 קריאת תמונה מקובץ בינארי למטריצה

כדי לקרוא קובץ בינארי מתוך Input stream (במימוש של האופרטור <<), עליכם להשתמש בפונקציה std::istream::read, שמקבלת char* ומספר בתים\תווים לקרוא. תוכלו למצוא תיעוד ומידע על הפונקציה בלינקים הבאים:

https://www.tutorialspoint.com/reading-and-writing-binary-file-in-c-cplusplus

https://www.cplusplus.com/reference/istream/istream/read

שימו לב, במקרה הזה תוכלו לעשות casting מפורש ל - char*.

3.1.4 Reduced Row Echelon Form (ייצוג מטריצה מדורגת מצומצמת) בונוס (עד 5 נקודות)

המתודה הנ"ל מחזירה מטריצה שהיא המדורגת מצומצמת של המטריצה המקורית, כאשר על המטריצה המקורית להישאר כפי שהייתה, ללא שינוי כלל.

הגדרות:

איבר מוביל - תהי $A \in M_{n\times n}$. האיבר המוביל בשורה של מטריצה הוא המקדם השמאלי ביותר בשורה ששונה מאפס. יתכן כי לא יהיה איבר מוביל בכל שורה שכן תהיה שורת אפסים.

מטריצה מדורגת מצומצמת - מטריצה נקראת מדורגת מצומצמת כאשר מתקיימים 4 התנאים הבאים:

1. האיבר המוביל נמצא בכל שורה (אם קיים) ושווה 1.
2. כל המקדמים מעל ומתחת לאבר המוביל הם אפסים.
3. האיבר המוביל נמצא משמאל לאיבר המוביל בשורה שמתחתיו. במילים אחרות, אם בשורה ה- k האיבר המוביל מופיע בעמודה ה- l, ובשורה ה- $k+1$ האיבר המוביל מופיע בעמודה ה- l', אז $l' > l$.
4. אם ישנן שורות אפסים הן יופיעו מתחת לשורה האחרונה שאינה שורת אפסים.

לכל מטריצה קיימת צורה מדורגת מצומצמת יחידה.

דוגמא:

$$\begin{bmatrix} 2 & 4 & 8 & 1 \\ 16 & 3 & 5 & 0 \\ 1 & 0 & 4 & 3 \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1.25 \\ 0 & 0 & 1 & 0.75 \end{bmatrix}$$

(Original Matrix) (Reduced Row Echelon Form matrix)

הערות:

- המטריצות עלולות להיות עם ערכים 0- (מינוס אפס) במקום 0 (אפס). בשפת C++ הם שקולים ואפשר להתעלם מכך. לקריאה נוספת על תופעה זו הוספנו קישור למטה.
- יש לממש את הפונ' בקובץ cpp.
- כדאי לחלק את המימוש למספר פונקציות, כאשר כל פונקציה עושה שלב אחד בתהליך.

קישורים:

RREF Matrix calculator:
https://www.emathhelp.net/calculators/linear-algebra/reduced-row-echelon-form-rref-calculator/?i=%5B%5B2%2C4%2C8%2C1%5D%2C%5B16%2C3%2C5%2C0%5D%2C%5B1%2C0%2C4%2C3%5D%5D&reduced=on
RREF Wikipedia Reference:
https://en.wikipedia.org/wiki/Row_echelon_form#Reduced_row_echelon_form
Signed Zero Wikipedia Reference:
https://en.wikipedia.org/wiki/Signed_zero

3.2 קבצי Activation

בקבצים אלה נגדיר את פעולת פונקציות האקטיבציה. בקבצי ה- Activation עליכם לממש את פונקציות האקטיבציה relu ו- softmax כפונקציות השייכות ל- namespace activation . כלומר, על מנת לקרוא לפונקציות צריך לרשום activation::relu או activation::softmax . על פונקציות אקטיבציה לעבוד על כל המטריצה, ולא רק על הטור הראשון, ולהחזיר אובייקט מטריצה בעל אותם מימדים כמו המטריצה המקורית.
טיפ: שימו לב, נשתמש בתרגיל זה ב- function pointers על מנת לגשת לפונקציות אלו. מומלץ להגדיר typedef של פוינטר לפונקציית אקטיבציה ולהשתמש בו בתוכנית.

3.3 המחלקה Dense

מחלקה זו מייצגת שכבה ברשת הנוירונים. **אין לממש קונסטרוקטור דיפולטיבי**.

Description	Name	Comments
	Constructors	
Constructor	`Dense(weights, bias, ActivationFunction)`	Inits a new layer with given parameters. C'tor accepts 2 matrices and activation function
	Methods	
Getter	`get_weights()`	Returns the weights of this layer.
Getter	`get_bias()`	Returns the bias of this layer.
Getter	`get_activation()`	Returns the activation function of this layer.
	Operators	
()	Parenthesis	Applies the layer on the input and returns an output matrix. Layers operate as per section 2.2.1 for example: Dense layer(w, b, act); Matrix output = layer(input);

3.4 המחלקה MlpNetwork

מחלקה זו תשמש אותנו לסדר את השכבות השונות למבנה רשת ותאפשר הכנסה של קלט לרשת וקבלת הפלט המתאים. מחלקה זו מממשת ספציפית את הרשת המתוארת במסמך זה (סעיף 2.2.1).
שימו לב ש struct digit מומש עבורכם בקבצים שקיבלתם. **אין לממש קונסטרוקטור דיפולטיבי.**
אתם יכולים להניח שהקלט תקין ואין צורך לעשות עליו vectorize.
נקודה למחשבה: מה היה נדרש לממש במחלקה זו על מנת לתמוך ברשת עם מספר כללי n שכבות וגודל שכבות כלשהו הניתן בזמן ריצת התוכנית?

Description	Name	Comments
Constructors		
Constructor	MlpNetwork(weights[], biases[])	Accepts 2 arrays of matrices, size 4 each. one for weights and one for biases. constructs the network described (sec. 2.2).
Operators		
()	Parenthesis	Applies the entire network on input. Returns a digit struct type. For example: MlpNetwork m(...); digit output = m(img);

4. טיפול בשגיאות

בתרגיל זה נדרוש מכם להשתמש ב- exceptions.

- חשבו בעצמכם היכן יכולות להיות שגיאות (בדיקת הקלטים של הפונ', בדיקת ערכי החזרה של פעולות שונות ועוד).
- במקרה של שגיאה, זרקו חריגה (exception) מתאימה בהתאם להוראות הבאות:
  - אם התרחשה שגיאה הנוגעת לאורכים ומימדים בעייתיים, זרקו std::length_error.
  - אם התרחשה שגיאה הנוגעת לגישה למיקום לא חוקי, זרקו std::out_of_range.
  - אם התרחשה שגיאה עקב קלט לא נכון מהמשתמש, או מבעיות רלוונטיות לנושא קבצים, זרקו std::runtime_error.
  - במקרה שהקצאת זיכרון נכשלה <u>אינכם נדרשים</u> לזרוק חריגה. החריגה std::bad_alloc תיזרק באופן אוטומטי במקרה זה. לפירוט,קראו: https://en.cppreference.com/w/cpp/memory/new/bad_alloc.

    אין זה אומר שלא יהיה לכם דליפות זיכרון, חישבו מתי נרצה לא לזרוק מיד אחרי הקצאה כושלת bad_alloc ונרצה לשחרר זיכרון קודם. שימו לב שמימוש טוב ויעיל ימנע בעיות כאלה.

    בתרגיל זה <u>אינכם</u> נדרשים לשחרר את הזיכרון במקרה של שגיאה (כיוון שזאת אחריות משתמש הספרייה לשחרר את הזיכרון).

רמז: שימו לב לדרישות ניהול הזיכרון במקרה של זריקת חריגה מ- constructor.

5. קימפול והרצה

בקבצי העזר לתרגיל הניתנים לכם, מצורף קובץ Makefile על מנת לקמפל את התוכנה. על התוכנה להתקמפל באמצעות הפקודה הבאה: make mlpnetwork
נריץ את התוכנית כמפורט בסעיף 2.3

5.1 Presubmit קובץ ה presubmit זמין בנתיב: ~labcc2/presubmit/ex4/srcs/presubmit.cpp

6. הקבצים להגשה

Matrix.h	Matrix.cpp
Activation.h	Activation.cpp
Dense.h	Dense.cpp
MlpNetwork.h	MlpNetwork.cpp

7. הערות וסיכום

7.1 הנחיות כלליות

- קראו בקפידה את הוראות תרגיל זה ואת ההנחיות להגשת תרגילים באתר הקורס.
- ממליצים מאוד לעשות את "A special code runner for Ex4 - binary files" בנושא קבצים בינאריים.
- בתרגיל נתייחס לוקטור כאובייקט מסוג מטריצה. (וקטור הוא מטריצה בעלת עמודה אחת ו- n שורות).
- ניתן להניח כי הקבצים שתקראו לא יהיו גדולים מהגודל של המטריצה.
- מעבר ל-++C: זכרו להשתמש בפונקציות ובספריות של ++C ולא C. למשל, נשתמש ב-new ו-delete ולא ב- malloc ו- free, ב- std::string ולא ב- *char, ובספריה <cmath> במקום math.h.
- איסור על שימוש ב-STL: נזכיר בשנית כי בתרגיל זה נאסר השימוש בספריית STL של ++C, ובפרט אסור להשתמש במבני נתונים כגון std::vector.
- ניהול זיכרון דינמי: כזכור, הקצאת זיכרון דינמית <u>מחייבת</u> את שחרור הזיכרון, למעט במקרים בהם ישנה שגיאה המחייבת סגירת התוכנית באופן מיידי עם קוד שגיאה (כלומר קוד יציאה 1). תוכלו להיעזר בתוכנה valgrind כדי לחפש דליפות זיכרון בתוכנית שכתבתם.
- שימוש ב-reference: הקפידו לא להעתיק by value משתנים כבדים, אלא להעבירם היכן שניתן by reference.
- שימוש ב-const: הקפידו להשתמש במילה השמורה const היכן שנדרש מכם בהגדרת הפונקציות והארגומנטים.
- חובה לתעד את כל המימוש שלכם בתרגיל.
- סקריפט Pre-submission: ודאו כי התרגיל שלכם עובר את ה-Pre-submission Script ללא שגיאות או אזהרות.
- כפילויות קוד: שימו לב שיש בתרגיל זה המון הזדמנויות למחזר קוד. אל תעבדו קשה, תעבדו חכם!
- אתם יכולים להניח שקבצי הקלט לא יהיו גדולים מהגודל של המטריצה. אבל לא ניתן להניח שקבצי הקלט לא קטנים מגודל המטריצה.
- אנחנו מעודדים אתכם לכתוב טסטים ולבדוק את עצמכם במהלך פתירת התרגיל.
- **והכי חשוב, שיהיה לכם בהצלחה :)**

האוניברסיטה העברית בירושלים
בית הספר להנדסה ולמדעי המחשב ע"ש רחל וסלים בנין