האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע`ש רחל וסלים בנין

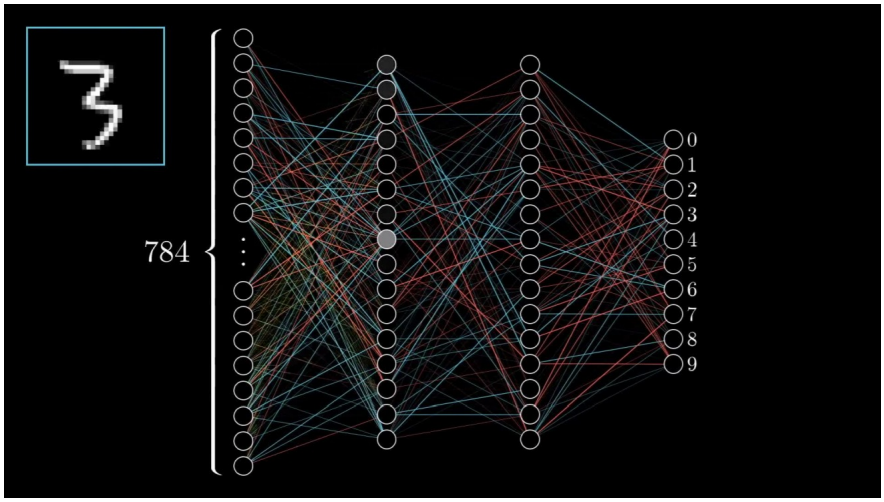
סדנאות תכנות בשפת C ו־++C (קורס 67312)
++C - תרגיל 1

תאריך ההגשה של התרגיל: יום שלישי, ה־15 ביוני, 2021 - עד השעה 23:59;

הגשה מאוחרת (בהפחתת 10 נקודות): יום רביעי, ה־16 ביוני, 2021 - עד השעה 23:59.

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

אנא הקפידו לקרוא את כל התרגיל מתחילתו ועד סופו לפני שתגשו לממשו.



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. על מנת לממש את המודל עצמו, נצטרך לממש גם מחלקות שימושיות כגון מטריצה, שכבה ברשת ופונקצית אקטיבציה. 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, ואם הגירוי הזה גדול מסף 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 רגיל זה. לצורך מימוש התרגיל אין צורך להבין איך ולמה עובדת רשת הנוירונים, ולא נאמן רשת בתרגיל. הרקע התיאורטי הנדרש יוצג בסעיף 1.2, פרטי הרשת שנבנה יובאו בסעיף 2.2, והמחלקות למימוש יובאו בסעיף 3.

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 2.1.2)
- פלט הרשת הוא וקטור המייצג את המסקנה של הרשת. ברשת שלנו, הוא מייצג את הספרה שהרשת זיהתה (סעיף 2.1.3)

1.2.2 שכבה ברשת

כל שכבה ברשת הופכת וקטור קלט $x \in \mathbb{R}^m$ אל וקטור פלט $y \in \mathbb{R}^n$ באמצעות הפעולות הבאות, בסדר בו הן מופיעות כאן:

- **העתקה לינארית** $T : \mathbb{R}^m \to \mathbb{R}^n$
  - T ניתנת לייצוג על ידי כפל במטריצה $W \in \mathbb{M}_{n\times m}(\mathbb{R})$

- נבצע $T(x) = W \cdot x$
- איברי המטריצה W נקראים המשקולות של השכבה (Weights)

- **היסט** $b \in \mathbb{R}^n$
  - לאחר ההעתקה, נחבר אל התוצאה את הווקטור b
  - הווקטור b נקרא ההיסט של השכבה (Bias)

- **פונקציית אקטיבציה** $f : \mathbb{R}^n \to \mathbb{R}^n$
  - לאחר ההיסט, נפעיל על התוצאה פונקציית אקטיבציה (סעיף 1.2.3)
  - הווקטור שיתקבל יהיה הפלט הסופי של השכבה

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

1.2.3 פונקציית אקטיבציה

פונקציה $f : \mathbb{R}^n \to \mathbb{R}^n$ אשר מחזירה את התוצאה הסופית של שכבה ברשת הנוירונים. פונקציה זו איננה לינארית. בתרגיל נממש שתי פונקציות אקטיבציה שונות:

- **פונקציית ReLU**

$$\forall x \in \mathbb{R} \qquad ReLU(x) = \begin{cases} x & if x \geq 0 \\ 0 & else \end{cases}$$

כאשר הפונקציה פועלת על וקטור $x \in \mathbb{R}^n$ היא מבצעת את הפעולה הזו על כל קוארדינטה בנפרד.

- **פונקציית Softmax**

$$\forall x \in \mathbb{R}^n \qquad Softmax(x) = \frac{1}{\sum_{k=1}^{n} e^{x_k}} \begin{bmatrix} e^{x_1} \\ e^{x_2} \\ e^{x_3} \\ \vdots \\ e^{x_n} \end{bmatrix} \in \mathbb{R}^n$$

x_k - הקוארדינטה ה־k של $x \in \mathbb{R}^n$

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

הפונקציה לוקחת וקטור $x \in \mathbb{R}^n$ וממירה אותו אל וקטור התפלגות (איברים חיוביים שסכומם הוא 1) באופן שתואם את הפלט הסופי של הרשת שלנו.

2 מימוש הרשת

2.1 שימוש ב־float ודיוק הרשת

טיפוס המידע איתו נעבוד יהיה float (32-bit). לדוגמה, איברי המטריצה שנממש יהיו מטיפוס זה. על אף שגודל float בבתים תלוי במערכת ההפעלה, ניתן יהיה להניח לאורך התרגיל שגודלו של float הוא 4 בתים.

מאופן המימוש של float במער, 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:

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

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	אקטיבציה
1 (כניסה)	$(W_1 \in M_{128\times784})$	$T_1 : \mathbb{R}^{784} \to \mathbb{R}^{128}$	$b_1 \in \mathbb{R}^{128}$	$Relu$
2	$(W_2 \in M_{64\times128})$	$T_2 : \mathbb{R}^{128} \to \mathbb{R}^{64}$	$b_2 \in \mathbb{R}^{64}$	$Relu$
3	$(W_3 \in M_{20\times64})$	$T_3 : \mathbb{R}^{64} \to \mathbb{R}^{20}$	$b_3 \in \mathbb{R}^{20}$	$Relu$
4 (מוצא)	$(W_4 \in M_{10\times20})$	$T_4 : \mathbb{R}^{20} \to \mathbb{R}^{10}$	$b_4 \in \mathbb{R}^{10}$	$Softmax$

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

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

- 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 בגודל 28×28 של פיקסלים בגווני אפור (Grayscale), וכל מספר במטריצה הוא ערך בין 0 ל־1
- המטריצה המייצגת את התמונה הינה $M_{28\times28}([0,1])$
- לנוחיותכם, בקבצי העזר נמצא הקובץ plot_img.py אשר מקבל כקלט נתיב לתמונה ומציג אותה בחלון חדש
- וקטור הקלט שיישלח לרשת יהיה וקטור עם עמודה אחת ועם $28 \times 28 = 748$ שורות (ההמרה תתבצע לפי סעיף 2.6)

2.2.3 וקטור הפלט

- וקטור הפלט מהשכבה האחרונה הוא וקטור התפלגות באורך 10
- כל אינדקס בווקטור 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 שיוויון, נחזיר את האינדקס הנמוך מבין השניים

$Value$	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

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

2.3 טעינת מטריצה מקובץ בינארי

המשקולות וההיסטים של הרשת, וגם התמונות שהן הקלט של הרשת, ניתנות כקבצים. אנחנו נצטרך לקרוא אותם בצורה בינארית ולטעון אותם אל אובייקט של מטריצה באמצעות הפונקציה read_binary_file (חתימתה של הפונקציה אמורה להיות כמעט זהה לזו של אופרטור הקריאה מקובץ לאובייקט $<<$ שראינו בכיתה, למעט השם שלה).

- כל קובץ בינארי מכיל רצף של בתים
- כל 4 בתים רצופים הינם $float$ אחד (זכרו: הגודל תלוי במערכת ההפעלה. בתרגיל הזה ניתן להניח שזהו הגודל של $float$ שאיתו נתעסק)
- המיפוי של אינדקס במערך אל אינדקס במטריצה ייעשה לפי סעיף 2.5

- על הקריאה למלא את כל איברי המטריצה, אחרת מדובר בשגיאה

הנה דוגמה לטעינת קובץ המשקולות עבור השכבה הראשונה:

- W_1 היא מטריצה בגודל 784×128
- לכן, מספר הערכים שעלינו לקרוא מהמערך הוא $784 \times 128 = 100352$
- כל ערך הוא מסוג float בגודל 4 בתים
- לכן, מספר **הבתים** שעלינו לקרוא מהקובץ הוא $100352 \times 4 = 401408$

2.4 הדפסת תמונה ממטריצה

נרצה להדפיס את התמונה שמיוצגת במטריצה A באמצעות אופרטור <<. נעשה זאת לפי הפסאודו־קוד הבא:

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

2.5 מהלך ריצת התוכנית

שימו לב: חלק זה ממומש עבורכם בקובץ main.cpp הניתן לכם עם קבצי התרגיל

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 אל מטריצה
2. תכניס את המטריצה אל הרשת כקלט
3. כאשר התקבלה תוצאה, התוכנה תדפיס את התמונה, את הספרה שהרשת זיהתה ובאיזו הסתברות
4. התוכנה תמתין לקלט חדש

כאשר נזין לתוכנה q - התוכנה תצא עם קוד 0.

2.6 אינדקס יחיד לזכרון דו־מימדי

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

$$A(i,j) == A[i \cdot \text{rowsize} + j]$$

- i - אינדקס השורה
- j - אינדקס העמודה
- $rowsize$ - אורך השורה (מספר העמודות)

לדוגמה, תהי $A \in M_{3\times 4}$, כלומר בעלת 3 שורות ו־4 עמודות. מתקיים: $A(2,1) == A[2 \cdot 4 + 1] == A[9]$
וודאו שאתם מבינים מדוע מיפוי זה הינו חד־חד־ערכי ועל.

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
- **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 צריך להשתמש ב־const, האם 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

במחלקה זו נממש את המטריצות הדרושות לריצת התוכנית (גם וקטור הינו מטריצה, בעלת עמודה אחת ו־n שורות). נזכיר כי איברי המטריצה יהיו מטיפוס float.

	Description	**Comments**
Constructor	Matrix(int rows, int cols)	Constructs Matrix rows × cols Inits all elements to 0
Default c'tor	Matrix()	Constructs 1×1 Matrix Inits the single element to 0
Copy c'tor	Matrix(Matrix)	Constructs 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 cols as int

	transpose()	Transforms a matrix into its transpose matrix.
		$(A.transpose())_{ij} = A_{ji}$
		Supports function calling concatenation.
		e.g.
		Matrix a(5,4), b(4,5);
		a.transpose(); // a.get_rows == 4, a.get_cols == 5
		b.transpose().transpose(); // b is same as before

	vectorize()	Transforms a matrix into a column vector. Supports function calling concatenation. e.g.: Matrix m(5,4); ... m.vectorize(); m.get_cols() == 1 m.get_rows() == 20
	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 dot product of this matrix and another matrix m: $\forall i, j : (A.dot(B))_{ij} = A_{ij} \cdot B_{ij}$
	norm()	Returns the Frobenius norm of the given matrix: $A.norm() = \sqrt{\sum_{i,j} A_{ij}^2}$
	read_binary_file(istream, Matrix)	Fills matrix elements as per section 2.3. Has to read input stream fully, otherwise it's an error (don't trust the user to validate it).

	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
+=	Matrix addition accumulation	Matrix a, b; → a += b
()	Parenthesis indexing	For i,j indices, Matrix m: m(i,j) will return the i,j element in the matrix
[]	Brackets indexing	For i index, Matrix m: m[i] will return the i'th element (section 3.2)
<<	Output stream	Pretty export of matrix as per section 2.4

3.2 המחלקה Activation

במחלקה זו נגדיר את פעולת פונקציית האקטיבציה.

	Description	**Comments**
Constructor	Activation(ActivationType act_type)	Accepts an ActivationType enum with one of two legal values: RELU/SOFTMAX. Defines this instance's activation accordingly
	Methods	
Getter	get_activation_type()	Returns this activation's type (ReLU/Softmax)
	Operators	
()	Parenthesis	Applies activation function on input. Does not change input. Matrix output = act(input)

3.3 המחלקה Dense

במחלקה זו נגדיר שכבה שכבה ברשת הנוירונים.

	Description	Comments
Constructor	Dense(w, bias, ActivationType)	Inits a new layer with given parameters. C'tor accepts 2 matrices and activation type
	Methods	
Getter	get_weights()	Returns the weights of this layer forbids modification
Getter	get_bias()	Returns the bias of this layer forbids modification
Getter	get_activation()	Returns the activation function of this layer forbids modification
	Operators	
()	Parenthesis	Applies the layer on input and returns output matrix Layers operate as per section 3.1 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 ממשת ספציפית את הרשת המתוארת במסמך זה (סעיף 3.1). נקודה למחשבה: מה היה נדרש לממש במחלקה זו על מנת לתמוך ברשת עם מספר שכבות וגודל שכבות הניתן בזמן ריצה?
נשים לב כי struct Digit מומש עבורכם בקבצים שקיבלתם.

	Description	Comments
Constructor	MlpNetwork(weights[], biases[])	Accepts 2 arrays, 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 digit struct MlpNetwork m(...); Digit output = m(img);

4 טיפול בשגיאות

בתרגיל זה לא נדרוש מכם להשתמש ב־exceptions. במקרה של שגיאה:

- נדפיס הודעת שגיאה **אינפורמטיבית** המתחילה בפתיח "Error:" ל־std::cerr, ובסיומה נדפיס ירידת שורה.

- נצא מהתוכנית עם קוד 1 (למעט אם נאמר מפורשות אחרת במסמך זה).
- במקרה שכזה, בתרגיל זה אינכם נדרשים לשחרר את הזכרון.

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 2.5

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 שבאתר הקורס.
- **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.
- **טסטים:** בתרגיל זה נחזור למתכונת של שיתוף טסטים בין סטודנטים.
- **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 מחייבת 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.
- **סקריפט Pre-submission:** ודאו כי התרגיל שלכם עובר את ה־Pre-submission Script **ללא שגיאות או אזהרות.**

בהצלחה!