דו"ח תרגיל 3

מגישים : אלינוי עמר 318532132

אלעד וויצנבליט 315944611

- שמות קבצי הריצה של הנתונים ל-train ול-test :

  לקבצי הtrain (בחלוקה שלנו 80% מהדאטא) קוראים :
  nn1.txt ,nn0.txt

  וקבצי הtest (בחלוקה שלנו 20% הנותרים) בהתאמה :
  nn_test1.txt ,nn_test0.txt

  כלומר buildnet.py קוראת משני קבצים, ושניהם צריכים להמצא בתיקייה של buildnet.py – אחד nn1.txt והשני nn_test1.txt עבור מודל 1, בעוד עבור מודל 0 זה יהיה nn0.txt ו- nn_test0.txt.

- מריצים את אותו אלגוריתם לnn0 ולnn1 מכיוון ועשינו את הבונוס.
  כלומר, אנחנו מגישים קובץ buildnet.py יחיד וקובץ runnet.py יחיד.

- לקובץ הפלט של runnet.py קוראים labels0.txt או labels1.txt בהתאם למודל שהורץ.
  ובנוסף במהלך הרצת התוכנית הוא מצפה לקובץ בשם testnet0.txt או testnet1.txt בהתאם למודל וגם wnet0.txt או wnet1.txt כפי שהוגדר בהוראות התרגיל.

- **<u>הרצה:</u>** יש ללחוץ דאבל קליק על הקובץ exe המבוקש. וכאשר זה יפתח זה ישאל אותך איזה מודל ברצונך להריץ. יש להכניס 0 או 1 בהתאם למודל המבוקש.

```
insert which model to run (select 1 or 0)
0
```

דוגמה להרצת מודל 0

נתונים על אימון הרשת

1) קראנו את הדאטא מהקובץ nn0.txt\nn1.txt כדאטא לאימון. כלומר, בהרצה שלנו 80% שומשו לtrain בעוד 20% , שקראנו מהקובץ nn_test1.txt/ nn_test0.txt, שימשו למבחן.

```
split_test_index = round(len(X) * 0.8)
build_inputs = np.array(X[0:split_test_index])
test_inputs = np.array(X[split_test_index:])
build_labels = np.array([Y[:split_test_index]])
test_labels = np.array([Y[split_test_index:]])
```

את הדאטא חילקנו לקלט ופלט רצוי. כמו כן, שמרנו אותם במערכים מסוג np.

מערך ראשון ל-input של הtrain

מערך שני ל-input של הtest

מערך שלישי ה-labels של הtrain

מערך רביעי ה- labels של הtest.

2) את ארכיטקטורת הרשת קבענו על ידי אלגוריתם גנטי.

הפרמטרים ההתחלתיים שהכנסנו לאלגוריתם הגנטי הם :

גודל אוכלוסייה (רשתות נוירונים) : 200

כמות דורות מקסימלית : 300

סף הצלחה שממנו נעצור את התוכנית : 99%

נתוני אימון (input,label) שעל פיו חישבנו את ציון ה-fitness.

```
agent = ga.execute(200,300,0.99,build_inputs,build_labels,network)
```

מבנה הרשת (יישום של הבונוס):

כתחילה ניצור 200 רשתות נוירונים באופן רנדומלי.

אופן היצירה של כל אחת מ-200 הרשתות הוא :

ראשית, מגרילים מספר רנדומלי בין 2-1 שהוא יהיה מספר השכבות החבויות במודל (שכבות הקלט והפלט לא כלולות במספר זה).

לאחר מכן, נוסיף את השכבה הראשונה, שכבת הקלט שהיא תמיד בגודל 16 מכיוון והקלט הוא 16 ביטים, ונגריל את גודל הפלט של השכבה (מספר רנדומלי בין 1 ל-10).

לכל hidden layer אנחנו מגרילים מספר בין 10-1 שהוא יהיה גודל פלט השכבה (גודל הקלט הוא כגודל הפלט של השכבה הקודמת).

לאחר כל שכבה הוספנו את פונקציית האקטיבציה sigmoid על מנת לשבור את הלינאריות.

בסוף, נוסיף את שכבת הפלט, שגודל הפלט שלה הוא תמיד 1 (הלייבל שהיא נותנת לדוגמה שעליה הרשת הורצה) עם sigmoid. מכיוון ואנו רוצים פלט של 1 או 0 אנחנו תמיד נבצע עיגול של התוצאה של הרשת כלפי הערך הקרוב יותר (0 או 1).

כל המשקלים ההתחלתיים נקבעים בצורה רנדומלית.

אופן פעולת האלגוריתם הגנטי:

כעת האלגוריתם הגנטי מחפש את הרשת הכי טובה בצורה הבאה :

- בתחילת כל דור, עבור כל מודל נחשב את ציון ה-fitness שלו על ידי הרצת כל הקלטים ברשת והשוואת הפלטים למול הפלטים הרצויים. לצורך חישוב הfitness השתמשנו בפונקצית הMSE loss, והערך עצמו מייצג את אחוז הדגימות שצדקנו עליהן.

- לאחר מכן, כשאנו יודעים את הfitness של כל רשת, האלגוריתם לוקח את 40% הרשתות הכי טובות, ומעביר אותם לדור הבא כמו שהן.

- את ה60% הנותרים אנו נבנה באמצעות crossover שעובד כך :
  א. מתוך האוכלוסייה של ה40% שאיתה המשכנו לדור הבא, בוחרים באופן רנדומלי 2 רשתות כ-parents.

ב. מבנה הרשת בעלת הfitness הגבוה יותר מתוך 2 ההורים, היא זו שתשמש כארכיטקטורה של הרשת.
ג. כעת עלינו להחליט על המשקלים. עבור כל שכבה נבחר רנדומלית אינדקס i.
עבור child1 ניקח את המשקולות מparent1 עבור אינדקס 0 ועד ל-i. ומparent2 מi+1 ואילך.
עבור child2 נעשה הפוך. כלומר, ניקח את המשקולות מ-parent2 עבור אינדקס 0 ועד ל-i. ומ-parent1 מi+1 ואילך.

במידה וכמות המשקולות חורגת מparent1\parent2 (מכיוון והארכיטקטורות שונות אז ייתכן כי לכל אחד יש כמות שונה של נוירונים בכל שכבה). ניקח את המשקולות הקיימות מהparent שאכן יש לו את זה (כלומר הparent שבו השתמשנו לארכיטקטורה)

- עבור כ- 50% מהאוכלוסייה מבצעים MUTATE (מגרילים מספר בין 0 ל-1 ואם הוא גדול מ0.5 מבצעים MUTATE, 0.5 זה היפר-פרמטר בשם MUTATE_RATE, שהביא תוצאות טובות).

הMUTATE מבוצע בצורה הבאה:
מגרילים מהתפלגות יוניפורמית מספר בין 0 ל-1.
אם הוא קטן מ0.8 (היפר-פרמטר בשם RESET_RATE):
בוחרים משקולת מתוך הרשת באופן רנדומלי, ומשנים את משקלה לערך שהוגרל מהתפלגות סטנדרטית.

אחרת, אם הערך גדול מ0.8:
מאפסים רנדומלית 20% מהמשקולות של הרשת. (כלומר 20% מהמשקולות מקבלים ערך 0).

וכך האלגוריתם פועל למשך 300 דורות. במידה ורשת כלשהי הגיעה לדיוק של 99% הוא מדווח לנו באיזה דור זה קרה.

- לאחר סיום ריצת האלגוריתם הגנטי אנחנו מדפיסים את ההצלחה של הרשת הטובה ביותר מתוך הדור האחרון, ואנחנו שומרים את מבנה הרשת הזו לתוך קובץ בשם wnet0.txt או wnet1.txt בהתאם לקובץ שהורץ בהתחלה (nn0.txt ו nn1.txt בהתאמה).

פורמט שמירת הנתונים הוא:
שורה ראשונה היא מספר השכבות שיש ברשת.

לאחר מכן מספר השורות במטריצה שבשכבה הראשונה.
לאחר מכן, המשקולות של כל שורה במטריצה נשמרות בשורה נפרדת בקובץ.

לאחר מכן מספר השורות במטריצה בשכבה השנייה
לאחר מכן המשקולות בכל שורה במטריצה (שמחברות בין השכבה השנייה לשלישית) נשמרות בשורה נפרדת.

וכך הלאה...

<u>אופן טיפול בהתכנסות מוקדמת:</u>
בכל ריצה עשירית (כלומר ריצה 10,20,30...)

הוא בודק האם הfitness הטוב ביותר בריצה הנוכחית השתפר בלפחות 0.0001 מהfitness הכי טוב שהיו ב10 ריצות הקודמות.

במידה ולא:

מוסיפים לmutation_rate 0.25. כלומר מעלים את גודל האוכלוסייה שמבוצעת עליה mutate ב25% (במידה ולהוסיף 0.25 גדול מ-1 הערך יהיה 1, כלומר על כל האוכלוסייה מבצעים mutate).

באופן דומה, לselection_percentage (אחוז הרשתות שממשיכות כמו שהן לדור הבא) מורידים 0.1,כלומר גודל האוכלוסייה של הדור הבא יורכב מ10% פחות רשתות שממשיכות כמו שהן ובמקומן יבוצע CROSSOVER, אם הערך קטן מ0.2 לאחר ההפחתה של 0.1 - משאירים את זה 0.2.

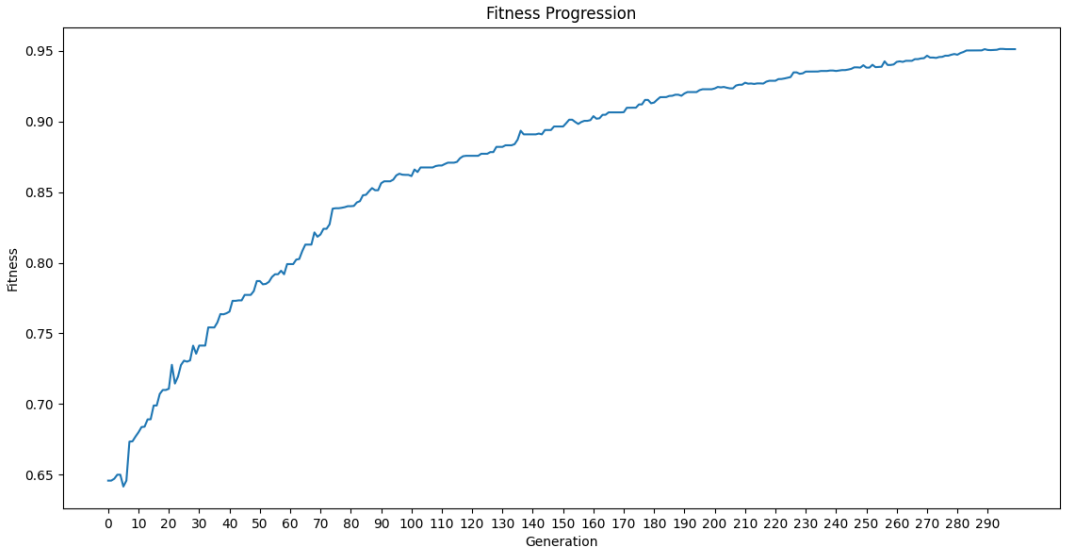
ביצועי התוכנית

nn0:

```
[0.9514375]
Test accuracy is 93.825
Model was saved in 'wnet0.txt' file.
```

דיוק בtrain 95.14%
דיוק בtest 93.825%

גרף fitness הכי גבוה – דור עבור הtrain:

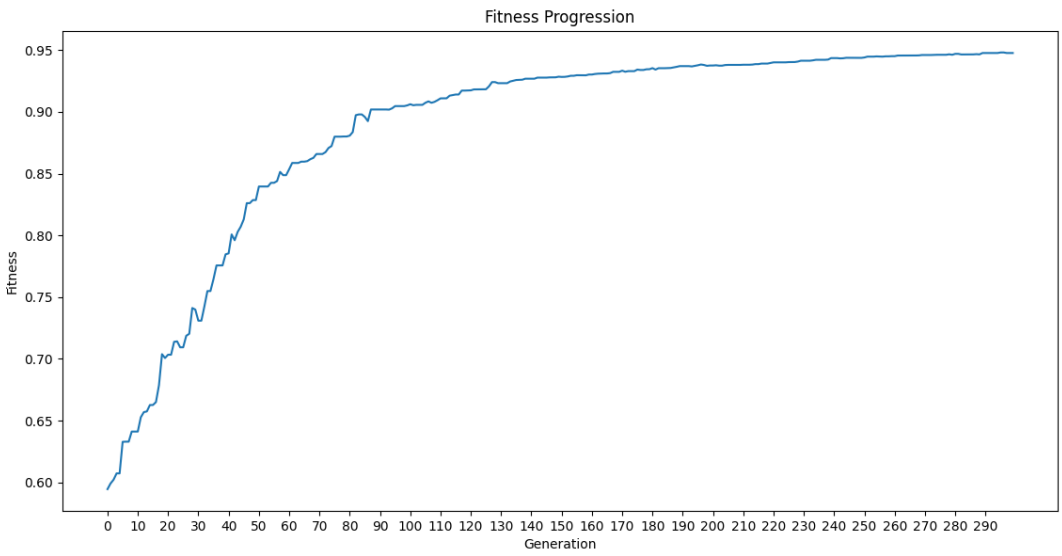


nn1:

```
[0.9479375]
Test accuracy is 94.75
Model was saved in 'wnet1.txt' file.
```

דיוק בtrain 94.79%
דיוק בtest 94.75%

גרף fitness הכי גבוה – דור עבור הtrain:



חוקיות משוערת

בקובץ nn0 – לפחות 8 אחדות בסטרינג (כולל 8) מקבל label 1 אחרת label 0

בקובץ nn1 – לפחות 8 אחדות בסטרינג (כולל 8) מקבל label 0 אחרת label 1