בס"ד

ביולוגיה חישובית – תרגיל 3

גיא עדני – 208642884
חן ברסנו – 319004339

הוראות הפעלה:

ראשית, מצורפים שני קבצים split0.py ו-split1.py. כל אחד מהקבצים כאשר מופעל, לוקח את הקבצים שנקראים nn1,nn0 בהתאמה ומייצא שני קבצים testset0.txt,trainset0.txt או testset1.txt, trainset1.txt.

הקבצים מחלקים את ה-data לtest set בגודל 20% מה-data ושאר ה-80% הם train set, החלוקה מתבצעת רנדומלית. לחיצה כפולה על הקובץ exe תריץ אותו.

לכן, **על מנת להריץ את buildnet0 ו-buildnet1** צריך ש**הקלט יהיה בתיקייה** וייקרא:

- **trainset0.txt, testset0.txt** עבור **buildnet0**
- **trainset1.txt, testset1.txt** עבור **buildnet1**

מומלץ להריץ את קבצי ה-exe של buildnet באמצעות קליק ימני על הקובץ ו-run as administrator , אחרת מערכת ההפעלה עלולה לא לתת לתוכנית מספיק משאבים והריצה תהיה איטית בצורה לא סבירה.

כדי להפעיל את buildnet0 ו-buildnet1 צריך ללחוץ לחיצה כפולה על הקובץ exe.

הקבצים runnet0 ו-runnet1 מורצים על ידי לחיצה כפולה על הקובץ exe.
על מנת שיורצו כמו שצריך **יש לוודא שהקבצים הבאים נמצאים בתיקייה:**

- **wnet0.txt, testnet0.txt** עבור **runnet0.**
- **wnet1.txt, testnet1.txt** עבור **runnet1.**

יש לוודא שבקבצי הקלט מסוג txt. אין רווחים מיותרים ויש שורה ריקה בסוף הקובץ.

הקדמה:

בתרגיל זה התבקשנו לממש אלגוריתם גנטי לאימון רשת נוירונים במטרה לזהות חוקיות / תבניות של רצפים בינאריים. האימון התבצע על קובץ המכיל 20,000 מחרוזות בינאריות באורך 16 ספרות, כאשר אחרי כל מחרוזת מופיעה הספרה 0 או 1 – האומרת האם המחרוזת מתאימה לחוקיות מסוימת.

בעיקרון buildnet0 ו-buildnet1 שלנו זהים, מצאנו שעבור שני סוגי החוקיות המודל הגנטי שלנו מצליח להגיע לאחוזי הצלחה יפים על ה-test set ולכן החלטנו להשאיר אותו דבר.

ייצוג הדאטה ומבנה הרשת:

את הדאטה ייצגנו ע"י רשימה של טאפלים – כל טאפל מכיל סטרינג (שהפכנו למערך NumPy של int) ואת הלייבל המתאים לו –

```
# each tuple now contains an encoded string and its corresponding label. (string -> int)
def preprocess_data(data):
    processed_data = [(np.array([int(bit) for bit in string]), label) for string, label in data]
    return processed_data
```

הרשת שלנו היא fully connected כלומר כל נוירון בשכבה אחת מחובר לנוירון בשכבה הבאה, ומורכבת מ-3 שכבות:

1. Input layer – בגודל של 16 נוירונים (hyperparameter).
2. Hidden layer – בגודל של 4 נוירונים (hyperparameter).
3. Output layer – בגודל של נוירון אחד (סיווג בינארי).

המשקלים שמחברים בין השכבות מאותחלים רנדומלית בזמן יצירה של אובייקטים מסוג NeuralNetwork או מתקבלים כארגומנט (עבור runnet):

```
def __init__(self, weights1=None, weights2=None):
    if weights1 is None and weights2 is None:
        self.weights1 = np.random.randn(INPUT_SIZE, HIDDEN_SIZE) - 0.5
        self.weights2 = np.random.randn(HIDDEN_SIZE, OUTPUT_SIZE) - 0.5
    else:
        self.weights1 = weights1
        self.weights2 = weights2
```

ה-Hidden layer מבצעת טרנספורמציה לינארית על הקלט בעזרת משקלים ומפעילה פונקציית אקטיבציה – sigmoid:

```
def forward(self, x):
    hidden = np.dot(x, self.weights1)  # Shape: (batch_size, hidden_size)
    hidden_activation = self.sigmoid(hidden)  # Shape: (batch_size, hidden_size)
    output = np.dot(hidden_activation, self.weights2)  # Shape: (batch_size, output_size)
    output_activation = self.sigmoid(output)  # Shape: (batch_size, output_size)
    return output_activation
```

<u>מאפייני האלגוריתם הגנטי</u>:

התוכניות buildent0, buildnet1 מקבלות 2 קבצים – קובץ למידה (80% מהמחרוזות) וקובץ מבחן (20% מהמחרוזות).

אתחלנו את האוכלוסיה ע"י יצירת רשתות עם משקלים רנדומליים כמספר POPULATION_SIZE לפי מבנה רשת שנקבע מראש. לאחר מכן, הערכנו את ציון ה-fitness של כל אחת מהרשתות באוכלוסיה בעזרת הפונקציה evaluate_fitness. הפונקציה עוברת על כל מחרוזת בדאטה ומעבירה אותה לאינדיבידואל (הרשת) ב-feed forward. לאחר מכן, בדקנו האם התוצאה הייתה גדולה מ-0.5 או קטנה, ובהתאם לכך קבענו את predicted_label. אם הלייבל שחזינו אכן היה זהה ללייבל האמיתי – הגדלנו את המשתנה correct_predictions ב-1, והוא סה"כ מה שקובע לנו את ציון ה-fitness.

מתוך האוכלוסיה, בחרנו 60 פרטים שיעברו מוטציה ולאחר מכן בחרנו בצורה רנדומלית 2 פרטים שעליהם נרצה לעשות crossover. את הבחירה עשינו כך שאליטה של האוכלוסיה, כלומר הפרטים שמהווים 20% מהאוכלוסייה, בעלי ה-fitness הגבוה ביותר יש להם פי 2 סיכוי להיבחר.

המוטציה מתבצעת באופן הבא: עבור כל מטריצת משקלים של הרשת נחשב mask. המסכה הזו בעצם קובעת לפי ה-MUTATION_RATE אילו משקלים יעברו מוטציה. אם הערך שהוגרל קטן מה-MUTATION_RATE המשקל המתאים יעבור מוטציה, אחרת לא. לאחר מכן, ניצור מערך של ערכים רנדומליים בין מינוס MUTATION_RATIO ל-MUTATION RATIO. ערכים אלו מייצגים את ה-"magnitude" של המוטציה.

את ה-crossover מימשנו כך: לקחנו את ממוצע המשקלים של weights1 מ-2 ההורים ואת ממוצע המשקלים של weights2 מ-2 ההורים וכך יצרנו את weights1,weights2 של הילד. בנוסף, כדי ליצור גיוון – בחרנו crossover point שעד אליה "החומר הגנטי" יהיה של parent1 וממנה והלאה של parent2.

<u>טיפול בהתכנסות מוקדמת</u>:

על מנת לטפל בהתכנסות מוקדמת, בדקנו האם במשך 10 דורות לא מתבצע שינוי ב-score של הרשת בעלת ה-score הגבוה ביותר עד כדי אפסילון. אם אכן האלגוריתם הגיע להתכנסות מוקדמת, הוא פשוט מפסיק ויש צורך להריץ מחדש.

יש לציין שמאוד לא סביר שיגיע להתכנסות מוקדמת, כאשר הגדרנו את אפסילון להיות 1e-6.

ביצועי התוכנית על קבוצת הלמידה ועל קבוצת המבחן:

1. nn0 – 98.32% על קבוצת הלמידה, 98.42% על קבוצת המבחן.
2. nn1 – 97.76% על קבוצת הלמידה, 97.65 על קבוצת המבחן.

תיאור החוקיות של 2 התבניות:

1. nn0 – אם מספר האחדות במחרוזת ≤ 8, אזי המחרוזת מסווגת כ-1. אחרת, 0.
2. nn1 – אם מספר האפסים במחרוזת ≤ 8, אזי המחרוזת מסווגת כ-1. אחרת, 0.

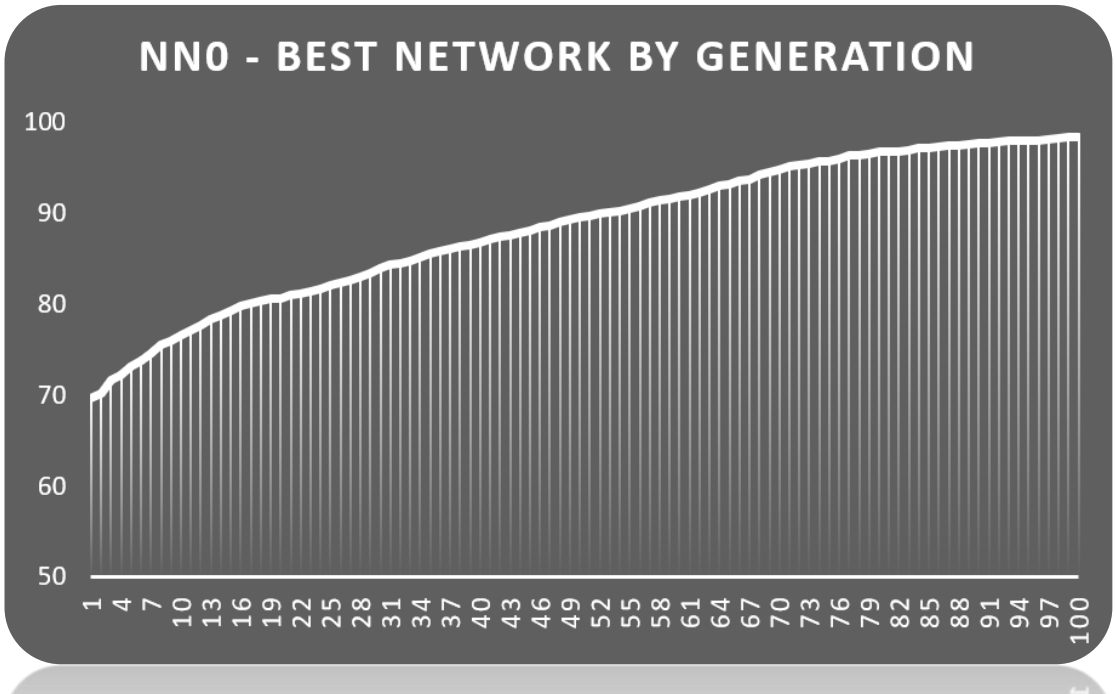
פרמטרים עבור הרשת האופטימלית של 2 התבניות:

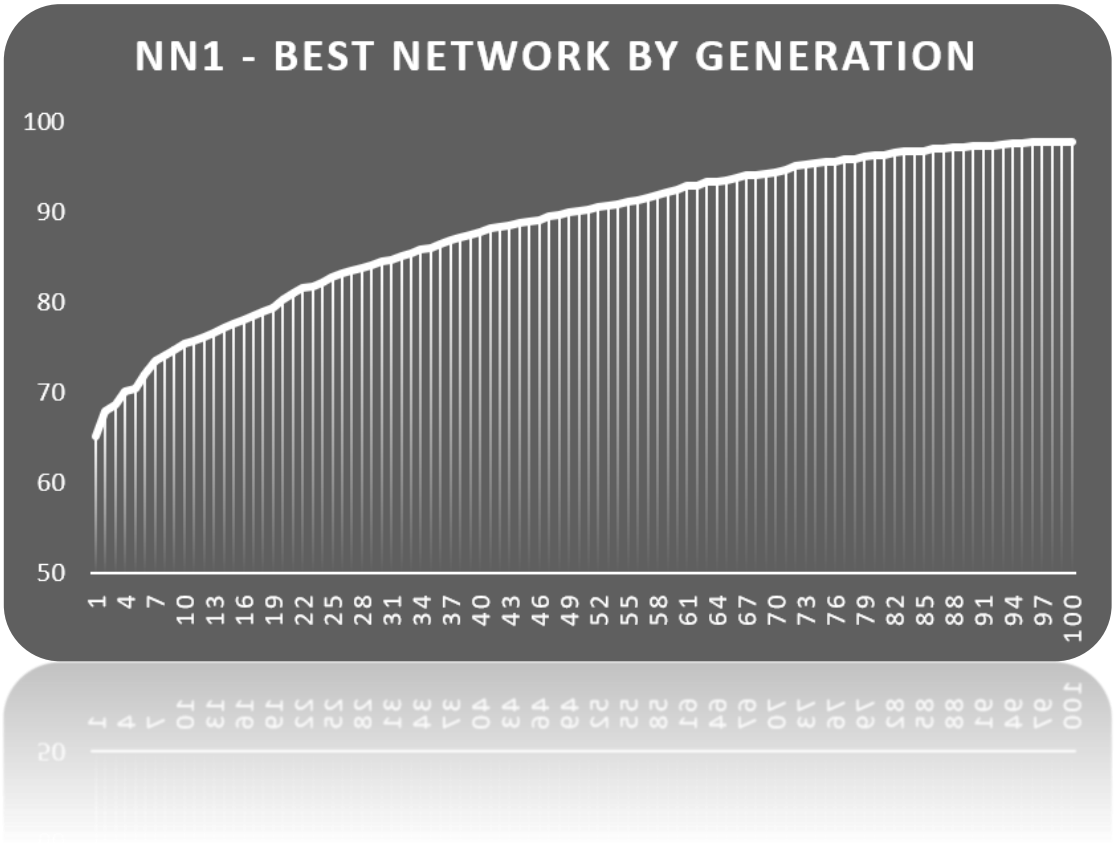
אלו הם הפרמטרים האופטימליים שמצאנו לאחר ששיחקנו הערכים –

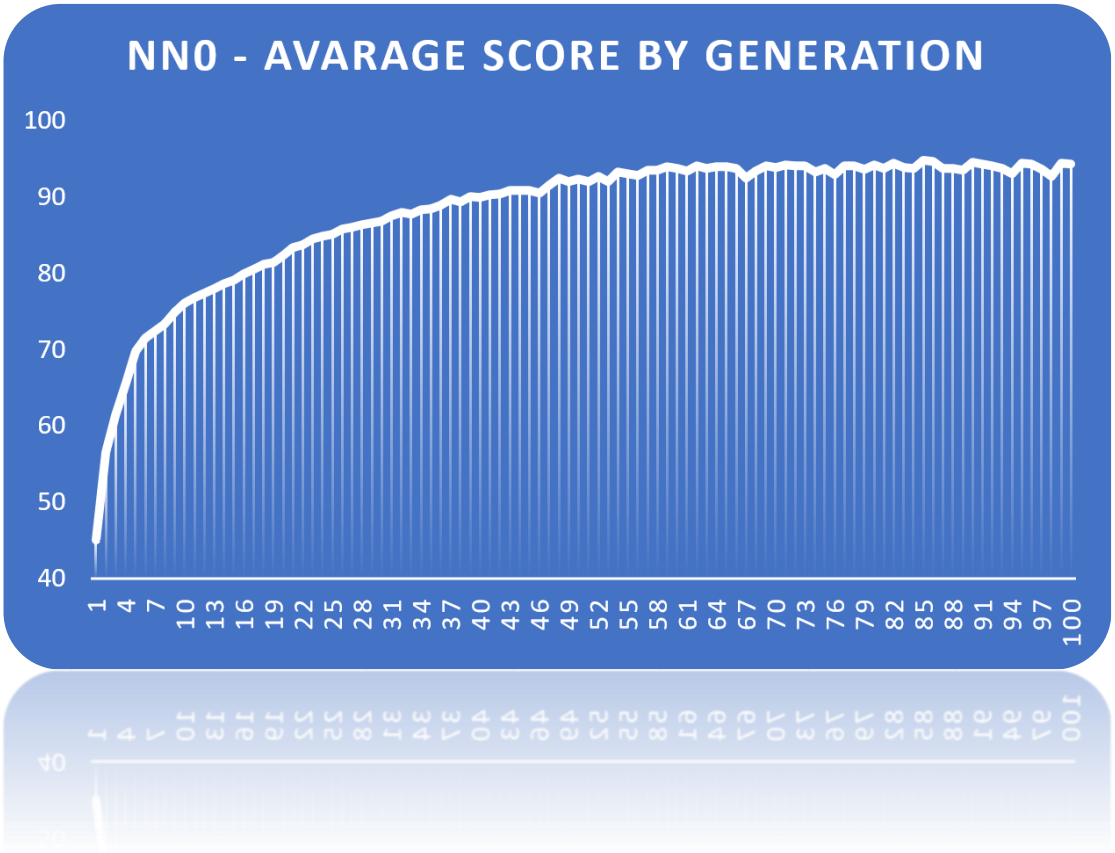
```
MAX_GENERATIONS = 100
POPULATION_SIZE = 120
MUTATION_RATE = 0.1
MUTATION_RATIO = 0.1
TOP_PERCENTAGE = 0.2
```

בגרפים הבאים ניתן לראות את תהליך הלמידה של buildnet0 ו-buildnet1 על הקבצים nn0, nn1 בהתאמה, כאשר הפרמטרים שנבחרו הם הפרמטרים של הרשתות האופטימליות שמצאנו.

ניתן לראות כי בשני המקרים של best score הרשת מתחילה מדיוק של 65%-70% ועולה בהדרגה, ואילו בדוגמה של ה-avarage score, הממוצע מתחיל מ45% ועולה בהדרגה. כמו כן, ניתן לראות שהוא לא מונוטוני כמו ה-best score ויכול לקבל נקודות מינימום מקומיות, כלומר גם לרדת מעט.




NN1 - BEST NETWORK BY GENERATION
100
90
80
70
60
50
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100


NN0 - AVARAGE SCORE BY GENERATION
100
90
80
70
60
50
40
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100