מטלה 3 ביולוגיה חישובית – זיהוי תבניות באמצעות רשת נוירונים

איתי אלקלאי 2060071110, אביאל זכריה 207456641 – תרגיל משולב

זיהוי התבניות

על מנת לזהות את התבניות ניסינו לנחש כל מיני לוגיקות שיכולות לאפיין פונקציה בוליאנית שמקבלת כקלט 16 פרמטרים בינאריים, ובהתאם לכתוב סקריפט פייתון שיאמת את זה:

```
# Read nn1.txt file
nn1 = open('nn1.txt', 'r').readlines()
nn1 = [i.split() for i in nn1 if i]

# Making it a binary list, and a Boolean output - ([1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0], 0)
nn1 = [(list(map(int, i)), bool(int(j))) for i, j in nn1]

# Print unmatched inputs
print([(i, o) for i, o in l1 if o != logic(i)])
```

כמובן שהנחנו שאין לנו Outliers, ובנוסף שהחוק הוא הגיוני וניתן להכללה בהנחה שאין לנו את כל מרחב הקלטים האפשרי בעולם (או לפחות מגוון מספיק כדי לזהות את הפונקציה בהכללתה)

nn1.txt

ניסינו המון אופרטורים בינאריים כמו AND, XOR, OR וכלום לא עבד, ניסינו לחלק את זה לשני חלקים של שמונה ועל זה להפעיל פונקציות וזה גם לא עבד וכו'. לבסוף ניסינו להדפיס hitmap של מספר האחדות אל מול התוצאה הרצויה ושמנו לב לתופעה הבאה:

```
print(set([(sum(i), j) for i, j in nn1]))

{(1, True),
 (2, True),
 (3, True),
 (4, True),
 (5, True),
 (6, True),
 (7, True),
 (8, False),
 (9, False),
 (10, False),
 (11, False),
 (12, False),
 (13, False),
 (14, False),
 (15, False)}
```

ראשית ניתן לראות שאין לנו סתירה – כלומר אין סכום אחדות שנותן לנו פעם אחת שייכות ללוגיקה ובפעם השנייה לא. בנוסף, חסר לנו data על סכום 0 וסכום 16 אך סביר להניח שהם ממשיכים את הלוגיקה הברורה ולא מופיעים כיוון שיש sample יחיד שמתאים לכל תוצאה כזאת. **לסיכום, הפונקציה logic שלנו היא האם מספר האחדות קטן ממש מ-8.**

כמובן שיש רשת נוירונים (ואפילו פתרון לינארי) שמוצא את החוקיות ללא שכבה פנימית (וכך גם עשינו).

nn0.txt

לאחר מכן ביצענו בדיוק את אותו ה-hitmap ל-nn0.txt וקיבלנו תוצאה אחרת מעניינת:

```
set([(sum(i), j) for i, j in nn0])
```

```
{(1, 0),
 (2, 0),
 (3, 0),
 (4, 0),
 (5, 0),
 (6, 0),
 (7, 0),
 (8, 1),
 (9, 1),
 (10, 1),
 (11, 1),
 (12, 1),
 (13, 0),
 (14, 0),
 (15, 0),
 (16, 0)}
```

ראשית ניתן לראות שאין לנו סתירה – כלומר אין סכום אחדות שנותן לנו פעם אחת שייכות ללוגיקה ובפעם השנייה לא. בנוסף, חסר לנו data על סכום 0 וסכום 16 אך סביר להניח שהם ממשיכים את הלוגיקה הברורה ולא מופיעים כיוון שיש sample יחיד שמתאים לכל תוצאה כזאת. **לסיכום, נראה שמספר האחדות צריך להיות בין 8 ל-12 כולל כדי שהפונקציה תתאמת.**

כמובן שיש רשת נוירונים שמחשבת את זה ומוצאת את החוקיות (פתרון לינארי), אך היא דורשת שכבה פנימית (שני תנאים על הקלט), אך גם שכבה אחת תספק פתרון מקורב שיכול להיות מספק כיוון שמרחב האופציות הגדול מ-12 די קטן ביחס למרחב הקלט.

הוראות הרצה

כאמור, הקוד ממומש בשפת פייתון וניתן להריץ אותו בפשטות בעזרת תיעוד הנמצא בקובץ ה-README.md המצורף בתרגיל.

Usage

Splitting samples data into train set and test set.

```
python split_data.py -h  // help message
python split_data.py nn0.txt nn0_train.txt nn0_test.txt  // split to output files
```

Building the NN using GA.

```
python buildnet0.py -h  // help message
python buildnet0.py nn0_train.txt nn0_test.txt  // -> wnet0.txt
```

Calculating success rate of a given model.

```
python success_rate.py -h  // help message
python success_rate.py nn0_test.txt wnet0.txt  // print success rate
```

Evaluating the NN.

```
python runnet0.py -h  // help message
python runnet0.py wnet0.txt samples0.txt // -> results.txt
```

בנוסף לתכניות המבוקשות בתרגיל, מימשנו עוד 2 תכניות שמטרתן להפריד את ה-data לקבוצת אימון וקבוצת מבחן באותו פורמט ו-2 קבצים שונים (split_data.py) ותכנית נוספת המחשבת את ההצלחה של מודל נתון על קובץ data (success_rate.py)

מימוש הקוד

רשת הנוירונים

הקובץ הראשי שמממש את רשת הנוירונים שלנו הוא neural_net.py

- מבנה הרשת הנבחר – כאמור, בהתאם לניחוש שלנו לחוק שמתאר את ה-data שקיבלנו נראה שפתרון לינארי יהיה מספק ולכן בחרנו לא לבחור שכבה חבויה ברשת הנוירונים שלנו.
  כלומר הרשת מורכבת משכבת כניסה בעלת 16 כניסות ושכבת יציאה בעלת יציאה אחת בלבד.
- Layer threshold – נשים לב שהפכנו את זה בפועל ל-17 שכבות על מנת שנוכל לאמן את ה-threshold, וכאשר עשינו feed forward תמיד הוספנו ניורון עם ערך 1 לפי השכפלנו את המטריצות.
- פונקציית אקטיבציה – בחרנו ב-sigmoid כסטנדרט לקבלת הסתברות לתשובה ולא תשובה בינארית.
- פונקציית ה-fitness (או loss) – כל טעות בסיווג שילמה מחיר של המרחק מההסתברות האמיתית.
  המחיר של סך דוגמאות היה ממוצע מחירי הטעויות של הדגימות שניתנו.

האלגוריתם הגנטי

הקובץ הראשי בו כתובה הלוגיקה של האלגוריתם הגנטי הוא genetic_algo.py בפונקציה genetic_train_nn.

- גודל אוכלוסייה: 50
- מספר סיבובים מקסימלי: 1000
- גבול לעצירה: 98 אחוז הצלחה
- מספר דורות ללא שינוי: 50
- בחירת המובילים: 10 אחוז
- יצירת מוטציות: 40 אחוז
- יצירת crossover: 40 אחוז

יצירת מוטציות

מעבר על 10 אחוז מהשכבות של המודל ועבור כל שכבה להוסיף למשקלים שלה משקלים רנדומליים של ההתפלגות הנורמלית N(0, 0.05)

יצירת crossovers

על מנת ליצור הכלאה בין שני פתרונות קיימים הגרלנו זוג רנדומלי בכל פעם ויצרנו רשת חדשה שמשקלה הוא ממוצע המשקלים של שני הרשתות הקודמות.

ביצועים

כל אלגוריתם הרצנו בכ~30 תהליכים שונים במקביל לכמה שעות טובות, ולבסוף לקחנו את התוצאה הטובה ביותר.

את התוצאות חישבנו באמצעות התכנית success_rate.py על nn0.txt ועל nn1.txt עם המודלים המתאימים

- המודל wnet0.txt על הקלט nn0.txt השיג תוצאה כוללת 98.7 של אחוזי הצלחה.
- המודל wnet0.txt על הקלט nn1.txt השיג תוצאה כוללת של 98.2 אחוזי הצלחה.