דוח תרגיל 3

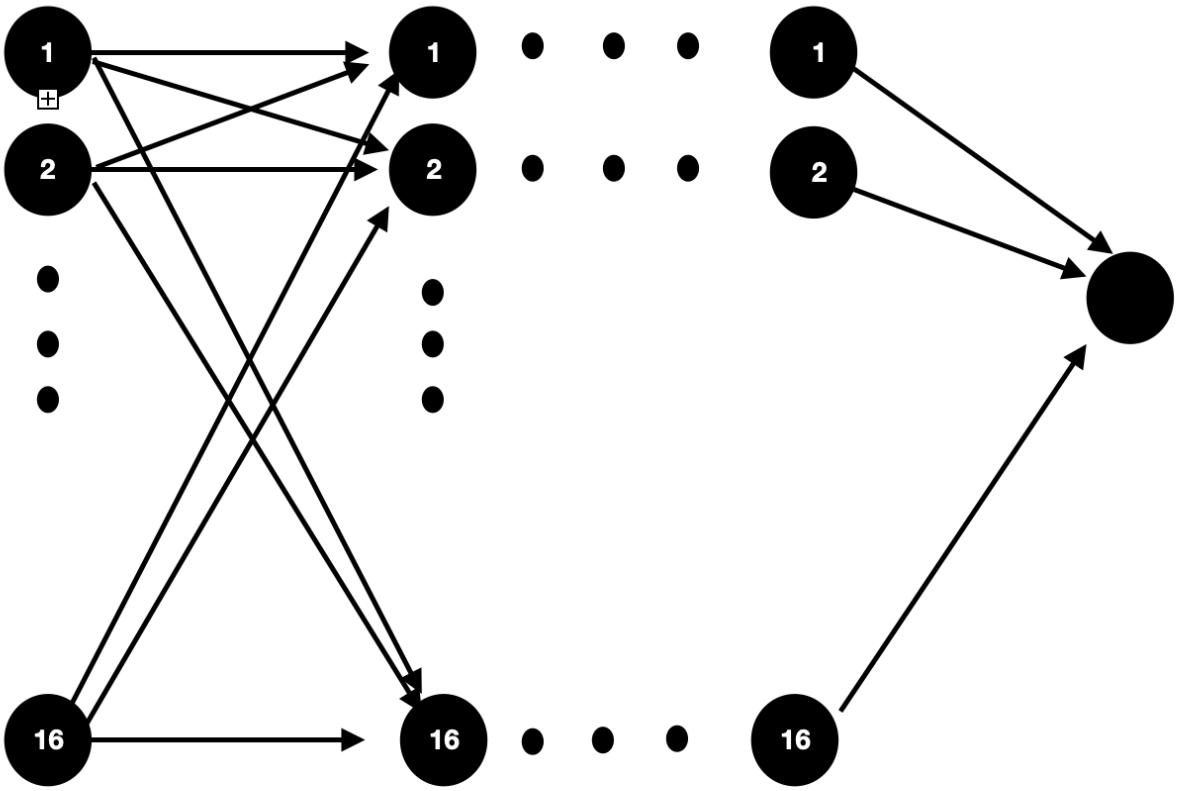
ביולוגיה חישובית
שם: יואב אליאב
ת״ז: 312498207

הקדמה

בדוח זה אני אציג את האלגוריתם והרשת שבניתי כדי לבנות רשת נוירונים, יש לנו 2 תכניות buildnet0 ו buildnet1 שניהם תוכניות שמותאמות לתבניות שונות והפלט שלהם מתאים לתכניות runnet0 ו runnet1 בהתאמה.

מבנה הרשת

מכיוון שהקלט היא מחרוזת בינארית באורך 16 אזי יצרתי רשת שכל שכבה כוללת 16 נוירונים, וכל נוירון מחובר לכל הנוירונים בשכבה הבאה. המחשה:



השוני בין buildnet0 ל buildnet1 הינו בעיקר בכמות השכבות ובהיפר פרמטרים, בשל העובדה ש buildnet1 מנסה לפתור תבנית קשה יותר יש לו יותר מספר שכבות. ואילו ל buildnet0 יש פחות שכן יש מימד של ביצועים שיפורט בהמשך.

מבנה קובץ המשקולות

קובץ המשקולות מוגדר לפי שורות כך שכל שורה מהווה חלק אחר בפרמטרים של הרשת

שורה 1 - מספר השכבות.

שורה 2 - הthreshold לקביעת הסיווג.

שאר השורות - מהוות את ערכי המשקלים, מכיוון שיש מספר קבוע של משקולות לכל שכבה אזי בקריאת הקובץ ניתן לדעת שה16 מספרים הראשונים מהווים 16 המשקולות של השכבה הראשונה וכך הלאה...

האלגוריתם

כדי לפתור את התרגיל השתמשתי באלגוריתם גנטי על מנת למצוא את המשקלים האופטימליים, התהליך של האלגוריתם הוא כלהלן:

1) יצירת אוכלוסיה התחלתית - התחלתי ביצירת מספר רשתות נוירונים בעל מבנה מסויים ואותם היפר פרמטרים, ההבדל בין הרשתות הוא במשקולות שלהן כאשר לכל רשת יש משקולות רנדומליות. בנוסף הוספתי לכל נוירון bias מסויים שגם הוא פרמטר נלמד וזאת על מנת שהמודל יהיה גמיש יותר.

2) חישוב ציון (Fitness Score) - על מנת לקבוע את הציון לכל רשת, עברתי על כל הדוגמאות לכל רשת, ובדקתי כמה דוגמאות מתוך כלל הדוגמאות היא הצליחה לתייג נכונה, הציון שלה היה רמת הדיוק שלה.

3) בחירת הורים - לאחר שקבעתי ציון לכל רשת, כעת בחרתי את ההורים שהגנים שלהם יועברו לדור הבא, לצורך כך בכל איטרציה בחרתי מתוך כלל האוכלוסיה קבוצה רנדומית בגודל מסויים x, כאשר x הינו היפר פרמטר, לאחר בניית הקבוצה לקחתי מהקבוצה את הרשת עם הציון הגבוה ביותר, הדבר הזה נותן יתרון לרשתות עם ציון גבוה אך מצד שני נותן סיכוי גם לרשתות עם ציון נמוך יותר להיבחר.

פונקציית הזיווג מופעלת כדי למצוא הורה יחיד, לכן כדי ליצור לדוגמא 100 צאצאצים יש צורך להפעיל את פונקציית הזיווג 200 פעמים שכן צריך 2 הורים לכל צאצא.

4) זיווג - לאחר בחירת 2 הורים, יש לעשות ביניהם זיווג (crossover) מכיוון שלשני ההורים יש אותו מבנה של רשת ורק ערך המשקולות ביניהם שונה, האלגוריתם עבר על כל משקולת ובצורה רנדומית בוחר אם המשקולת הזאת תגיע מהורה 1 או מהורה 2. בנוסף ישנו היפר פרמטר שיקבע מה הסיכוי האם נבחר ליצור זיווג או פשוט להעביר לדור הבא את הורה 1 או הורה 2, הסיבה לכך היא שלפעמים אני רוצה בסיכוי מסויים לשמר הורה עם משקולות טובות.

5) מוטציה - לאחר שרשת נבנתה מזיווג של 2 הורים, יש ליצור מוטציה, השאלה האם המוטציה תקרה היא היפר פרמטר שתלויה בערך y כלשהוא, פעולת המוטציה עוברת על כל משקולת ובסיכוי y תקבע האם לשנות את ערך המשקולת לערך רנדומי. הערך של y הוא יחסית נמוך וזאת בשל כמות המשקולות הרבה שיש והרצון להימנע ממצב של אלגוריתם חיפוש.

6) התפתחות האוכלוסיה - בשלב הזה אנחנו בעצם בלולאה עוברים על כל השלבים הקודמים, אנחנו מחשבים את הציון, בוחרים הורים, ויוצרים דור חדש. האלגוריתם פועל במשך n איטרציות כאשר מספר האיטרציוות הינו היפר פרמטר.

7) **תוצאה סופית -** כדי לא לאבד פתרונות טובים, לאורך כל האלגוריתם שמרתי את הרשת הטובה ביותר בכל הדורות, מה שהבטיח שבכל דור הציון של הרשת הטובה ביותר רק יעלה, לאחר שסיימתי לפתח את האוכלוסיה במשך מס׳ דורות מסויים הפיתרון שנשמר ויכתב הינו הפיתרון הטוב ביותר.

ביצועים

הנושא של ביצועים היווה שיקול מאוד בעייתי באלגוריתם, האלגוריתם של בניית הרשת הינו מאוד כבד, דבר היוצר שקלול תמורות בין הרצון להגיע למודל הטוב ביותר לבין הרצון ליצור מודל יעיל מבחינת מהירות, בנוסף היכולת למצוא את ההיפר פרמטרים הטובים ביותר נפגעה, שכן ריצת buildnet לוקחת זמן רב וקשה ליצור השוואה בין הפרמטרים.

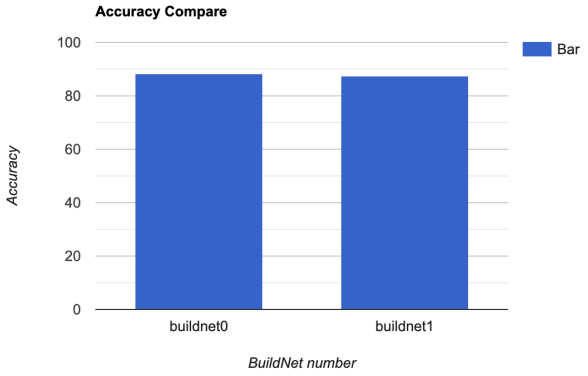
בנוסף אלגוריתם גנטי אינו מבטיח תמיד תוצאה טובה בשל מימד הרנדומיות שבו, לכן הרבה פעמים קשה לאמוד מודל טוב.

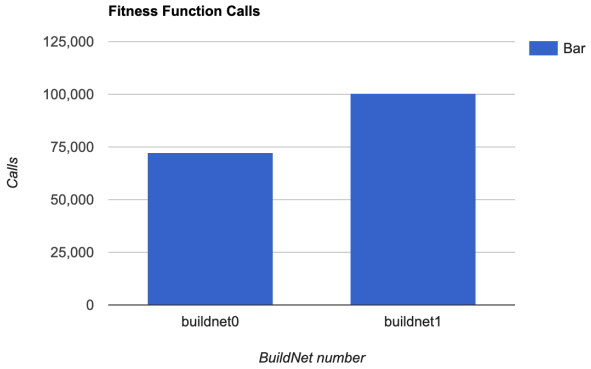
כדי להתמודד עם עניין הביצועים פעלתי ב2 דרכים

1) מודל בהתאם לקושי התבנית - בשל העובדה שbuildnet0 אמורה לפתור תבנית קלה יותר, השתמשתי במודל של 4 שכבות, מתוך הנחה שמודל של 4 שכבות יספיק לבעיה פשוטה ואילו בbuildnet1 השתמשתי במודל קצת יותר מורכב שמשתמש ב6 שכבות.

2) תכנות מקבילי - דבר נוסף שעשיתי היה שימוש בmulti threading, הפעולה של בדיקת הציון של כל רשת הינה פעולה עצמאית, לכן פעולה זו בוצעה בצורה מקבילית וחולקה בין ליבות המעבד, זאת כדי למקסם לחלוטין את הביצועים, בזכות זה זמן החיצה הופחת בצורה משמעותית.

השאיפה שלי הייתה לעבור את ה85% דיוק תוך שמירה על זמן ריצה סביר (מקסימום חצי שעה) להלן הביצועים:





כפי שניתן לראות אחוז הדיוק גבוה בשניהם יחסית (מעל 85%) כאשר buildnet0 מצליח להביא בצורה זניחה דיוק יותר גבוה, מבחינת זמן ריצה הביצועים בשניהם היו איטיים אך buildnet1 הייתה משמעותית יותר איטית וזאת בשל כמות השכבות הרבה שלה והצורך ביצירת אוכלוסיה התחלתית גדולה יותר וגם לפתח את האוכלוסיה למספר גדול יותר של דורות.

חוקיות

קשה להבין את החוקיות, גם בשל העובדה שהרבה פעמים ANN מהווה קופסא שחורה שמקשה על ההבנה, לכל אופן לדעתי בקובץ nn0 החוקיות היא מה הרוב של המספרים, ובמקרה שיש כמות שווה אזי התווית תהיה 1.