חלק א' יבש:

1)

a. המדד הביצועי המדובר הוא צריכת זיכרון, ב IDASTAR לעומת ASTAR צריכת הזיכרון היא לינארית באורך המסלול ,O(bd) וב ASTAR פרופורציונית למספר הצמתים שנוצרו O(b^d) .

b. מספר האיטרציות של DFS-ID : d (כעומק הפתרון)

מספר האיטרציות של IDA* הוא כסכום הערכים השונים ש f יכולה לקבל. (העמקה נעשית לפי ערכי f עולים) ולכן מספר זה תלוי במחירי הקשתות ובערכי היוריסטיקה על הצמתים, אם ערכי הf של כל הצמתים זהים - עם יוריסטיקה מושלמת – נקבל את המקרה הכי טוב- איטרציה אחת, ובמקרה הכי גרוע –כמספר הצמתים בגרף. בממוצע מספר האיטרציות יהיה גדול מ1 כנראה ביחס כלשהו למספר הצמתים וזה מספר זה יהיה גדול משמעותית מ d.

c. סעיף ג'

i. לכל היותר האלגוריתם ירוץ $C_D^* * K$ איטרציות כיוון שייתכן מצב שיוצא שרוך אינסופי בעל מחירי קשתות 1/k כך שצומת המטרה לא מופיע עליו, ועם יוריסטיקת האפס נקבל שבכל פעם שמתקדמים בשרוך האינסופי Qk(f_limit) שווה ל previous_f_limit + 1/k - החל מ 0 יגדל כל פעם ב 1/k ובשביל להגיע לפתרון צריך לעבור מספיק צמתים בשרוך כדי שה f_limit יאפשר להגיע לצומת במרחק C_D^*

ii. כמה קרוב לפתרון אופטימאלי אלגוריתם זה מתקרב במקרה הגרוע? חסם עליון הדוק זה הפרש המחירים של המסלול למטרה באלגוריתם 1A לבין המסלול האופטימאלי. $\varepsilon(A_1, D)$ לכל היותר יהיה 1/k במקרה של כמה קשתות באורכים שונים בטווח בין 1/k-0 שמתחברות מצמתים בחזית החיפוש, וה f_limit לא ישתנה באיטרציה הנוכחית- כלומר ייבחר previous_f_limit + 1/k ואז באקראי ייבחר להיבדק מסלול שאורכו שואף מלמטה ל 1/k כלומר ערך f של צומת המטרה תיהיה בגבול וייבחר מסלול גרוע לכל היותר ב 1/k מהמסלול האופטימאלי, זה המקרה היחיד שמצאתי שלא בוחר את המסלול האופטימאלי.

במקרה אחר האלגוריתם יבחר את המסלול האופטימאלי כמו IDASTAR.

d. סעיף ד'

i. החסם על מספר האיטרציות לא ישתנה כיוון שהמקרה הגרוע ביותר יהיה זהה לסעיף ג', כלומר $C_D^* * K$ כי במקרה גרוע מחיר כל קשת בשרוך האינסופי ישאר 1/k כדי להגיע לחסם המקבימאלי של מספר האיטרציות.

ii. במצב החדש$\varepsilon(A_2, D)$ יהיה 0 כי המצב שתואר בסעיף ג' ii לא יתאפשר אם ערך ה f על צומת יחושב לפי Qk

e. סעיף ה'

i. אותה דוגמה כמו בסעיף ג' נותנתן חסם של $C_D^* * K$

2) חיפוש מרובה סוכנים:

ערך המינימאקס לא ישתנה. נתון שזהו משחק סכום אפס. k ערכי המינימקס הטובים ביותר עבור אותו השחקן יהיו שווים לתוצאת מינימקס של הילדים שחזרו ()k_best .

באלגוריתם המקורי כל אחד מה Children מחזיר ערך מינימאקס ונבחר מביניהם הערך המקסימאלי על-ידי שחקן ה max והערך המינימאלי על-ידי שחקן הmin

אילו כבר היה נבחר רק הילד עם הערך הקיצוני הרצוי עבור השחקן הנוכחי, היה מוחזר הערך מינימאקס שלו. אילו k=1 זה היה המצב. אבל k קבוע לא ידוע וצריך לעבור על כל הילדים אותם ()k_best החזיקה למצוא את ערכי המינימאקס שלהם ולהחזיר את ערך המינימאקס הטוב ביותר עבור אותו שחקן. הוא בטוח נמצא שם אם k > 0.

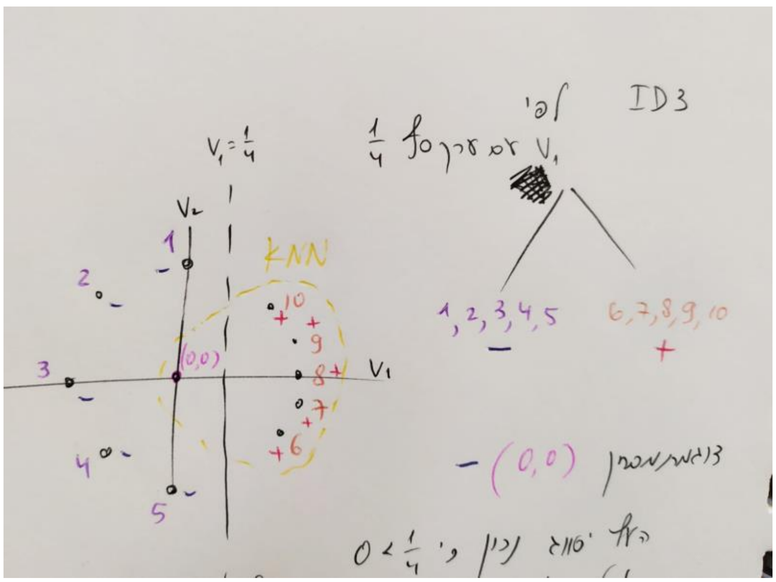
נניח בשלילה שקיים ילד עם ערך מינימאקס הכי טוב לשחקן הנוכחי שלא נבחר ב children באלגוריתם החדש – זאת סטירה להגדרת פעולת ()k_best.
כלומר ערך המינימאקס של השורש (שהאלגוריתם מחזיר) לא ישתנה והאלגוריתם יעבוד כמו מינימאקס הרגיל רק בלי לבדוק מצבים מיותרים.

3) מערכות לומדות

a.

נניח ש v1, v2 הם צירים של מישור וניתן למקם את הדוגמאות על מישור במרחק 1 מראשית הצירים על מעגל היחידה:

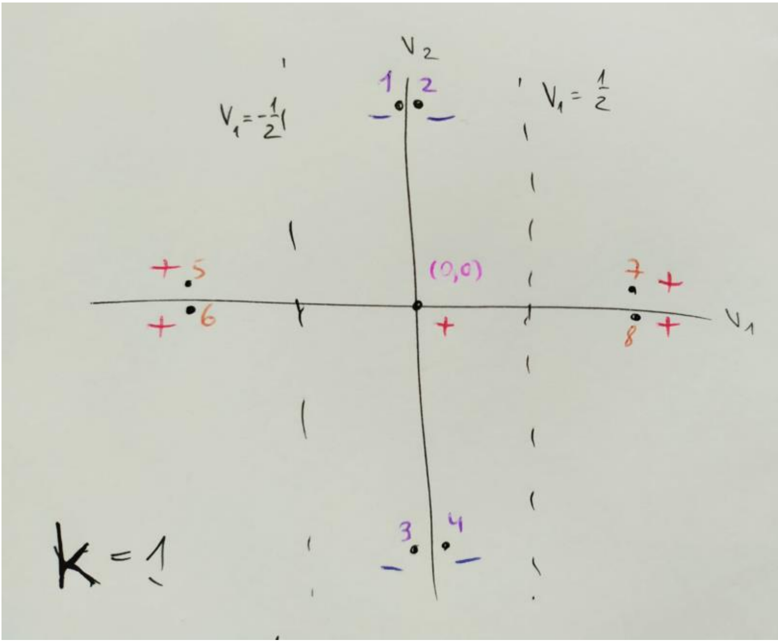
1,2,3,4,5 כולן דוגמאות אימון עם סיווג שלילי **–** בעלות ערכי v1 קטנים מ 0.25
6,7,8,9,10 דוגמאות אימון עם סיווג חיובי **+** נמצאות מימין לקו v1 = 0.25 כלומר בעלי ערכי v1 גדולים מ 0.25
העץ ID3 שייבנה יסווג לפי תכונה V1 עם ערך הסף 0.25 **ויסווג נכון** את דוגמת המבחן 0,0 שסיווגה הוא שלישי **-** אבל לפי הגדרות אופן סיווג של KNN שהוצג, בהנחה ש K קטן ממספר דוגמאות האימון ואי זוגי – הוא יבחר מתוך כל דוגמאות האימון הנמצאות במרחק שווה מדוגמת המבחן את הדוגמאות עם ערכי v1 הגדולים יותר – וכך החיוביים מהווים את הרוב בהחלטה על סיווג דוגמת המבחן ותוצאת הסיווג יוצאת **שגויה**.

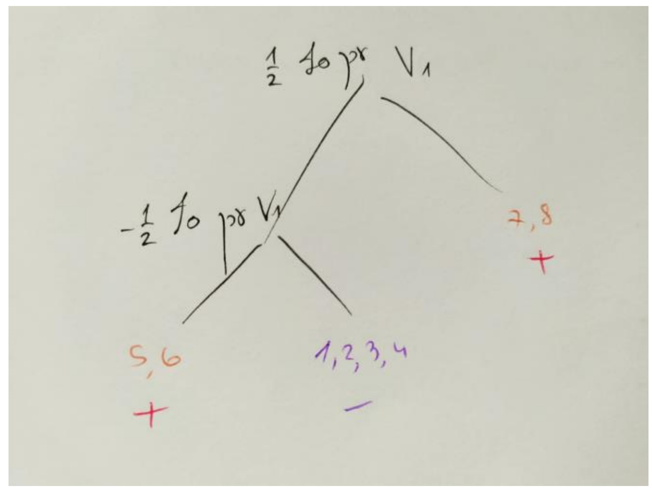


b. נרצה לבחור K, ודוגמאות אימון כך שמסווג ID3 יטעה בסיווג של דוגמת מבחן כלשהי ומסווג KNN לא יטעה. עבור K=1 והדוגמאות אימון כולן נמצאות על מעגל היחידה כלומר במרחק 1 מהראשית:

1,2,3,4 דוגמאות אימון עם סיווג שלילי **–**
5,6,7,8 דוגמאות אימון עם סיווג חיובי **+**
KNN יסווג כל דוגמה לפי הדוגמה הקרובה אליה ביותר כלומר יהיה סיווג מטרה מדויק והעץ יראה כך עם פיצול פעמיים לפי התכונה V1:

(0,0) דוגמת מבחן עם סיווג **חיובי** לפי 1NN תקבל את הסיווג הנכון כי ערך ה v1 של דוגמה 7 ושל דוגמה 8 החיוביות גבוה משאר הדוכמאות ואחת מהן תיבחר באמצעות ה1NN. ולעומת זאת בעץ ה 3ID יתקבל סיווג שגוי כי (0,0) תגיע לעלה המכיל את הדוגמאות השליליות.





משימה 2 דוח וניסויים:

ערכי ה default שבחרתי לפרמטרים: K=5, M = 2, epsilon = 0.01

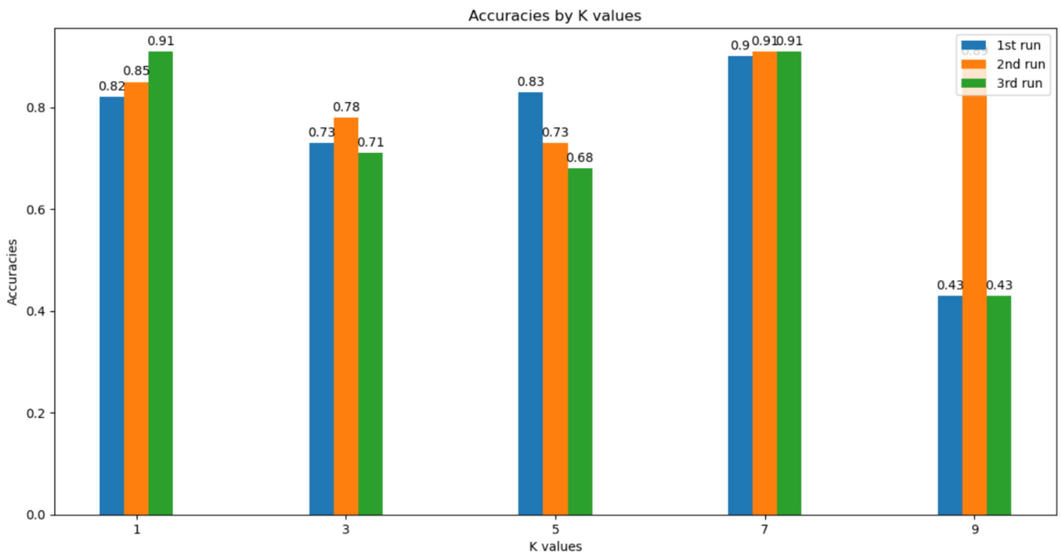
מקוצר הזמן בניסויין נבחרות ערכי סף באמצעות הפונקציה np.percentile

בטווח: [25, 50, 75]

לפי הגדרת האלגוריתם התכונה לפיה מתפצל העץ נבחרה רנדומלית מבין התכונות בעלי הדיוק הממושקל המקסימאלי לערך סף מסויים, העצים שלי יצאו שונים בכל הרצה בגלל אופן פיצול זה ולכן ערכתי 3 פעמים כל ניסוי עם אותו טווח של ערכים

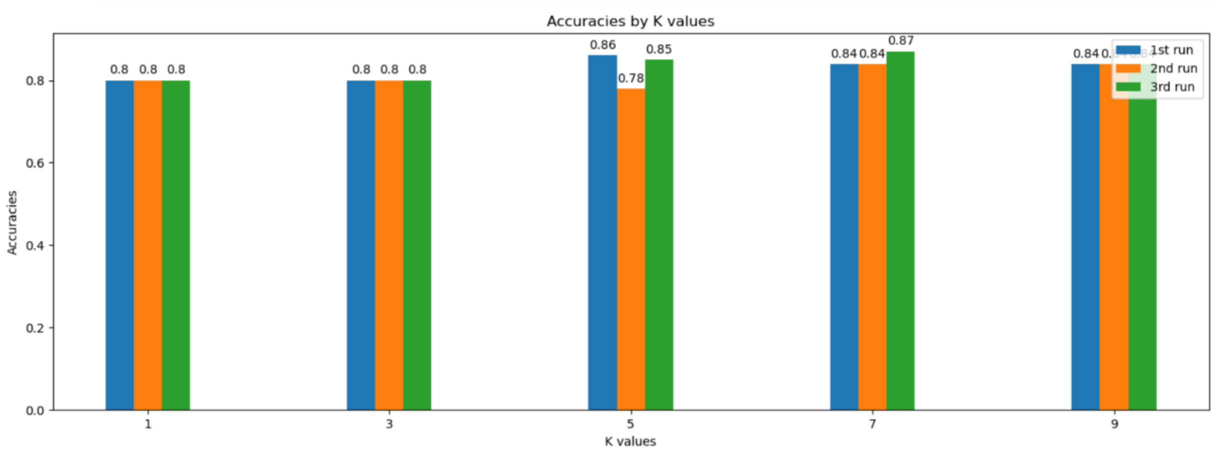
ערכי K [1, 3, 5, 7, 9]ל 9 dataset

כאשר M = 2, epsilon = 0.01



ערכי K [1, 3, 5, 7, 9]ל 12 dataset

כאשר M = 2, epsilon = 0.01



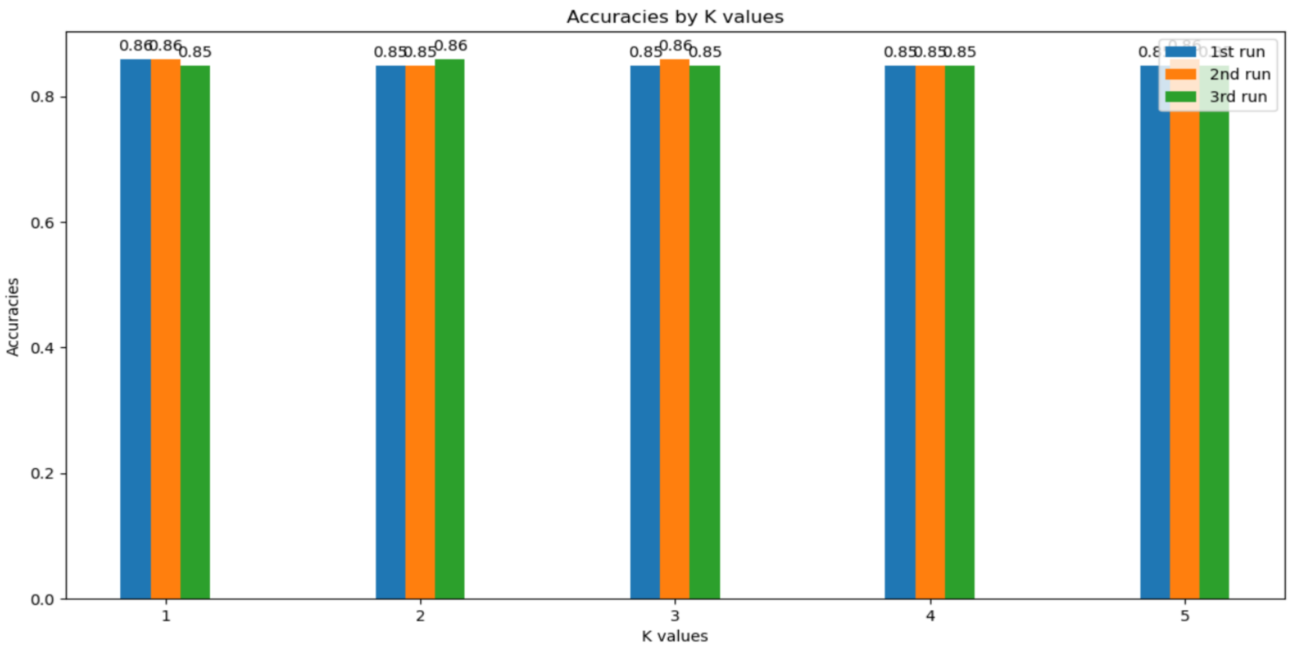
נראה כי עבור K = 7 התקבל דיוק גבוה יותר בממוצע של ההרצות עבור שני הdatasets

אם K גדול מידי יש רעש בסיווג, ואם קטן מידי לפעמים חסר מידע להתאמת הסיווג

ערכי M [1, 2, 3, 4, 5] ל dataset 9

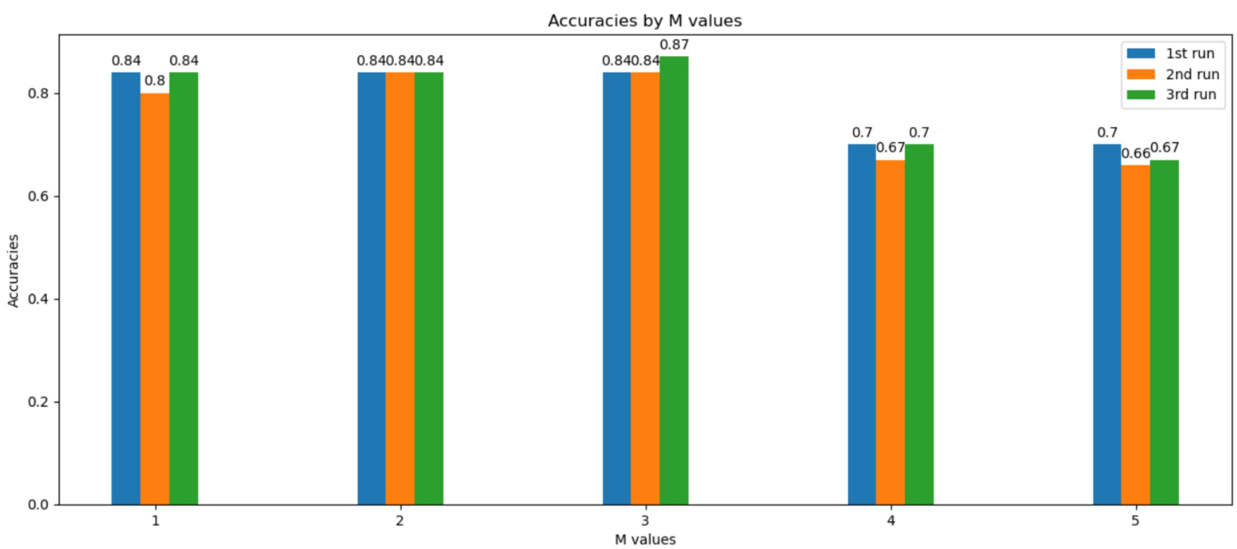
בטעות בכותרות רשום K זה בעצם M אין לי זמן להריץ שוב

כאשר K = 7, epsilon = 0.01



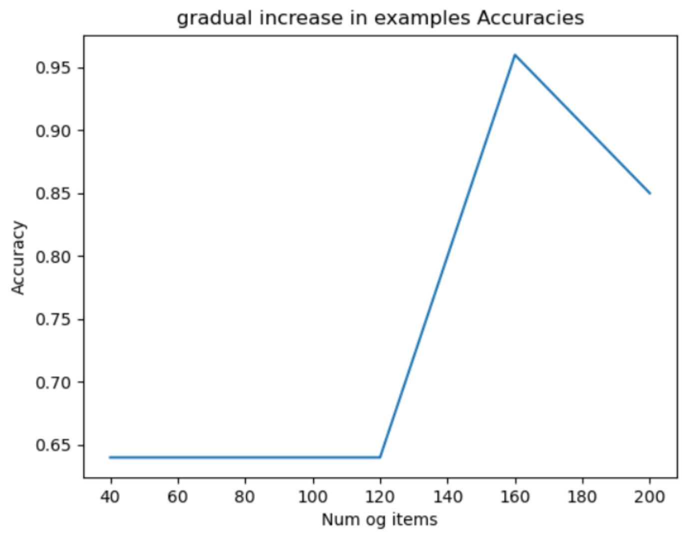
ערכי M [1, 2, 3, 4, 5] ל dataset 12

כאשר K = 6, epsilon = 0.01

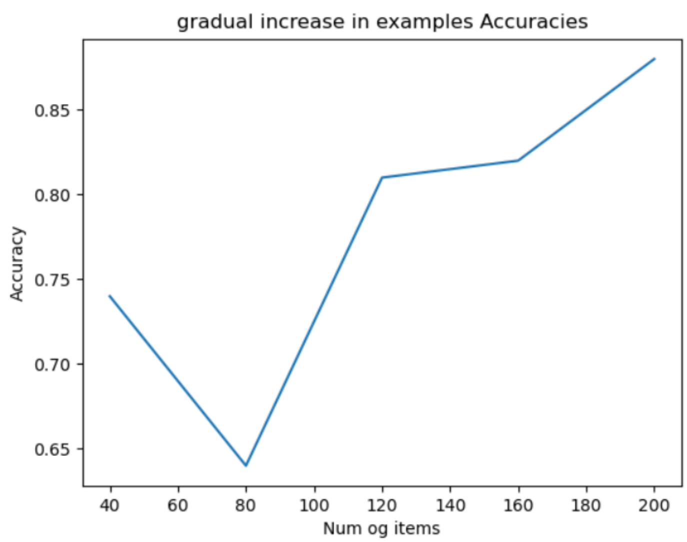


נראה M = 3 נותן דיוק של גבול לגודל עלה אופטימאלי

3. ניסויים של exp.py עבוד dataset9



עבוד dataset 12:



נראה שככל שעולה מספר הדגימות האימון כך עולה הדיוק.

משימה 3 שיפור האלגוריתם

ניסיון 1:

לבנות 7 עצים שונים עם הפרמטרים הנבחרים לאחר הניסויים ולבחור עלפי החלטת הרוב

(7,3,0.01) דיוק: 91, 65, 86

ניסיון 2:

לבנות 7 עצים עם הצלבות שונות של פרמטים

דיוק ממוצע של 0.95 בהרצות

נסיון 3:

לכל עץ לשמור את השגיאה הגדולה ביותר שהייתה בעלה ואת גודל העלה הגדול ביותר והקטן ביותר – ואת הגודל הממוצע של העלים

לקחת את העצים עם סטיית טקן הקטנה ביותר של גודל העלים.