בס"ד

חשוב לזכור!!

* DFS לא מובטח פתרון ולא תמיד קיים complete. (יכול להתקע במקרה אינסופי)
* זמן ריצה: DFS, BFS, ID - $O(b^d)$
Uniform cost = $O(d\, b^d \log(b))$
* סיבוכיות מקום: DFS, ID - $O(bd)$
uniform
BFS, UC - $O(b^d)$
* DFS לא מובטח מסלול קצר ביותר - כאשר עלות שלב (קשתות שוות ערך)
* UC מובטח מסלול קצר ביותר כאשר עלות שונה (קשתות שונות ערך)
* בהגדרה מקסימום ומינימום - $\alpha=-\infty, \beta=+\infty$
ב Max יש ערך $\geq \beta$ לא נכנסים לתת עץ
ב Min יש ערך $\leq \alpha$ לא נכנסים לתת עץ
- מרחק מנהטן - סכום המרחק $d=1+3+2=6$
- מרחק המינג - כל מילה שמופיעה=1 אחרת 0. $\sum(x_i = y_i)$
- מרחק צ'בישב - המרחק המקסימלי בין 2 צלעים במרחק מנהטן.
- מרחק אוקלידי - המרחק הרגיל $d=\sqrt{(x-x')^2+(y-y')^2}$
* MCV - נבחר את המשתנה הכי מוגבל (משתנה עם הכי פחות אפשרויות).
* LCV - מי שמגביל הכי פחות את השאר (בחירה ששומרת [illegible]).
* CP - הבטחה שמונעת את מה [illegible] ביותר של ערכים אפשריים.
* Percision - כמה מהולל דייק, Recall - כמה הרבה זיהה
* מבצעים את האיטרציות בK-means עד אשר המרכזים מפסיקים לזוז (או עד אשר המרחקים מפסיקים לקטון).
* Unsuper L. ע"פ משימות אחרות בהינתן [illegible] מודל כלומר ragged data

המשך מושגים

* Backtracking - טכניקה שבה האלגוריתם מתקדם קדימה במרחב
החיפוש עד שמגיע למצב שבו אין לו להתקדם, אז חוזר אחורה
ומחפש דרך אחרת עד שמוצא פתרון.

* Hill-climbing - אלגוריתם חיפוש מבוסס מידע שמשתמש בפונק'
הערכה (היוריסטיקה) כדי לבחור את הצעד הבא בדרך למציאת פתרון.
בוחר את המהלך שמביא לשיפור מידי (כמו טיפוס על גבעה)

* Gradient Descent - אלגוריתם חיפוש מבוסס מידע שמשתמש בפונק'
הערכה רציפה כדי לבחור את הצעד הבא. האלגוריתם בוחר
בצעד שמוריד את ערך פונק' הערכה בצורה המשמעותית ביותר.

* Simulated Annealing - האלגוריתם מאפשר "קפיצות" למצבים
פחות טובים בהתחלה ובמשך הזמן הופך להיות "זהיר".

* Local search Algorithm - מתחיל בהצבות ערכים אקראיים ולאחר מכן
מבצע שיפורים מקומיים כדי למצוא קונפליקטים ולהתאים להגבלות

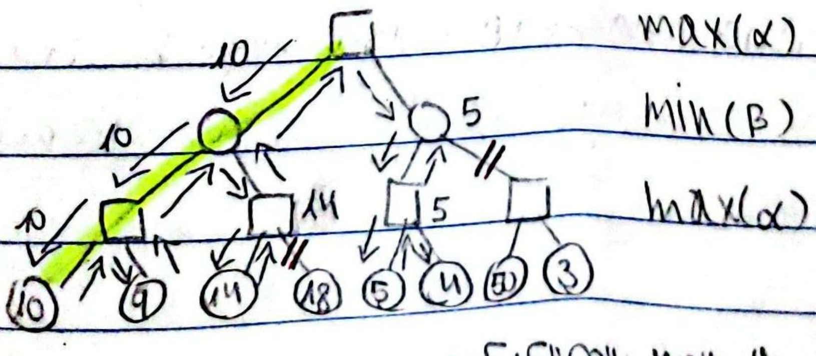
* Infogain מדד מרכזי לבניית עץ החלטה

אלגוריתם Minimax

* זהו אלגוריתם המשמש למשחקים בו שחקנים עם סכום אפס
כי הניצחון של שחקן אחד הוא ההפסד של השני
- האלגוריתם מחשב את הערך של כל מצב אפשרי בעץ
מתחילים מהעלים ועולים כלפי השורש.

○ Alpha-Beta-PA. שיפור לאלגוריתם Minimax המשמש
• לצמצום מס' הצמתים הנבדקים בעץ המשחק, המטרה לשפר
יעילות ע"י חיתוך ענפים בעץ שאינם משפיעים על ההחלטה הסופית.
- שמירת גבולות: במהלך החיפוש האלגוריתם שומר שני ערכים
α (הערך המקסימלי שמובטח לשחקן max) ערך התחלתי: $-\infty$
β (הערך המינימלי שמובטח לשחקן min) ערך התחלתי: $+\infty$
- אם האלגוריתם מוצא מהלך שבו $\alpha \geq \beta$ הוא חותך את הענף הנוכחי.
○ בצומת max אם ערך הצומת $\geq \beta$ נגזום את הצומת.
○ בצומת min אם ערך הצומת $\leq \alpha$ נגזום את הצומת.
- ערכים טובים משמאל לימין האלגוריתם יגזום הרבה צמתים.
- ברגע שאין ערך שיותר טוב נוכל לגזום את הענף.

דוגמה לגיזום:



○ גיזום אופטימלי - בעץ מקס אופטימלי לסדר את הערכים הגבוהים משמאל.
בעץ מין הפוך, אופטימלי: שהערכים הנמוכים משמאל.

בס"ד

מושגים

* הסילואט של האשכול - הוא מדד שמשמש להערכת מידת ההפרדה של קבוצת נתונים. משמש כדי להעריך את הנתונים מאורגנים באשכולות טובים או "מעורבבים" בצורה.
  - ערך קרוב ל-0 מצביע על שהנתונים מתפלגים "מעורבב"
  - ערך קרוב ל-1 מצביע על שהנתונים מאורגנים באשכולות ברורים.
* אלגוריתמים לפתרון CSP:
  - MCV (Most Constrained Variable) - בוחר את המשתנה עם הכי הרבה מגבלות על הערכים האפשריים שלו, כאשר המשתנה עם הכי הרבה המגבלות הוא - (נבחר את הכי מוגבל - הכי מעט מקומות בערכים).
  - LCV (Least Constraining Value) - בוחר את הערך שמגביל הכי פחות את האפשרויות של המשתנים האחרים. כאשר משתמשים בשיטה מינימלית את תחומי הערכים של המשתנים האחרים (מפחית את הצורך ב-Backtracking)

כלומר בפועל בשאלות:

MCV - נבחר את המשבצת עם הכי פחות אפשרויות

LCV - נבחר במה שמגביל הכי פחות את האחרים

* KNN - K המשתנים הקרובים שהאלגוריתם יחשב (בד"כ במרחק אוקלידי) ערך K קטן יכול להביא למודל עם שונות גבוהה, וערך K גדול יכול להביא להטיה גבוהה ושונות נמוכה, לכן חשוב למצוא איזון ביניהם
* K means - אלגוריתם נפוץ המשמש לחלוקת נתונים לאשכולות.
* אלגוריתם עצלן זה אלגוריתם שלא נשען על מודל (כמו KMeans).
* נזכור שערך ה recall הוא לפי שורה וה Precision לפי עמודה.

חשוב לזכור!

* בזיהוי דואר זבל FP מזיק יותר
ובזיהוי בדיקות רפואיות FN יותר קריטי

* כשנתון 2 נק' במישור ונפעיל עליהן רגרסיה לינארית
נקבל שגיאה 0 כי זה יוצר ישר ביניהם

* האלגוריתם BFS מצריך את הסיבוכיות הכי גבוהה של
זיכרון $O(b^d)$ (יותר מ-ID, DFS, A*, Alpha-Beta).

* חישוב מרחקים: דוגמה: $y=(-3,2,0,1)$ $x=(1,2,3,4)$

מרחק אוקלידי: $d=\sqrt{4^2+0^2+3^2+3^2}=5.83$

מרחק מנהטן: $d=4+0+3+3=10$

מרחק המינג: $\sum^4 (x_i \neq y_i)$

A כאשר משחק בו משחקים שני שחקנים בתורות, ואשר סוף
המשחק הוא תיקו או בנצחון לאחד השחקנים והפסד
לשחקן השני יש לפחות לאחד השחקנים אסטרטגיה
שמבטיחה שלא יפסיד.

* פונק' היוריסטית אשר היא מרחק מנהטן מכיון שהוא מחשב
כמה מהלכים היו צריך לעשות במינימום עד להגיע למקום הנכון.

* KMeans הינו אלגוריתם חמדני.

* כשתוצאות של GR ושל IG סותרות מקבלים את GR
נבחר את המבחן שהשיטה שלו יותר מבוססת.

* האלגוריתם KNN משתמש בזיכרון. ויש הבדל בין מימושים של
האלגוריתם כך ש $k=1$ או משתמש בזיכרון הכי מינימלי.

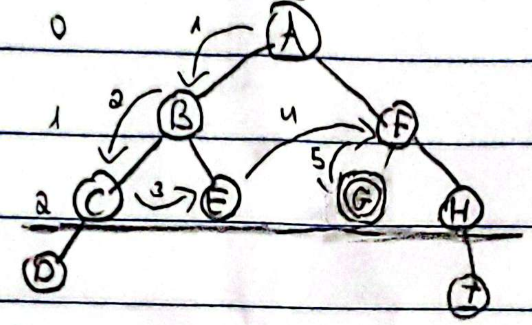
מבוא לבינה מלאכותית - אלגוריתמים

* BFS - כמו שלמדנו עם אלגוריתם חיפוש לרוחב (תור)
* DFS - כמו שלמדנו במבנים אלגוריתם חיפוש לעומק (מחסנית)
* IDS - כמו DFS רק שנותנים מגבלה, ואז לוקחים עד כמה שמותר כל פעם מגבילים את העומק חיפוש - ואז הוא מתנהג כמו BFS.

ABCEFG

דוגמא:

מגבילים בעומק 2



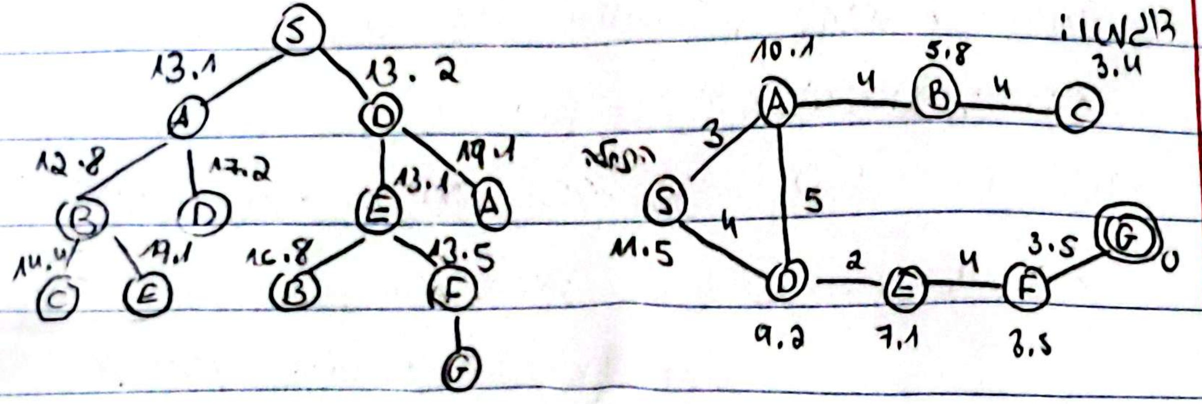
- DFS מוצא פתרון ומשתמש בזכרון פולינומי, אבל לא תמיד מוצא את הדרך הקצרה ביותר
- BFS מוצא את הדרך הקצרה ביותר אבל משתמש בזכרון רב.
- IDS מוצא את הדרך הקצרה ביותר, בזמן ריצה דומה לDFS.

* UCS (uniform-cost) (עלות) - מחפש את המסלול בעל העלות הכוללת הנמוכה ביותר ומחשב את העלות הכוללת של כל מסלול. מתאים למצבים שבהם יש עלויות שונות לצעדים ומחשב את המסלול הכי משתלם כלכלית. g(n) משתמשים בהם בעיקר בבעיות עם מצבים ממוינים לפי עלות שלהם.
* GBFS (Greedy) - אלגוריתם חיפוש מבוסס מידע שמשתמש בפונקציית הערכה כלשהי כדי לבחור את הצומת הבא להרחבה. האלגוריתם משתמש רק ב h(n) (הערכה) ולא בדרך האמיתית. h(n)

המשך בינה מלאכותית

* A* Algorithm - אלגוריתם לחיפוש המסלול הקצר ביותר בגרפים או במבוכים. האלגוריתם משתמש בפונק׳ היוריסטיקה שמעריכה את המרחק המשוער מכל צומת לצומת היעד. ויש את העלות הממשית מצומת ההתחלה לכל צומת.

דוגמה:



כדי לחשב את ההערכה נסכם את הקילומטרים מהיעד h(n) ואת הקילומטרים במסלול g(n) $f(n) = h(n) + g(n)$

העלות הממשית של המסלול

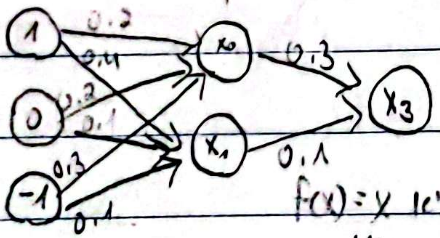
- פונק׳ היוריסטית קבילה אם $h^*(n)$ הערכה מ-n ליעד $h(n) <= h^*(n)$

* IDA* - גרסה של האלגוריתם A* שמשלבת את הרעיון של חיפוש מעמיק מוגבל (ID) כדי לשמור על שימוש בזיכרון נמוך יותר. האלגוריתם מתחיל בעומק חיפוש התחלתי הקרוי סף (threshold) ומבצע חיפוש בעומק מוגבל (DFS) עד שעלות הערך חורגת של הקודקוד הנוכחי. חורג מהסף הנוכחי.

* משמעות של גיזום - בשורש max אם $\beta \geq x$ גוזם את שאר הבנים של צומת המקס, ואם $\alpha < x$ מעדכן את α להיות x. בשורש min אם $x \leq \alpha$ גוזם את שאר הבנים, ואם לא היה גיזום אם $x < \beta$ מעדכן את β להיות x.

איך לפתור מבחנים?

* בשאלות של גיזום מקסימלי נסדר את הערכים בסדר עולה. הבנים של max שהערכים בשמאל יהיו הכי גדולים שאפשר. הבנים של min שהערכים בשמאל יהיו הכי קטנים שאפשר.
* בשאלות של הסטטיסטיקה של המסקנים מדבר על איך כדאי לסווג ושכולות. לכן נבדוק אם יש חלוקה ברורה לקבוצות "1 ו-0" שהנקודות מפוזרות באקראי.
ככל שהנקודות מפוזרים באקראי נקבל ערך קרוב ל-0.
ככל שהסיווג ברור לקבוצות נקבל ערך קרוב ל-1.
* בשאלה של רשתות נוירונים צריך לרשום את הפלט



פונקציית ה Activation היא $f(x)=x$

$$x_0 = 1\cdot0.2+0\cdot0.2+(-1)\cdot0.3+\overset{bias}{0} = \underline{\underline{-0.1}}$$

$$x_1 = 1\cdot0.4+0\cdot0.1+(-1)\cdot0.1+\overset{bias}{0} = \underline{\underline{0.3}}$$

בגלל שהפונק' היא $f(x)=x$ נקבל: $f(-0.1)=-0.1$, $f(0.3)=0.3$

שלב שני: נמשיך את אותו רעיון רק לפי השכבה בפונקציית Activation

לכן התשובה היא: $x_3 = -0.1\cdot0.3+0.3\cdot0.1 = \underline{\underline{0}}$

* בשאלות של עצים עם האלגוריתם A^* כדי לחשב את הטריקים משכים. נכתוב הקודות מוצקה לקדקוד האיסטר ההערכה היוריסטית. $f(n)=g(n)+h(n)$ והכל שלב ניקח את הקדקוד עם הערך הנמוך יותר עד שנגיע לקדקוד המטרה.

חשוב לזכור!

* העדפתי דגם לגבי FP נמוך יותר
ובבעיות רפואיות חשובות FN יותר קריטי
* כשנתון 2 נק' במישור ואם נפעיל עליהן רגרסיה לינארית
נקבל שגיאה 0 כי זה יעבור ישר ביניהם
* האלגוריתם BFS מצריך את הסיבוכיות הכי גבוהה של
זיכרון $O(b^d)$ (יותר מ ID, DFS, A*, Alpha-Beta).
* חישוב מרחקים: דוגמא: $y=(-3,2,0,1)$ $x=(1,2,3,4)$
מרחק אוקלידי: $d=\sqrt{4^2+0^2+3^2+3^2}=5.83$
מרחק מנהטן: $d=4+0+3+3=10$
מרחק המינג: $\sum^4 (x_i \neq y_i)$
* כדי לשחק בו משחק של שחקנים בתורות, כאשר סוף
המשחק הוא תיקו או ניצחון לאחד השחקנים והפסד
לשחקן השני יש לבחור לאחד השחקנים אסטרטגיה
שמבטיחה שלא יפסיד.
* פונק' היוריסטית טובה היא מרחק מנהטן מכיוון שהוא מחשב
כמה מהלכים הוא יצטרך לעשות במינימום עד להגיע למקום הנכון.
* KMeans נחשב אלגוריתם חמדני.
* כשמודדים את IG ועם זה הרבה ערכים מקבלים יותר IG
נבחר את המבחן שהשונות שלו יותר מפוזרת.
* האלגוריתם kNN שהוא עצלני. ועם פרמ' מממש את
האלגוריתם רק כש $k=1$ אז השגיאה הכי מינורית.

בס״ד

השג פתרון שאולי לא מושלם

* מדד Gain ratio נשתמש בו כאשר יש קטגוריה מכילה הרבה תתי קטגוריות ספציפיות מידי שיכולה להגיע לערך אנטרופיה שווה לאפס. המדד הזה מתחשב "בעונש" של תתי קטגוריה. ככל שהוא יותר גבוה הוא יותר טוב!

נוסחאות:

$$* \ \text{Gain ratio} = \frac{\text{Gain}(S,A)}{\text{IntrinsicInfo}(S,A)}$$

$$* \ \text{IntrinsicInfo}(S,A) = -\sum \frac{S_i}{S} \log_2\left(\frac{S_i}{S}\right)$$

S_i שכיחות של כל תת קטגוריה, S סכום השורות (הקטגוריות), $\frac{S_i}{S}$ יחס של תת קטגוריה מתוך הקטגוריה הכוללת.

* חישוב מדד Gini ימדוד את המצב הכי טוב מקטגוריה מסוימת, ועבור כל הקטגוריה הנפרדות עם המידע הכולל.

נוסחה ג'יני:

$$\text{Gini}(P_1, P_2 \dots P_n) = 1 - \sum^{n} P_i^2$$

נשים ♡: הנוסחה מתייחסת לתת קטגוריה, ולכן אם נצטרך לחשב לכל קטגוריה יותר כללית (עם תתי קטגוריות) נצטרך להכפיל כל מדד ג'יני של תת קטגוריה ביחס שלה לקטגוריה הכוללת.

למשל נרצה לחשב את הGini של Humidity נתונים כאלה:

תתי קטגוריה של Humidity	Yes	No	סה"כ	
High	3	4	7	ג'יני: $1-\left(\left(\frac{3}{7}\right)^2+\left(\frac{4}{7}\right)^2\right)$
Normal	6	1	7	ג'יני: $1-\left(\left(\frac{1}{7}\right)^2+\left(\frac{6}{7}\right)^2\right)$

היחס של High ושל Normal הם $\frac{7}{14}$ לכן הג'יני הכללי:

$$\frac{7}{14}\left(1-\left(\left(\frac{3}{7}\right)^2+\left(\frac{4}{7}\right)^2\right)\right)+\frac{7}{14}\left(1-\left(\left(\frac{1}{7}\right)^2+\left(\frac{6}{7}\right)^2\right)\right)$$

- מדדי Entropy ו IntrinsicInfo, Gini - ערך נמוך.
- מדדים IG, Gain ratio - ערך גבוה.

הערות ממבחנים

* בפונק' experiment באלגוריתם k means השורה if cls[j] != cls[i]
נדרשת כדי לספור מקרים שבהם שני המרכזים ההתחלתיים
זהים מה שיוביל לתוצאה חסרת משמעות באלגוריתם k means
(לא ניתן להריץ kmeans אם בוחרים בנקודות זהות כמרכזים התחלתיים).

* האלגוריתמים IG ו Gain ratio משמשים לחיזוי
והאלגוריתמים logistic regression וkmeans משמשים לסיווג.

* כאשר יש לנו הרבה קטגוריות עדיף להשתמש
ב Gain ratio ובמקרה שיש לנו יותר עולם "בינארי"
עדיף להשתמש ב IG

* רגרסיה לוגיסטית נותנת סיווג ולא ערך
רגרסיות לינאריות ופולינומיות נותנות ערכים
אם עקומים => רגרסיה פולינומית, קו ישר => לינארית.

* Intrinsic Information גבוהה משמע שיש אי וודאות
גבוהה, כלומר כמות גדולה של ערכים שונים.

* כאשר יש מטרה לאבחן חולים אך עדיף לאבחן הרבה
אנשים בריאים כחולים מאשר לפספס חולים לכן
עדיף recall מאשר precision

* שאלו שאלה נוספת על מערך אחר שבו שני המחקרים לא ישנו
את המחלק שלהם כי לא שווה להם.

* forgiving זוהי רגרסיה שאומרת שלא משנה "מה" כי אין אף
אותן טעויות הגדולות בשלב ההתחלה כדי שהמודל ישפר אותו בצורה שלמה.

מג

המשך סיכום שאלות ממבחנים

* בשאלות של kmeans כאשר נתבקש לחשב מרחק אוקלידי בהינתן נקודות מרכזיות נחשב מרחק אוקלידי בין כל נקודה לנקודות המרכזיות $d(p_1,p_2)=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$
ונסווג את התוצאות לכל קטגוריה לפי המרחק הקצר יותר.
לאחר מכן נחשב נקודות מרכז חדשות לפי ממוצע.
ונחשב מחדש מרחק אוקלידי עם הנק' מרכז החדשות,
ככה נמשיך עד שנגיע למצב שאין סיווגים חדשים.

* בשאלות של בעיית סיווג עם שימוש באלגוריתם KNN עם משקולות מרחק הדרך לחישוב ההסתברות לסיווג היא:
1. נחשב מרחק אוקלידי של כל אחת מהנקודות לנקודה הנתונה.
2. נחשב את פונק המשקולות של כל אחת מהנקודות.
3. ניקח את k המשקלים הגדולים ביותר.
4. נסכם את המשקל בכל קטגוריה ולפי זה נבחר את הclass.
5. נחשב את ההסתברות של כל class לפי משקל הclass חלקי סכום המשקלים.

* בשאלות של ID כאשר שואלים מהו סדר הסריקה צריך לזכור כי ID זה אלגוריתם שעושה DFS עם הגבלת עומק, ויש לו גבול מוגבל את מה שהוא מחפש ואנחנו מגדילים את העומק ומריצים שוב את DFS. נתחיל מעומק 0.

* KNN נחשב lazy בגלל שמשתמש למשהו שלא הולך בזמן אימון ולכן מודל מקדים.

איך לפתור מבחנים - המשך

A. Information Gain - בשאלות אלו יש לנו נתונים מסוימים למשל על הסטודנטים כגון גיל, מקום, ציון וכו' ואנחנו רוצים לחזות אם הם יתקבלו למוסד לימודים או לא, נחשב את האנטרופיה של כל תכונה ונבחר את התכונה עם רווח המידע הגבוה ביותר כדי לקבוע את הקריטריון לפיצול הצומת.

הנוסחאות שנשתמש בהן (יש בנוסחון)

I. $\text{Information Gain} = E(T) - E(T,X)$

II. $E(X) = -\sum p_i \cdot \log_2 p_i$ — X תכונה שנבחן, T המטרה שבודקים.

III. $E(T,X) = \sum p(c) \cdot E(c)$

קטגוריה של תכונה מסוימת ביחס למטרה — $\frac{\text{מס' הפעמים שהתכונה מופיעה}}{\text{מס' השורות סה"כ}}$ — האנטרופיה של התכונה שבודקים בהקשר של המטרה

התכונה עם ה-IG הכי גבוה - אותה נבחר לפצל איתה את ההחלטה.

קיצורים: 1) אם בתכונה מסוימת התוצאות מתחלקות שווה ה $E = 1$

2) אם אחת התוצאות של תכונה מסוימת היא 0, אז $E = 0$

1. דבר ראשון נגדיר את המטרה של הנתונים ונבנה את $E(T)$.

נחשב כמה שורות יש סה"כ (y), כמה לא (x_0) וכמה כן (x_1)

$$E(T) = -\left(\frac{x_0}{y}\log_2\frac{x_0}{y} + \frac{x_1}{y}\log_2\frac{x_1}{y}\right) = T$$

2. לאחר מכן נחשב IG עבור כל תכונה, נתחיל לחשב עבור התכונה הראשונה (אם היא מורכבת מקטגוריות נחשב עבור כל קטגוריה בנפרד ונסכם).

$$E(T,X) = \frac{\text{הקטגוריה}}{y}\left(-\left(\frac{x_0}{k}\log_2\frac{x_0}{k} + \frac{x_1}{k}\log_2\frac{x_1}{k}\right)\right) = k$$

$$IG = E(T) - E(T,X) = T - k$$ נחשב עבור כל קטגוריה

ובסוף נבחר את ה-IG של הקטגוריה הכי גבוהה = התכונה הכי משמעותית.

המשך פתרון מבחנים

* בשאלות שצריך לחשב את F1 score נחשב קודם את precision ואת recall שמחשבים עבור כל הקטגוריה חלקי סכום הקטגוריות. ואז נשתמש בנוסחא:

(F score)

$$\text{F-Measure} = \frac{2 * precision * recall}{precision + recall}$$

* בשאלות של חישוב מרחקים בין 2 וקטורים

דוגמא: $y=(2,1,2,3,0)$ $x=(1,0,2,0,4)$

- מרחק מנהטן: $|1-2|+|0-1|+|2-2|+|0-3|+|4-0| = 9$
כלומר, סכום המרחקים.
- מרחק המינג: כל מילה שמופיעה בשני מילים נרשום 1 ואחרת 0, המרחק זה כמות המספרים השונים. $x=0,1,0,0$ $y=1,0,1,0$ $=3$
- מרחק אוקלידי: $d=\sqrt{(1-2)^2+(0-1)^2+(2-2)^2+(0-3)^2+(4-0)^2}=\sqrt{27}$
- מרחק צ'בישב: נבדוק חיסור בערך מוחלט בין הזוגות ונבחר את הזוג עם הערך הגבוה ביותר.

• אם כתוב בשאלה binary נבחר וקטורים של 0 ו1.

• אם אין שני מילים ניצור שני מילים משותפים.

* בשאלות של מציאת כמה תחנות ב Uniform cost

הרעיון הוא לפתוח כל פעם את הקודקודים הכי זולים עד שמגיעים לאותו קודקוד היעד במסלול הקצר ביותר! כל עוד יש איך לפתוח ולהגיע ליעד בדרך זולה יותר. ממשיכים לפתוח (גם אם כבר הגענו אליו במסלולים קודמים).

חשוב פתרון מבחנים

A רגרסיה לינארית - בהינתן מס' נק' נרצה למצוא קו ישר שמייצג
קירבה כמה שיותר לנקודות, קו ישר מהצורה $y=\theta_0+\theta_1 x$

לזכור בעל פה!

$$\mathrm{I}\quad \frac{1}{m}\cdot\sum_{i=1}^{m}(\theta_0+\theta_1 x_i-y_i)=0$$

$$\mathrm{II}\quad \frac{1}{m}\cdot\sum^{m}(\theta_0+\theta_1 x_i-y_i)x_i=0$$

+ רגרסיה פולינומית - נרצה למצוא פולינום מהצורה $y=\theta_0+\theta_1 x+\theta_2 x^2\ldots$
נקבל נק' עם 3 קורדינטות (x_i, z_i, y_i) ומערכת של 3 משוואות:

$$\mathrm{I}\quad \frac{1}{m}\sum^{m}(\theta_0+\theta_1 x_i+\theta_2 z_i-y_i)=0$$

$$\mathrm{II}\quad \frac{1}{m}\sum^{m}(\theta_0+\theta_1 x_i+\theta_2 z_i-y_i)x_i=0$$

$$\mathrm{III}\quad \frac{1}{m}\sum^{n}(\theta_0+\theta_1 x_i+\theta_2 z_i-y_i)z_i=0$$

+ רגרסיה לוגיסטית - רגרסיה זו נועדה לכך שנרצה ערך של
1 או 0, נשתמש בפונק' הסיווג (כאשר $z=0$ נקבל סיווג והתשובה היא $\frac{1}{2}$).

פונק' הסיווג: $\dfrac{1}{1+e^{-z}}=\dfrac{1}{1+e^{-(\theta_0+\theta_1 x_i+\theta_2 y_i\ldots)}}$

z הוא ערך של נק' מסוימת

A בשאלות סודוקו של אלגוריתם csp קודם נבדוק מי יכול
בכלל להיות במשבצת המבוקשת ואז כשהמשל להיוריסטיקה
נבחר באפשרות המתאימה.

MCV - נבדוק לאיזה משבצת יש הכי פחות אפשרויות ואותה נבחר.

LCV - נבחר במי שיש הכי פחות אפשרויות (מגביל הכי פחות את האחרים).

A בשאלות של puzzle-9 - המרחק המינ' נספר שינויים בין המצבים
(כל חלק שלא במקום זה 1) המרחק מנהטן - צריך לבדוק כמה כל
מס' צריך להתקדם מרחק מינימלי וסכום הכל.