למידה חישובית – שיעור 3

Entropy

הEntropy הינו מדד נוסף לאי הוודאות הממוצעת של משתנה מקרי כלשהו. בהינתן מערכת עם k מצבים, מדד ה-Entropy יתייחס להסתברות לקבל כל אחד מ-k המצבים. אם נקבל שההסתברויות זהות, לא נוכל להסיק אינפורמציה על המערכת ובעצם לא למדנו כלום, אבל אם יש מאורע שהוא יותר שכיח משאר המאורעות (לדוגמה: יורד גשם באוגוסט), אז נוכל להסיק ממנו יותר מידע.

ככלל, אם נקבל Entropy גבוה, לא נוכל להסיק הרבה לגבי הdata שלנו, ולהפך.

הנוסחה הכללית של האנטרופיה, עבור משתנה מקרי x המקבל n ערכים שונים בהסתברות של p_i

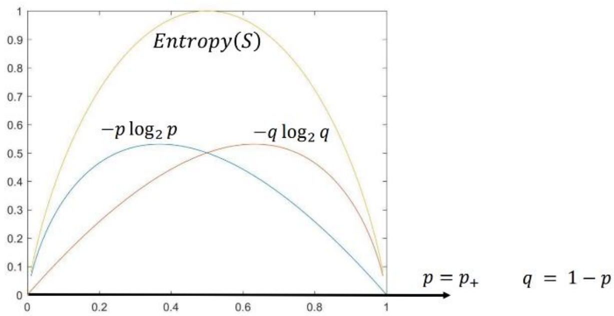
$$H(X) = -\sum_{i=1}^{n} p_i \log(p_i)$$

פונקציית האנטרופיה:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2(p_i) = \sum_{i=1}^{c} -\frac{|S_i|}{|S|}\log_2\left(\frac{|S_i|}{|S|}\right)$$

בדוגמה שלפנינו הEntropy המקסימלי מתקבל כאשר $p = \frac{1}{2}$.

ערכו של הEntropy הינו 1.



$$\begin{aligned} Entropy(S) &= -p\log_2 p + q\log_2 q \\ &= -p_+\log_2 p_+ - p_-\log_2 p_- \end{aligned}$$

נרצה לדעת באופן כללי, עבור איזה ערכים נקבל אנטרופיה מקסימלית

$$entropy\left(\frac{1}{k}\dots\frac{1}{k}\right) = H\left(\frac{1}{k}\dots\frac{1}{k}\right) = -\sum_{i=1}^{k}\frac{1}{k}\log\left(\frac{1}{k}\right)$$

$$H\left(\frac{1}{k}\dots\frac{1}{k}\right) = +\sum_{i=1}^{k}\frac{1}{k}logk$$

$$H\left(\frac{1}{k}\dots\frac{1}{k}\right) = logk\sum_{i=1}^{k}\frac{1}{k}$$

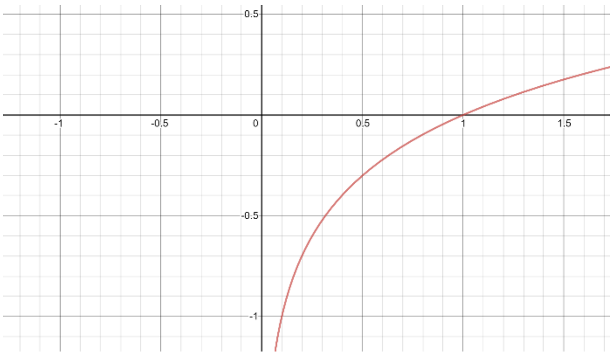
$$H\left(\frac{1}{k}\dots\frac{1}{k}\right) = logk \cdot 1 = logk$$

כלומר מתקיים כי האנטרופיה המקסימלית תתקבל עבור $logk$, כאשר k הוא מספר המצבים האפשריים של המערכת (לדוגמה מספר האפשרויות להטלת קובייה הוגנת הינה $k = 6$).

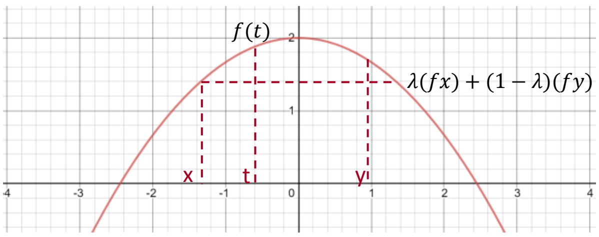
נרצה להוכיח כעת מדוע האנטרופיה מקבלת מקסימום בנקודה $logk$.

נביט בגרף של $logx$.

הגרף מייצג פונקציה קמורה שהנגזרת השנייה שלה שלילית.



נרצה להראות שהתכונה שציינו מתקיימת לכל פונקציה שהנגזרת השנייה שלה שלילית (כל הפונקציות העצובות). ניקח דוגמה יחסית קיצונית לפונקציה מסוג זה:



ניקח נקודה באמצע x,y. אנו יכולים לכתוב את הנקודה t כ:

$$\lambda x+(1-\lambda)y$$

זה בעצם ממוצע משוקלל של x ו-y.

נוכל להסתכל על הנקודות גם כממוצע משוקלל של הפונקציה f על הנקודות:

$$\lambda(fx)+(1-\lambda)(fy)$$

זה ממוצע הערכים.

דרך נוספת להסתכל על הנקודות היא הערך בממוצע

$$f(t)=f(\lambda x+(1-\lambda)y)$$

הערך בממוצע גדול יותר מממוצע הערכים בפונקציות עצובות. הטענה נכונה גם כאשר יש לנו אינסוף נקודות.

טענה: נאמר כי פונקציה היא עצובה אם ורק אם מתקיים:

$$\underbrace{f\ \left(\sum \lambda\ x_i\ \right)}_{\text{הפונקציה } f \text{ בממוצע הנקודות}} \quad \geq \quad \underbrace{\sum \lambda_i\, f\ (\ x_i\ )}_{\text{ממוצע ערכי הפונקציה בנקודות}}$$

אם התכונה מתקיימת לכל ממוצע של נקודות, אזי הפונקציה עצובה.

כעת נרצה לראות היכן הביטוי מקסימלי. נכתוב את האנטרופיה:

$$\sum p_i log p_i \;=\; \sum p_i log\left(\frac{1}{p_i}\right) \;\leq\; \log\left(\sum_{i=1}^{k} p_i \cdot \frac{1}{p_i}\right) \;=\; \log\left(\sum_{i=1}^{k} 1\right) \;=\; logk$$

כלומר, לכל אוסף של k, אנטרופיה קטנה או שווה ל$logk$.

טענה: לכל פונקציה g של המשתנה המקרי x, מתקיים: $H\big(g(x)\big) \leq H(x)$.

אינטואיציה: אם הפעלתי פונקציה כלשהי על הdata שלי, לדוגמה פונקציה המקבלת חיה ונותנת לו ערך אחד (בלבד)- שהוא מספר הרגליים של החיה. תכולת האינפורמציה שאני אקבל מהפונקציה תהיה לכל היותר שווה לתכולת האינפורמציה שאני מקבלת מהמשתנה המקרי שלי. ברוב המקרים אני אקבל פחות אינפורמציה- כי הפונקציה מחזירה ערך אחד ויחיד שלא בהכרח יעזור לי לגלם בתוכו את כל המידע על המשתנה.

הוכחה:

$$\begin{aligned} H(X) &= -\sum_{x} p_X(x)\log p_X(x) \\ &= -\sum_{y}\ \sum_{x:g(x)=y} p_X(x)\log p_X(x) \\ &\geq -\sum_{y} p_Y(y)\cdot \max_{x:g(x)=y}\log p_X(x) \\ &\geq -\sum_{x} p_Y(y)\log p_Y(y) = H(Y) = H\big(g(X)\big) \end{aligned}$$

נגיד את פונקציית הGain של האנטרופיה:

$$\Delta\phi(S,A) = \phi(S) - \sum_{v\in Value(A)} \frac{|S_v|}{|S|}\phi(S_v)$$

$$InfoGain(S,A) = H(S) - \sum_{v\in Value(A)} \frac{|S_v|}{|S|}H(s_v)$$

אלגוריתם ID3 (Iterative Dichotomiser 3)

- בונה בצורה רקורסיבית עץ לפי Information gain.
- ההיפותזה היא עץ.
- מרחב כל ההיפותזות* הוא מרחב כל העצים.
- מרחב ההיפותזות מושלם. כל חלוקה סופית של instanceים במרחב יכולים להיות מיוצגים באמצעות עץ החלטה.
- עצים הינם יקרים, לכן נשאף להשתמש בהם כאשר העץ החזוי קטן יחסית.
- עץ נוקט גישה חמדנית. הוא יבחר את הattribute שייתן את התוצאה הטובה ביותר כרגע.

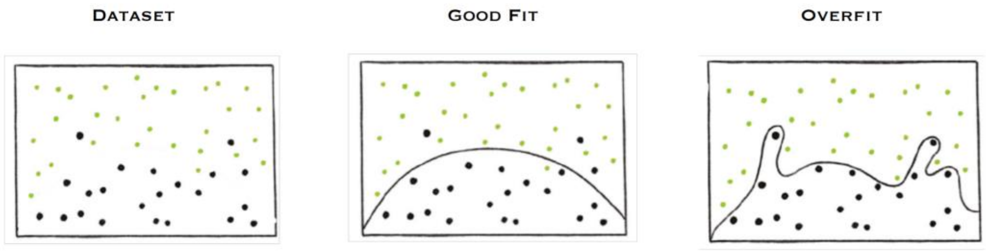
*__מרחב ההיפותזות__ (בהקשרי עצים)- כל הקומבינציות האפשריות של attributes בכל המיקומים.

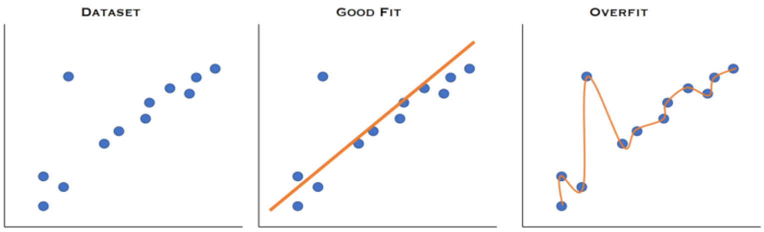
העץ הזה מייצג היפותזת מודל. אנו חושבים שכך העולם מתנהג. נשתמש בהיפותזה הזו לחיזוי ונריץ עליה את אלגוריתם ה- executional.

נסכם את מה שראינו עד כה..

	Gini	Entropy
Impurity	$GiniIndex(S) = 1 - \sum_{i=1}^{c}\left(\frac{\lvert S_i\rvert}{\lvert S\rvert}\right)^2$	$Entropy(S) = -\sum_{i=1}^{c}\frac{\lvert S_i\rvert}{\lvert S\rvert}log\frac{\lvert S_i\rvert}{\lvert S\rvert}$
Goodness of split	$Gini\_Gain =$ $GiniIndex(S) - \sum_{v\in Values(A)}\frac{\lvert S_v\rvert}{\lvert S\rvert}GiniIndex(S_v)$	$Information\_Gain =$ $Entropy(S) - \sum_{v\in Values(A)}\frac{\lvert S_v\rvert}{\lvert S\rvert}Entropy(S_v)$

Overfitting





היפותזה $h \in H$ עושה $overfitting$ ל-data אם קיימת היפותזה $h' \in H$ כך שמתקיים:

$$error_{train}(h) < error_{train}(h') \,\&\&\, error_D(h) > error_D(h')$$

כאשר D זה הdata **בעולם**.

כלומר, נגדיר $Overfitting$ להיות מצב בו הטעות שלנו על ה-$training\ data$ קטנה, אבל הטעות שלנו על ה-$testing\ data$ גדולה. למדנו את ה-$data$ טוב מדי ולכן לא ביצענו הכללה עליו. במצב של $Overfitting$ כל שינוי קטן ב-$training\ data$ יניב שינוי גדול באלגוריתם הלמידה המתקבל. אנו יכולים להגיע למצבים כאלה, כאשר אנו מבצעים למידה מכוונת מדי לdata שבtrain, ולא למידה על ה-data באופן כללי והאלגוריתם שאנו מקבלים יהיה אופטימלי לtraining אבל לא אופטימלי (או אפילו לא טוב) למקרה הכללי. עשינו התאמה טובה מדי לtraining data אבל איבדתי דיוק לtest.

Overfitting יכול לנבוע מכך שה-training data שלי לא שייך למרחב ההיפותזה, ואז בעצם למדתי על data שאינו מייצג את ה-data שעליו אני רוצה להפעיל את האלגוריתם. סיבה נוספת ל-overfitting הינה רעש של הניסוי או ה-data (ה-data לא מסודר, לא מייצג תמונה נקיה של המציאות וכו').

אם כן, כיצד נוכל להימנע ממצב של *Overfitting*?

נוסיף data נוסף: $Validation\ Set$ שהוא זר ל-$training$ ונבדוק עליו את הטעות, כך שאנו לומדים ע"י ה*training* ומודדים את הטעות על ידי ה*Validation*. נעצור את ריצת האלגוריתם הלומד, כאשר ה-Validation Error גדל.

כיצד נימנע ממצב של Overfitting בעצים?

1. נפסיק "לגדל" את העץ לפני שהגענו לעץ המלא (לפני שיש 0 טעויות). נקרא: $pre\ pruning$.
2. נגדל עץ מלא, ואז נקצץ את העלים שיצרו לנו error. נקרא $post\ prune$.
3. נשלב בין pre pruning ו-post pruning.

הערה: מדוע להעדיף לגדל עץ מלא ואז לגזום? נשמע בזבזני לגדל עץ ואז לגזום אותו. הסיבה שנשתמש בשיטה זו היא להימנע מ*underfitting*, כלומר לוותר על ענפים שחשובים לנו ואנו זקוקים להם.

כיצד נדע מתי להפסיק לגדל את העץ?

1. כאשר המצב שלנו לפני הפיצול ולאחריו זהים.
2. כאשר ה-*validation* מניב לנו דיוק נמוך יותר.

כיצד נדע מתי ואיזה ענפים לקצוץ מהעץ?

Chi Square Measure

מבחן האומר לנו האם חלוקה לפי *attribute* מסוים נותנת לנו פיזור רנדומלי או שלפיזור שהתקבל יש יכול ניבוי כלשהי. נבדוק האם הפיצול שהתקבל בהתאם לattribute הנבחר הניב לנו פיזור הדומה לפיזור רנדומלי לחלוטין.

$$\chi^2 = \sum_{f \in Values(x_j)} \frac{\left(p_f - E_0\right)^2}{E_0} + \frac{\left(n_f - E_1\right)^2}{E_1}$$

כך ש:

- $\underline{\boldsymbol{P(Y=0)}}$- ההסתברות להוצאת *instance* עם class value=*0* מהקבוצה הנוכחית (לפני הפיצול).
- $\underline{\boldsymbol{D_f}}$ כמות ה-instances שערך ה-attribute שלהם הוא - *f*.
- $\underline{\boldsymbol{p_f}}$ כמות ה-instances **מתוך** D_f שערך ה-class value שלהם הוא 0.
- $\underline{\boldsymbol{n_f}}$ כמות ה-instances מתוך D_f שערך ה-class value שלהם הוא 1.
- $E_0 = D_f P(ClassValue = 0), E_1 = D_f P(ClassValue = 1)$.

נחשב את ערך ה$Chi\ square$ עבור כל $attribute$. נציב בטבלה:

- שורות הטבלה: נחשב את מספר ה*attributes* פחות 1, כמספר דרגות החופש
- עמודות הטבלה: נקבע α שיהיה רמת הביטחון שלנו, כלומר כמה בטוחים אנו רוצים להיות.

כעת, אם הערך בטבלה גדול מהערך אותו חישבנו - לא נמשיך לפצל לפי ה-*attribute* הנוכחי, כי הוא לא מניב לנו מידע נוסף.

Cross Validation

עקרון של עבודה של 2 training set, העוזר לנו להימנע ממצב של Overfitting.

- מעריך את הדיוק של ההיפותזה המתקבלת באמצעות האלגוריתם. ינבא את הדיוק של ההיפותזה עבור *data* עתידי שטרם ראינו.
- יבחר את ההיפותזה האופטימלית מתוך קבוצת ההיפותזות
  - גיזום בעצי החלטה
  - בחירת מודל ההחלטה – ליניארי או פולינום ממעלה גבוה
  - בחירת מספר ה-featureים עליהם נלמד. נזכור כי בהגדרה, ככל שנעלה את המימד ב*training* נקבל דיוק גבוה יותר, אך אנו בהכרח נקבל ב-*test* דיוק נמוך ולא מדויק.
- שילוב של מספר classifiers.

נשים לב של-$Cross\ validation$ יש תפקיד חשוב בהבנה של הדיוק של המודל שהתקבל.

הערות נוספות..

1. התמודדות עם ערכים חסרים

אם יש לנו $data\ points$ שאנו לא יודעים מה הסיווג שלהן (Missing values), ננחש את הערכים שלהן כך שהן יקבלו את הערך השכיח ביותר ב-data (לדוגמה: אם יש לי 20 ערכים של גבוה, 5 ערכים של בינוני, 5 ערכים של נמוך וערך 1 שאינו ידוע, ניתן לערך הלא ידוע ערך גבוה).

נשלים את הערך החסר לפי הרוב.

2. עבודה עם ערכים רציפים

אם יש לנו ערכים רציפים, נשתמש בסף (threshold) שיגדיר לנו ערך בינארי לשאלה האם הinstance עונה או לא על שאלת הfeature.

3. פיצול המידע וה-$\boldsymbol{Gain\ Ratio}$ לattributes עם הרבה ערכים

הGain Ration הוא תחליף ל-Gain. אם יש attribute כלשהו שיש לו הרבה ערכים, הGain שלו יהיה בדרך כלל גבוה יותר ולכן האלגוריתם יבחר בו (כמעט) תמיד.

הGain Ration עוזר לנו להימנע ממצב זה על ידי כך שאנו מתייחסים לקבוצת המופעים ביחס לattribute (ולא רק לערך הGain). נגדיר פונקציה חדשה שנמצא באמצעותה איזה attribute מניב לנו Gain גבוה ביותר, כאשר אנו מצליחים לנטרל את בעיית המופעים שתיארנו:

$$GainRatio = \frac{Gain(S,A)}{SplitInformation(S,A)} = \frac{Gain(S,A)}{Entropy(S,A)}$$

4. הכנסת ה-$\boldsymbol{Cost}$ של attributes

יש שאלות שאין להן את אותו ה"מחיר". אנו מעוניינים לדעת כמה עלה לי למדוד את הattribute , לדוגמה בבדיקה רפואית MRI יקר יותר מבדיקת רנטגן. אנו מעוניינים לכלול את עלות המדידה של הattribute בתוך הGain שלנו:

$$GainWithCost_1 = \frac{Gain^2(S,A)}{Cost(A)}$$

$$GainWithCost_2 = \frac{2^{Gain(S,A)} - 1}{(Cost(A)+1)^w} \quad where\ w \in \{0,1\}\ \text{קביעת חשיבות}$$

5. גבולות מורכבים

גבולות ההחלטה שנקבל מהעץ יראו כמו "תיבות", אך לא בהכרח תמיד נגיע לתיבות שכאלה (כמו בציור שמשמאל). לתיבות יהיה קשה להגיע למישור (לא בלתי אפשרי, אבל קשה).