

למידת מכונה

רוב העבודות הקשורות ללמידת מכונה מבוססות על יוריסטיקות - כלומר אין אפשרות לחסום את הטעות, אלא רק לעשות ניסויים ולהראות סטטיסטית שהמודל טוב. מאמנים מודל על training data ובודקים אותו על testing data ומחשבים את ההסתברות שהוא צודק. למרות שיש אלמנט יוריסטי, עדיין יש תיאוריה של למידה. התיאוריה נקראת PAC - Probability Approx- imately Correct Learning. אנחנו רוצים אלגוריתמים שבהסתברות גבוהה, יתנו לנו היפוטזות שבהסתברות גבוהה יתאימו למודל החבוי.

אלגוריתם למידה

X סט הדוגמאות האפשריות
 D התפלגות על X שבאמצעותה אנחנו דוגמים דוגמאות
 H קבוצת ההיפוטזות. יש פונקציה $f \in H$ שאותה אנחנו רוצים ללמוד. למשל: האם בן אדם ישלח הודעה שוב או לא?

אלגוריתם הלמידה יחזיר פונקציה $h \in H$. אנחנו רוצים $h \approx f$ - כלומר שהפונקציה שנקבל תהיה מאוד דומה ל- f . אנחנו רוצים $h(x) = f(x)$, אבל זה לא תמיד אפשרי בכל המקרים, ולכן נרצה:

$$P\{h(x) \neq f(x) | x \in X \text{ distributes by } D\} \leq \varepsilon$$

כלומר

$$\text{error}(h) \leq \varepsilon$$

את ההיפוטזות האלה נסמן ב- H_{good} - כדור- ε מסביב ל- f . שאר ההיפוטזות הן H_{bad} . נרצה שההסתברות לקבל $h \in H_{\text{good}}$ תהיה טובה

המטרה: למצוא אלגוריתם למידת מכונה כך ש- $P(h \in H_{\text{bad}}) \leq \delta$

והשאלה היא - כמה דוגמאות אנו באמת צריכים כדי לקבל הסתברות כזו?

אם יש לנו m דוגמאות - מה ההסתברות לקבל $h \in H_{\text{bad}}$ כאשר עבור כל m הדוגמאות $h(x) = f(x)$. ההסתברות ש- h_{bad} כלשהי תסכים עם f על דוגמה קטנה מ- $1 - \varepsilon$ (נובע מהגדרת H_{bad}), ולכן עבור m דוגמאות:

$$P(h_{\text{bad}}(x) = f(x) \text{ for } m \text{ samples}) \leq (1 - \varepsilon)^m$$

אבל זו ההסתברות עבור היפוטזה גרועה אחת - ויש לנו הרבה $h \in H_{\text{bad}}$! מה ההסתברות שנקבל אחת מהן? אנחנו רוצים לחסום את זה עם δ :

$$P(H_{\text{bad}} \text{ contains a bad hypothesis that agrees on the } m \text{ samples}) \leq |H_{\text{bad}}| (1 - \varepsilon)^m \leq |H| (1 - \varepsilon)^m \leq \delta$$

ולכן מספר הדוגמאות שאנו צריכים הוא:

$$m \geq \frac{1}{\varepsilon} \left(\ln \frac{1}{\delta} + \ln |H| \right)$$

ϵ, δ קבועים, ולכן מה שמעניין כאן זה $|H|$. כמה היפוטזות יש לנו?
אם ניקח את הפונקציות הכי פשוטות, שמקבלות קלט באורך n ומחזירות תשובה חיובית או שלילית - כלומר
 $H \ni f : \{0, 1\}^n \rightarrow \{0, 1\}$. יש לנו 2^n קלטים אפשריים, ולכן מספר ההיפוטזות האפשריות הוא 2^{2^n} ! זה אומר
ש $\ln |H| = \ln 2^{2^n} \in O(2^n)$.
מספר הדוגמאות שאנחנו צריכים הוא אקספוננציאלי - ולכן אנחנו נאלצים להסתמך על יוריסטיקות.

פתרונות אפשריים

- להקטין את מרחב ההיפוטזות. למשל, במקום לייצג פונקציה בוליאנית באמצעות טבלה, לייצג אותה באמצעות \wedge של ביטויים חיוביים ושליליים, ואז יש לנו רק 2^n דוגמאות.
- לדוגמה - perceptrons משתמשים בשיטה הזו. הבעיה היא שאם הפונקציה לא ניתנת להצגה טובה בצורה לינארית אז נקבל פונקציה גרועה.
- צריך תמיד לזכור שרוב הבעיות המעניינות במדעי המחשב הן NPC - ובמציאות מספיק קירוב טוב בשביל לקבל דברים מועילים.

Reinforcement Learning

כל הדוגמאות שהיו לנו עד עכשיו היו מתוייגות - ידענו מה התשובה הנכונה לכל דוגמה. אבל הרבה פעמים במציאות אין לנו דוגמאות - אנחנו צריכים ללמוד מתוך התנסות.

נשים ♡ בניגוד ל־Unsupervised Learning, שם יש לנו דוגמאות לא מתוייגות, Reinforcement Learning אין דוגמאות, ואנחנו צריכים ליצור לבד את הדוגמאות שלנו, ולקבל עליהן תמורה חיובית או שלילית.

אנחנו צריכים לאסוף לבד את הדוגמאות - ואיסוף הדוגמאות עולה לנו!

תזכורת - MDP (Markov Decision Process)

$$MDP = \langle S, A, T, R \rangle$$

כאשר:	S	מצבים (States). העולם תמיד נמצא באיזשהו מצב
	A	פעולות (Actions) שהסוכן יכול לבצע
	T	Transition (לפעמים P - Probabbility). מה ההסתברות שהסוכן יהיה במצב $s \in S$, ביצע פעולה $a \in A$ והגיע למצב $s' \in S$ - $T(s, a, s') : S \times A \times S \rightarrow [0, 1]$
	R	גמול (Reward). אפשר להשתמש ב־ $R(s)$, ב־ $R(s, a)$ או ב־ $R(s, s', a)$ (כי לפעמים יש עלות לפעולה, ולפעמים רוצים להתחשב במצב הקודם).

אנחנו נרצה למצוא אסטרטגיה $\Pi : S \rightarrow A$ שתהיה אופטימלית - כלומר כזו שתמקסם את ה־expected utility - את ה־reward שנקבל על כל פעולה ברצף הפעולות. אם היתה נתונה לנו סדרה סופית של מצבים היינו יכולים לחשב את ה־expected utility:

$$Eu(s_1, s_2, s_3, \dots) = \sum_{s_i} R(s_i)$$

בפועל אנחנו לא יודעים מתי הסדרה תסתיים - היא בכלל יכולה להיות אינסופית! לכן בד"כ מניחים ש"תכף העולם יגמר".

Bellman Equation

יש הסתברות $\gamma \leq 1$ שהעולם ימשיך - שנוכל לבצע עוד פעולות. לכן התועלת של הסוכן משימוש ב־ Π היא:

$$u^\Pi(s) = R(s) + \gamma \sum_{s'} T(s, \Pi(s), s') u^\Pi(s')$$

כאשר אין לנו Π ספציפי, אפשר להסתכל על התמורה המקסימלית שהסוכן יכול לקבל:

$$u(s) = R(s) + \gamma \max_{a \in A} \sum_{s'} T(sa, as') u(s')$$

זה בעצם התמורה מ־ Π אופטימלית.

- הערות: • בספרי כלכלה, γ משתמש בשביל להעדיף כסף היום על פני כסף מחר.
- מבחינה מתמטית, γ משתמש כדי לגרום לסדרות האינסופיות להתכנס.

זו בעיה מאוד קשה, כי זו לא בעיה לינארית וצריך לעבור על הרבה אפשרויות.

נחזור לRL

מה קורה אם אנחנו לא יודעים את הMDP?

- אפשרות נאיבית - לשלוח את הרובוט לאסוף דוגמאות. אחרי שהוא אוסף הרבה סדרות של פעולות אפשר לעשות ממוצעים על כל הפעולות שהוא עשה, ולראות מה הכי טוב.

אפשר ככה להשוות בין Π_1 ל Π_2 - לשלוח אותו לעשות את שניהם ולהשוות את הutility.

יתרונות: - קל לחישוב

- לא צריך לעשות דברים מיוחדים

חסרונות: - זה בעצם סוג של supervised learning