למידה חישובית - שיעור 1

Machine Learning

זהו תחום במדעי המחשב העוסק בתהליך הלימוד של המחשב לבנות אלגוריתם בעצמו, על ידי data ודוגמאות שהוא מקבל (דוגמאות חיוביות המגדירות מה נכון, ודוגמאות שליליות המגדירות מה שגוי). יהיו 3 מקרים עיקריים בהם ניעזר באלגוריתמים של למידה חישובית:

1. המשימה מורכבת והפתרון שיתקבל מאלגוריתם רגיל יהיה לא טוב (או לא מספיק טוב). דוגמה: בעבר כשניסו לייצר אלגוריתם לזיהוי פנים הוא הצליח לזהות רק ב25% מהמקרים.
2. כאשר נרצה לייצר בינה מלאכותית. ישות מלאכותית צריכה להיות בעלת יכולת למידה (בינה) כך שתוכל ללמוד ולהגיב למציאות המשתנה.
3. כמות הdata עצומה ולא ניתן לעבד את הdata בזמן ריאלי. לדוגמה עיבוד המידע על ידי אדם עלול לארוך 1,000 שנים ועל ידי מכונה מספר חודשים.

אלגוריתם של למידה חישובית מקבל מידע (data) רב ודוגמאות ובעזרתן הוא לומד מה עליו לגלות. לדוגמה באלגוריתם לזיהוי פנים האלגוריתם ילמד על ידי הפיקסלים של כל תמונה.

לרוב, False Positive ייחשב גרוע יותר מFalse Negative.

הערכת ביצועים

לאחר שלב הלמידה, נרצה לדעת האם האלגוריתם שלנו למד בצורה טובה. נבדוק את איכות הלמידה של האלגוריתם על ידי מתן data זר (כזה שלא הוצג לאלגוריתם בשלב הלמידה) ונבחן האם האלגוריתם מצליח לבצע עליו את פעולותיו.

טרמינולוגיה:

$data$– כל המידע ממנו אנו לומדים

$instance$ – מופע. הdata מורכב ממופעים. לדוגמה באלגוריתם לזיהוי פנים מתמונה, כל תמונה מהdata הינה instance.

כל instance יסומן בx_i כאשר x_i הוא וקטור.

$feature$– תכונה. לכל instance יש לכל הפחות feature אחד. לדוגמה באלגוריתם לזיהוי פנים לכל תמונה יש מספר featureים כמו לדוגמה צבע עיניים, צבע שיער וכו'. נשאף תמיד להבין איזה featureים חשובים יותר ואיזה פחות. לדוגמה: צבע עיניים הוא feature חשוב ואילו צבע חולצה הינו feature לא חשוב. נרצה לנתח את הdata שלנו באמצעות הfeatureים החשובים ביותר, כאלה שיתנו לנו אינדקציה טובה ביותר לשאלה האם הinstance שקיבלנו עונה על שאלת האלגוריתם.

אנו נבצע תהליך סיווג על הdata שלנו שבמהלכו ננסה להבין האם הinstance שיתקבל עונה על שאלת האלגוריתם, לדוגמה האם התמונה שהתקבלה אכן מזוהה כפנים של אדם כלשהו שאנו מחפשים. לתהליך הזה אנו קוראים $labeling$. אם תמונה זוהתה, נאמר שהיא labeled.

לlabel של הdata שאותו אנו לומדים נקרא y_i.

ככלל נאמר שמספר הfeatureים הוא n ומספר הinstanceים הוא m.

כלומר, יש לנו m וקטורי x_i וכל וקטור באורך n, ולכן ניתן להסתכל על הdata שלנו כעל **מטריצה** $n \times m$.
כל עמודה במטריצה תייצג feature כלשהו של כל הinstanceים בdata.

<u>יש מספר סוגים של אלגוריתמי למידה חישובית:</u>

supervised Learning

למידה בה יש כביכול מורה המלמד אותנו לסווג את הdata שלנו. תהליך הסיווג נקרא כאמור labeling.
כלומר מה שמייחד את הsupervised learning הוא שלכל instance יש label.

רגרסיה

ננסה למצוא פונקציה המקיימת $y_i = f(x_i)$ כלומר ננסה למצוא קשר בx_i לy_i

המידע שנעבד באלגוריתמי רגרסיה הינו רציף.

קלסיפיקציה

סיווג הקלט ל"מחלקות" של תכונות. באלגוריתמי קלסיפיקציה נחפש תשובה לשאלה האם הdata מקיים או לא תכונה כלשהי. נשים ♥ לכך שלעיתים קרובות תהיה יותר מתכונה אחת שתעניין אותי. (צבע עיניים, צבע שיער וכו').

הערה: אלגוריתם קלסיפיקציה יכול להיות כזה שהתשובה לשאלה היא כן או לא (האם התמונה היא אובמה או לא), או כזה שיש לו אינסוף תשובות (מי/מה נמצא בתמונה שהתקבלה).

supervised Learning

באלגוריתמים מסוג זה הקלט של הלמידה הוא הdata בלבד. האלגוריתם יקבל את הdata והוא יחפש דפוסים ותבניות מעניינות. באלגוריתם זה אנו לא "מלמדים" כיצד לסווג את הdata, בעיקר כי ברוב המקרים אנו לא יודעים מה אנו מחפשים או איך מה שאנו מחפשים אמור להיראות.

clustering

נשתמש באלגוריתמים מסוג זה כאשר אין באמת שאלה, אלא רצון למצוא "משהו מעניין" בתוך הdata.

Density Estimation

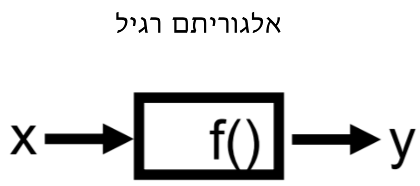
הכללה של clustering. נחפש איפה יש לנו אזור של "דברים שכיחים" ואזור של "דברים נדירים". לדוגמה יש פחות בנות במדעי המחשב, לכן בנות הינן שכיחות בקרב סטודנטים במסלול מדעי המחשב (דוגמה גרועה, תאשימו את אריק).

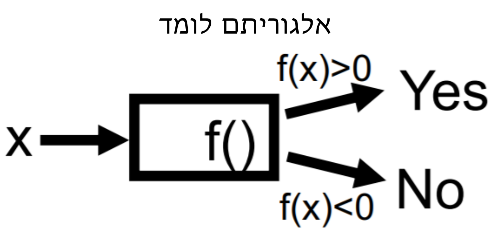
אם כן נרצה לחדד את **ההבדל בין אלגוריתם רגיל לאלגוריתם לומד**

באלגוריתם לומד, האלגוריתם הוא זה שייצר לך את הפונקציה שבאמצעותה תבצע את החישוב הרצוי- וכן באמצעות הdata ניתן לקבל את תוצאת החישוב הרצוי.

בעוד שבאלגוריתם רגיל אתה זה שצריך לייצר את החישוב ובאמצעות החישוב תקבל את תוצאת החישוב.

הערה: לעיתים לא נדע מהו האלגוריתם שקיבלנו מתהליך הלמידה החישובית. האלגוריתם יעבוד בצורה תקינה, ונקבל את הפלט הרצוי, אך אנו לא נדע בהכרח כיצד האלגוריתם מבצע את החישובים שלו.





מרחב ההיפותזה (רגרסיה לינארית)

מרחב הפונקציות הלינאריות. באלגוריתם ללמידה חישובית אנו מחפשים פונקציה שתקשר בין הקלט לפלט ונרצה לצמצם את הבעיה לפונקציות לינאריות בלבד.

נרצה לבחור את הפונקציה הלינארית הטובה ביותר* **באמצעות האלגוריתם הלומד** ולכן נחפש היפותזה (פונקציה לינארית כלשהי) שמהווה את הפונקציה הלינארית הטובה ביותר. יש לנו אינסוף פונקציות לינאריות.

<u>פונקציית היפותזה עם feature אחד:</u>

$$y = \theta_0 + \theta_1 x$$

כאשר θ הוא סקלר.

<u>פונקציית ההיפותזה עם מספר רב של featureים:</u>

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

ומתקיים $x_0 = 1$.

את ערכי $\theta_0, \theta_1, \dots, \theta_n$ האלגוריתם צריך למצוא.

<u>*הפונקציה הלינארית הטובה ביותר:</u>

נרצה למצוא את האלגוריתם שייתן לי $f(x_i) = y_i$. מציאת הפונקציה שתקיים את זה בדיוק על ידי אלגוריתם לומד אינה קלה, ולכן אנו נשאף למצוא את הפונקציה שתתן ערך מינימלי עבור

$$\sum_i [f(s) - y_i]$$

נשים לב שייתכן שנקבל ערכים שליליים ולכן נעלה את הכל בריבוע על מנת לקבל ערך חיובי (לא נעשה ערך מוחלט כי נקבל פונקציה לא רציפה ולא גזירה) ולכן:

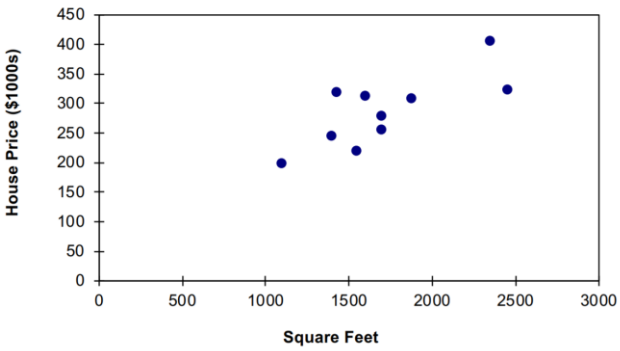
$$\boxed{\sum_i [f(s) - y_i]^2}$$

וזוהי פונקציית ריבוע הטעות.

דוגמת תמחור בית:

נרצה לתמחר בית, בהתבסס על הfeatureים השונים שלו, לדוגמה גודל הבית, מיקום, מספר החדרים וכו'. נרצה למצוא פונקציה שתוכל לחשב לנו את מחיר הבית בהינתן כל הfeatureים שלו.

נניח לשם הדוגמה שהfeature היחיד של הבית הינו גודלו. נחפש פונקציה $f(x) = y$ המקבלת גודל בית x ומחזירה כפלט את מחירו.



כל נקודה באיור הינה נקודה (x_i, y_i). נרצה למצוא פונקציה שעוברת בין כל הנקודות הללו, אבל ניתן לראות שלא מדובר בפונקציה! כיוון שאותו x קיבל 2 y‏ים שונים. לכן נסתפק בפונקציה שתקרב אותנו לתוצאה המדויקת.

מרחב ההיפותזות שלי היא הפונקציות הלינאריות. כלומר אנו לא מחפשים כל פונקציה שבעולם אלא רק פונקציות לינאריות בלבד. הסיבה לכך היא שכל פונקציה היא סוג של היפותזה. כל אחת מהפונקציות הלינאריות היא סוג של היפותזה, ואנו נרצה לבחור מתוך כל פונקציות ההיפותזה הטובה ביותר.

נגדיר פונקציה לינארית עבור feature אחד:

$$y = \theta_0 + \theta_1 x_1$$

כאשר θ הוא סקלר.

אם נרצה לייצג featureים רבים כמו גודל דירה, מיקום ועוד, נגדיר את הפונקציה הלינארית הבאה:

$$y = \theta_0 + \theta_1 x_1 + \cdots x_n$$

פונקציית המחיר Cost Function

נחפש את ה- θות שימזערו את הטעות בחיזוי שלנו. נרצה להקטין את פונקציית המחיר המורכבת מממוצע ההפרשים בין הפרדיקציה לבין הערך האמיתי:

$$J(\boldsymbol{\theta}) = \frac{1}{2}\frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}\cdot\mathbf{x}^{(\mathbf{i})} - \mathrm{y}^{(\mathrm{i})}\right)^2$$

הפונקציה נקראת MSE- Mean Square Error.

<u>הערה 1:</u> לעיתים המרחק יהיה מעל המישור, ולעיתים מתחתיו. אם המרחק יהיה מתחת למישור- נקבל שגודל המרחק שלילי, אך אנו מתעניינים רק בגודלו ולכן אנו לוקחים את ההפרש- ומעלים בריבוע. אנו לא משתמשים בערך מוחלט כי פונקציית ערך מוחלט אינה גזירה.

<u>הערה 2:</u> אנו כופלים ב$\frac{1}{2}$ על מנת שנוכל לגזור את הפונקציה מאוחר יותר בייתר קלות.

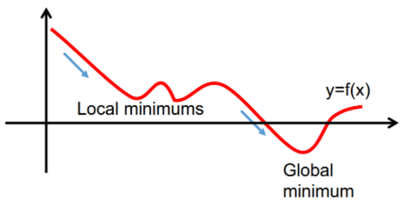
זוהי פונקציה של θ (כאשר θ בעצמה היא וקטור של כל ה-θות).

אנו מחפשים את הθ שתביא את $J(\theta)$ לערך המינימלי, כאשר $J(\theta)$ הטוב ביותר שווה ל-0. נרצה למצוא את ה- θות שמקיימות את זה. נוכל לבחור בצורה רנדומלית, אבל קיימות אינסוף θות ולכן נרצה למצוא פתרון טוב יותר:

מציאת המינימום

נרצה למצוא את θ_0, θ_1 שמביאה את פונקציית העלות למינימום, כלומר $\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$.

עד היום גזרנו את הפונקציה והשוונו ל-0 על מנת למצוא נקודת מינימום, אך לא נוכל להשתמש בטכניקה זו בפונקציה עם ריבוי משתנים, כי אנו **עלולים למצוא מינימום לוקלי ולא מינימום גלובלי.** הרעיון הכללי של הפתרון הוא להתחיל בנקודה כלשהי ולהמשיך להתקדם כל עוד אנחנו במגמת ירידה, כלומר נבחן היכן הנגזרת של הפונקציה יורדת, ונרד עמה.



אנו עלולים להיתקל במינימום לוקלי ולחשוב שמדובר במינימום גלובלי ולכן על מנת להימנע מהישנות מקרים אלה נבחר מספר נקודות ונבצע את תהליך הירידה ממספר נקודות בו זמנית.

נגזרת כיוונית

נמצא את המינימום של פונקציית המחיר באמצעות נגזרת כיוונית

$$D_u f(x_1, x_2) = \lim_{s\to 0}\frac{f(x_1 + su_1, x_2 + su_2) - f(x_1, x_2)}{s} = \left(\frac{df}{ds}\right)_u$$

נרצה למצוא את הכיוון שבו הירידה הכי גדולה. אם אין כזה כיוון- אני נמצאת במינימום הגלובלי. אם יש- אז בהכרח יש מינימום גלובלי אחר.

גרדיאנט

נוכל להסתכל על הנגזרת הכיוונית גם כסכום של גזירה בכיוון של x_1 וגזירה בכיוון של x_2:

$$D_u f(x_1,x_2)=\left(\frac{df}{ds}\right)_u=\frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s}+\frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s}=$$
$$=\frac{\partial f}{\partial x_1}u_1+\frac{\partial f}{\partial x_2}u_2=\left(\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2}\right)\cdot(u_1,u_2)=\nabla f(x_1,x_2)\cdot \mathbf{u}$$

אנו מסתכלים על הנגזרות ומכפילים בכל אחד מהכיוונים העיקריים (של הוקטור).

קיבלנו את הנגזרת לפי x_1 והנגזרת לפי x_2 כפול 2 הרכיבים של u (הכיוון שבמקור רציתי לגזור לפיו. יכול להיות כל וקטור).

הביטוי שקיבלנו נקרא **הגרדיאנט** - וקטור שיש בו 2 נגזרות חלקיות ל-2 כיוונים (ובאופן כללי - אם אנחנו במימד x אז x נגזרות חלקיות לx כיוונים).

גרדיאנט - כל הנגזרות החלקיות, בתוך וקטור:

$$\nabla f=\left(\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2}\right)$$

וברב מימד הגרדיאנט יראה כך:

$$\nabla f=\left(\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\ldots,\frac{\partial f}{\partial x_n}\right)$$

וקטור הגרדיאנט בכל נקודה נתונה הוא בכיוון בו יש עלייה הכי גדולה, או במילים אחרות: הירידה הכי גדולה היא מינוס הגראדיאנט:

$$-\nabla f(x_1,x_2,\ldots,x_n)$$

Gradient Descend

כיוון הגראדיאנט הוא: $\beta=0$ ואז $\cos\beta=1$ ולכן, נגד הכיוון הוא : $\beta=180°$ ואז $\cos\beta=-1$

ובעצם באלגוריתם זה, הולכים נגד כיוון הגראדיאנט עד שמגיעים למינימום של פונקציית המחיר.

תרגול 1

חישוב הנגזרות החלקיות של פונקציית המחיר

$$\frac{\partial}{\partial\theta_j}J(\theta_0,\dots,\theta_n)=\frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(\theta x^{(i)}-y^{(i)}\right)^2$$

$$=\frac{1}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta_j}\left(\theta_0+\theta_1x_1^{(i)}+\dots+\theta_jx_j^{(i)}+\dots+\theta_nx_n^{(i)}-y^{(i)}\right)^2$$

נבצע את הנגזרת לכל אחד מן ה-θ.

$$=\frac{1}{2m}\sum_{i=1}^{m}2\left(\theta_0+\theta_1x_1^{(i)}+\dots+\theta_jx_j^{(i)}+\dots+\theta_nx_n^{(i)}-y^{(i)}\right)*x_j^{(i)}$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left(\theta_0+\theta_1x_1^{(i)}+\dots+\theta_jx_j^{(i)}+\dots+\theta_nx_n^{(i)}-y^{(i)}\right)*x_j^{(i)}=\frac{1}{m}\sum_{i=1}^{m}\left(h_0(x^{(i)})-y^{(i)}\right)*x_j^{(i)}$$

שימוש בגרדיאנט בתוך אלגוריתם:

1. בחירה של θ_0,θ_1 רנדומליים כלשהם
2. בצע את העדכונים הבאים:
   a. $\theta_0 := \theta_0-\alpha\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)=\theta_0-a\frac{1}{m}\sum_{i=1}^{m}(\theta_0+\theta_1{x_1}^i-y^i)$
   b. $\theta_1 := \theta_1-a\frac{1}{m}\sum_{i=1}^{m}\left(\theta_0+\theta_1{x_1}^i-y^i\right)\cdot{x_1}^i$
3. חזור על שלב (2) עד אשר הטעות קטנה דייה.

את העדכונים נבצע באופן סימולטני:

$$temp0=\theta_0-\alpha\frac{\partial}{\partial\,\theta_0}J(\theta_0,\theta_1)$$

$$temp1=\theta_1-\alpha\frac{\partial}{\partial\,\theta_1}J(\theta_0,\theta_1)$$

$$\theta_0=temp\ 0;\theta_1=temp1$$

- When we are to the right of the minimum, gradient is positive, so we move left
- When we are to the left of the minimum, gradient is negative, so we move right
- When α is too small, what happens?
- When α is too big, what happens? (possible to diverge)