שיעור 1

7 בפברואר 2019

1 בסיס ללמידה חישובית

1.1 הגדרת עולם הבעיה

במחקר מידע אנו מעוניינים להשתמש במידע הקיים לרשותנו כדי להפיק תובנות וכדי לחזות התנהגות של מידע עתידי. הדבר דומה במובן מסוים לעבודתו של הפיזיקאי (ממש כמו בקורס במעבדה): עלינו להתבונן בתוצאות ה`ניסוי` או המדידה, ולנסח מודל שיאפשר תחזיות של ניסויים דומים. בבנייה או ניסוח של מודלים תמיד נצטרך לתת את הדעת לשלושת המאפיינים הבאים:

1. הפונקציה המתארת את המודל (מיד על כך בהמשך)
2. הדרך להערכת המודל
3. התפלגות המידע המעניין אותנו

נתחיל מהעיסוק בפונקציה של המודל ונפרמל את הרעיון בצורה מתמטית:
אל המידע נתייחס כאל וקטורים במרחב וקטורי (או לשם הנוחות, כאל סדרות של מספרים). נניח שכל וקטור הוא באורך n, כך שהמידע שלנו הוא למעשה נקודות ב־$\mathbb{R}^n$. נהוג להתייחס אל היצוג הוקטורי כאל ה"פיצ'רים" של המידע. נסמן את מרחב המידע ב־X, כאשר כל נקודה תסומן ב־

$$x = (x_1, ..., x_n)$$

באופן כללי נחשוב בקורס על וקטורים כעל וקטורים במרחב אוקלידי, או כעל n־יות של מספרים. נניח שאנו רוצים לקבל `תחזית` עבור כל וקטור ־ למשל, מספר ממשי כלשהו. את מרחב התחזיות נסמן ב־Y. תחזיות תהיינה לרוב מספריות, אך טווח התחזיות עשוי להשתנות: לעתים נחזה ערך "רציף", $Y = \mathbb{R}$. בעיות שכאלו יקראו בעיות רגרסיה, בעוד לעתים נרצה לחזות הסתברות לאירוע, כך ש־$Y = [0, 1]$. בבעיות סיווג, classification נחזה class, או קטגוריה, כך ש־$Y = \{class\_1, ..., class\_n\}$.

עלינו למצוא פונקציה $f : X \to Y$ שלכל וקטור תתן את התחזית עבורו. פונקציה זו היא ה`מודל`. מעתה כשנתייחס למודלים, זכרו כי מודל הוא בסך הכל פונקציה פשוטה. עם זאת, מודלים לא מגיעים מהשמים, ונצטרך לעבוד קשה כדי לקבל מודלים טובים (ולהבין באיזה מובן הם טובים).

supervised learning הוא תת־תחום גדול למדי בלמידת מכונה, בו נעבוד עם מידע מתויג. כלומר, חלק מהמידע שלנו יגיע מתויג עם התחזית הרצויה. אם אוסף המידע שלנו הוא זוגות מהצורה $(x_i, y_i) \in X \times Y$, אז מודל טוב יהיה מודל עבורו מתקיים

$$f(x_1) = f((x_{11}, ..., x_{1n})) = y_1$$

נעסוק בשיעור הקרוב בהתמודדות עם בעיות מסוג זה. נגדיר דרכים להעריך תוצאות של מודלים. אחר כך נכיר כמה מודלים פשוטים להתמודדות עם משימות supervised. בהמשך נכיר כמה מודלים מורכבים יותר, ואת הקשיים המתלווים לאימון שלהם. נעסוק בתור התחלה בבעיות רגרסיה, בהן אנו חוזים ב־Y ערך ממשי כלשהו.

1.1.1 דוגמא

באיזה אופן מידע הוא וקטורים? פעמים רבות נאלץ להשקיע מאמץ כדי לנסח בעיות במובן זה, בתהליך שנקרא feature extraction, בו אנו מפיקים וקטורים מתוך המידע הבסיסי. כלומר, אנו מתארים את המידע באמצעות וקטורים. למשל, אם המידע שלנו כולל אוסף של דירות, נוכל לתאר את הדירות הללו באמצעות תכונות מספריות:

$$Appartment \mapsto (\underset{size\,(m^2)}{120}, \underset{\#rooms}{5}, \underset{floor}{5}, \underset{\#windows}{3}, ...)$$

נרחיב על כך עוד בהמשך. בינתיים, נקודה למחשבה: לא תמיד קל `לקודד` את המידע בצורה כזאת. למשל, איך אפשר לקודד את שם השכונה כמספר? מהן הדרכים הטובות לעשות זאת?

1.1.2 דוגמא

למעשה נתקלתם בעבר במודלים: למשל, בקורס במעבדה א'. נגדיר את הניסוי הבא: החוקר מפיל כדור ברזל מגבהים שונים ורוצה לחזות כמה זמן יקח לכדור להגיע לקרקע. המידע בניסוי נתון ע`י הוקטורים בני האיבר האחד הבאים:

$$\begin{aligned}
experiment_1 &\mapsto (3)\\
experiment_2 &\mapsto (1)\\
experiment_3 &\mapsto (5)\\
experiment_4 &\mapsto (2)\\
experiment_5 &\mapsto (2)
\end{aligned}$$

כאשר התיוג נתון ע`י

$$result\_1 \mapsto 1.5$$
$$result\_2 \mapsto 1.0$$
$$result\_3 \mapsto 1.8$$
$$result\_4 \mapsto 1.2$$
$$result\_5 \mapsto 1.3$$

נרצה למצוא פונקציה f שבהינתן הגובה (ביחידות של מטרים) תחזה כמה זמן יקח לכדור ליפול (ביחידות של שניות). אתם יודעים כי הפונקציה `הנכונה` היא

$$f(h) = \sqrt{\frac{2}{g}h}$$

נרצה למצוא את המודל על סמך המידע בלבד.

1.1.3 דוגמא

נניח שאנו רוצים לחזות, בהינתן תמונה של כלב, מהו סוג הכלב שבתמונה, בהינתן חמישה סוגים אפשריים. מה יהיו הפיצ'רים במקרה זה? נניח כי נוכל לקחת את הפיקסלים בתמונה כפיצ'רים (מה היתרונות והחסרונות בכך?). אולם כיצד נבטא את ה־target? במקרים כאלו נהוג לבנות מודל שיחזיר התסברות לכל class, כלומר $Y \subset \mathbb{R}^5$ הוא אוסף וקטורים מאורך 5, כאשר סכום האיברים בהם הוא 1, שכן הם מבטאים הסתברות להיות מ־class מסוים.

1.1.4 אימון מודלים

הגדרנו מהו מודל, אולם כיצד מעריכים כמה טוב הוא מודל? נתייחס אם כן למאפיין השני. כדי להעריך מודל יש צורך בפונקציית Loss. זו פונקצייה שתעריך כמה גדולה היא השגיאה של המודל. נסמן לרוב

$$L : Y \times Y \to \mathbb{R}$$

פונקציית ה־Loss תקבל את הערך החזוי ע`י המודל שלנו $(f(x))$ ואת התיוג המקורי (y), ותחזיר את מידת החומרה שאנו מייחסים להבדל ביניהם. פעמים רבות נעדיף לסמן

$$L : Y^m \times Y^m \to \mathbb{R}$$

ולקבל פונקציה שמקבלת רשימה של תחזיות ורשימה של תיוגים, ומחזירה הערכה. הסיבה לכך פשוטה: כאשר אנו מעריכים מודל, אנו רוצים להעריך את הביצועים שלו על כמות `מייצגת` של המידע, ולא בנקודה אחת.

נוכל לגזור כמה חוקים שאנו רוצים שכל פונקציית Loss טובה תקיים:

1. אם $f(x) = y$ נצפה שיתקיים $L(f(x), y) = 0$.

2. ככל שהשגיאה גסה יותר נצפה לערך L גבוה יותר.
הנה כמה דוגמאות לפונקציות Loss מוכרות:

$$mean\,absolute\,error:\ L(f(x_i),y_i)=\frac{1}{m}\sum_i|f(x_i)-y_i|$$

$$mean\,squared\,error:\ L(f(x_i),y_i)=\frac{1}{m}\sqrt{\sum_i(f(x_i)-y_i)^2}$$

$$binary\,crossentropy:\ L(f(x_i),y_i)=-\frac{1}{m}\sum y_i\cdot log(f(x_i))+(1-y_i)log(1-f(x_i))$$

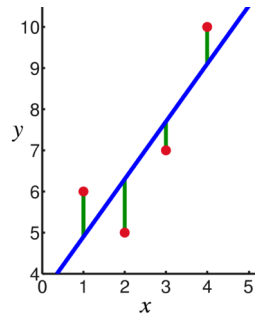
שתי הפונקציות הראשונות הן פונקציות loss נפוצות בבעיות רגרסיה. השלישית נפוצה בבעיות סיווג, קלאסיפיקציה, ומגיעה מעולם ההסתברות, ובפרט היא דרך להערכת מרחק בין התפלגויות.
בהינתן כמות מידע עליה אנו מתאמנים, חשוב שמראש נבצע הפרדה לסט האימון, training set, ולסט המבחן, test set. בסט המבחן לא ניגע, למעט כאשר נרצה לבדוק את המודלים שלנו. לא נתאמן עליו, כדי לא "להזליג מידע" מהמבחן לאימון - מתוך ההבנה הפשוטה שבמקרים מציאותיים לא נקבל את התיוג של המידע עליו נרצה לבצע תחזיות, אלא נאלץ לבצע תחזיות בכוחות עצמנו.
כעת אנו יודעים מהו מודל טוב (נזכיר, מודל כפונקציה שמתאימה למדידות הניסוי את הערך החזוי): עלינו למצוא f שתתן Loss מינימלי.
מילה על פילוסופיה של מודלים: פעמים רבות נתייחס לפונקציה `האמיתית`, כלומר לפונקציה שאותה היינו רוצים לקרב באמצעות מודל. נוח לחשוב על העולם כאילו הוא נשלט אמצעות פונקציות, שעם כל מורכבותן, אפשר לקרב במודלים, ואכן פעמים רבות נעשה זאת. אולם פעמים רבות אין זה כך: בין אם יש אלמנטים `הסתברותיים` במציאות, ובין אם יש לנו `משתנים נעלמים` שאיננו מכירים ושמשפיעים על המציאות.
נראה כעת דוגמאות לכמה מודלים פשוטים ושימושיים:

1.2 רגרסיה ליניארית

כעת משהגדרנו את עולם הבעיה, נכיר את המודל הראשון שלנו: הרגרסיה הליניארית. בהינתן המידע שלנו נרצה להתאים מודל ליניארי שיתאר את המידע בצורה הכי טובה, כלומר שימזער את פונקציית ה־Loss. ברגרסיה ליניארית נמזער את פונקציית mean square:

$$L(f(x_i),y_i)=\frac{1}{m}\sum_i(f(x_i)-y_i)^2$$

קל לראות זאת באופן גרפי:



איור 1:

רגרסיה ליניארית

ה־Loss הוא סכום הריבועים של המרחקים מהישר הכחול. מהו המודל הליניארי הטוב ביותר שנוכל להתאים? במקרה זה ניתן למצוא נקודת מינימום לפונקציית ה־Loss בצורה אנליטית. נדגים זאת במקרה החד־מימדי, כלומר כאשר המידע מיוצג ע`י וקטורים חד־מימדיים.

במקרה שכזה המודל הוא פונקציה ליניארית (אפינית)

$$f : X(=\mathbb{R}) \to Y(=\mathbb{R})$$

שמשום כך נתונה ע`י

$$f(x) = a \cdot x + b$$

כלומר, המודל מתואר לחלוטין ע`י a, b ונרצה למצוא a, b אופטימליים כדי למזער את הביטוי

$$L(f(x_i), y_i) = \frac{1}{m}\sum_i (f(x_i) - y_i)^2 = \frac{1}{m}\sum_i (a \cdot x_i + b - y_i)^2$$

נגזור לפי a, b לקבלת

$$0 = \nabla L = \begin{pmatrix} \frac{\partial L}{\partial a} \\ \frac{\partial L}{\partial b} \end{pmatrix} = \frac{2}{m}\begin{pmatrix} \sum_i x_i \cdot (a \cdot x_i + b - y_i) \\ \sum_i a \cdot x_i + b - y_i \end{pmatrix}$$

נסמן

$$\begin{aligned} \tilde{x} &= \textstyle\sum_i x_i \\ \bar{x} &= \textstyle\sum_i x_i^2 \\ \tilde{y} &= \textstyle\sum_i y_i \\ \bar{y} &= \textstyle\sum_i x_i y_i \end{aligned}$$

ונקבל כי

$$0 = \begin{pmatrix} \bar{x} \cdot a + \tilde{x} \cdot b - \bar{y} \\ \tilde{x} \cdot a + m \cdot b - \tilde{y} \end{pmatrix}$$

קיבלנו שתי משוואות ליניאריות בשני נעלמים, a, b. שימו לב: אנו יודעים את כל ערכי x_i ו־y_i - אלו נקודות המידע שלנו. לכן מתקיים

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \bar{x} & \tilde{x} \\ \tilde{x} & m \end{pmatrix}^{-1} \cdot \begin{pmatrix} \bar{y} \\ \tilde{y} \end{pmatrix} = \frac{1}{m\bar{x} - \tilde{x}^2} \begin{pmatrix} m \cdot \bar{y} - \tilde{x}\tilde{y} \\ -\tilde{x}\bar{y} + \bar{x}\tilde{y} \end{pmatrix}$$

כלומר, מצאנו פתרון אנליטי לישר שממזער את השגיאה כפי שהגדרנו אותה. השתכנעו שנקודת הקיצון שמצאנו היא אכן נקודת מינימום (קל לראות שהפונקציה יכולה להתבדר לאין־סוף, למשל עבור a קבוע וערכי b הולכים וגדלים). לכן זהו המודל הליניארי הטוב ביותר שניתן לקבל.

האם אנחנו מרוצים? ובכן, לא. כמו שאתם יכולים לנחש, התורה לא נעצרת ברגרסיה ליניארית, ובעתיד נדרש להתאים מודלים מורכבים יותר. שימוש בכלים אנליטיים לצורך המשימה עשוי להיות די מוגבל. נצטרך לפתח כלים טובים יותר לפתרון בעיות מינימום.

הערה פילוסופית: שימו לב, למעשה כל הבעיה הסתכמה, בסופו של דבר, בפתרון בעיית מינימום. אכן, באופן בסיסי למדי, כל בעיית machine learning היא כזו: אנו מתאימים מודל שמצמצם פונקציית שגיאה מסוימת. המודל הוא לרוב פונקציה ממשפחה מסוימת של פונקציות. למשל, כאן התאמנו פונקציה ממשפחת הפונקציות הליניאריות. לרוב נבחר `משפחה` של פונקציות, שניתן לתאר את כל חבריה באמצעות פרמטרים (למשל בדוגמא לעיל, a ו־b) ונפתור בעיית מינימום על הפרמטרים הללו. זו כל התורה על רגל אחת. אבל שימו לב שלהתאים מודלים ולעשות machine learning נשמע הרבה יותר מרשים מ`לחפש נקודות מינימום`. עכשיו כשאתם חשופים לסוד הזה, אתם צריכים לשמור עליו היטב, כדי להמשיך ולשמר את המסתורין.

1.2.1 שימוש ב־Gradient Decsent

נכיר את הכלי הבסיסי והשימושי ביותר בארגז הכלים של מדען המידע למציאת מינימום לפונקציה: Gradient Decsent. בהינתן פונקציה g (חלקה) נרצה למצוא לה נקודת מינימום. נניח כי $g = g(u)$. האלגוריתם איטרטיבי. נתחיל מניחוש של נקודת מינימום, u_0. בכל צעד של האיטרציה אנו מעדכנים את הנקודה כך:

$$u_0 \leftarrow u_0 - \varepsilon \cdot \frac{\partial g}{\partial u}(u_0)$$

כאשר ε הוא קצב הלמידה (learning rate). אנו יודעים כי בסביבה קטנה של u_0, אם נלך בכיוון הנגזרת נקבל ערך גבוה יותר בפונקציה. בהתאם, כאשר אנו הולכים כנגד כיוון הנגזרת אנו מקבלים ערך נמוך יותר.

השימוש ב־gradient decsent מוצדק היטב במקרה של פונקציות קמורות - פונקציות בהן אין נקודות מינימום מקומיות אליהן אנו עשויים להתכנס. הוא שימושי מאוד גם במציאת מינימום מסוגים שונים בפונקציות לא קמורות.

1.2.2 דוגמא

נשתמש ב־gradient decsent כדי למצוא את נקודת המינימום במקרה של רגרסיה ליניארית. שימו לב: הפונקציה שאנו מנסים למזער במקרה זה היא אומנם L, אך אנו איננו חושבים עליה יותר כעל פונקציה $L(f(x_i), y_i)$, אלא כעל פונקציה של הפרמטרים אותם אנו מנסים לקבוע, $L(a, b)$. נזכר כי

$$L = \frac{1}{m}\sum_i (a \cdot x_i + b - y_i)^2$$

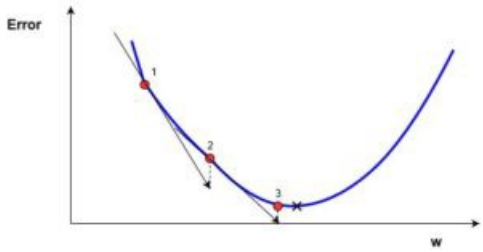
וכי

$$\begin{pmatrix} \frac{\partial L}{\partial a} \\ \frac{\partial L}{\partial b} \end{pmatrix} = \frac{2}{m}\begin{pmatrix} \sum_i x_i \cdot (a \cdot x_i + b - y_i) \\ \sum_i a \cdot x_i + b - y_i \end{pmatrix}$$

כעת כלל העדכון הוא

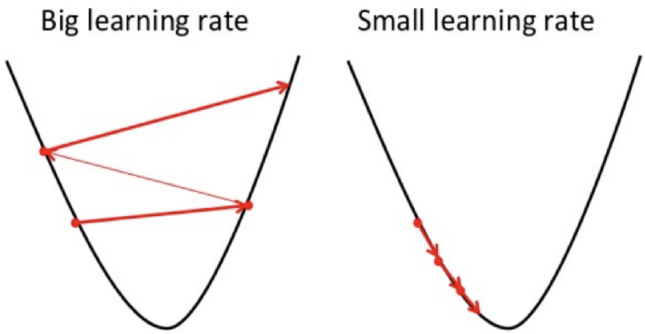
$$\begin{aligned} a &\leftarrow\!\shortmid a - \varepsilon \cdot \tfrac{2}{m} \cdot \left((\textstyle\sum_i x_i^2) \cdot a + (\sum_i x_i) \cdot b - (\sum_i x_i y_i)\right) \\ b &\leftarrow\!\shortmid b - \varepsilon \cdot \tfrac{2}{m} \cdot ((\textstyle\sum_i x_i) \cdot a + m \cdot b - (\sum_i y_i)) \end{aligned}$$

נשים לב כי ככל ש־ε קטן נקבל התכנסות טובה יותר לנקודת המינימום, אך זו תיקח יותר זמן.



איור 2:

האם ε יכול להיות גבוה מדי? בהחלט.



איור 3:

כפי שניתן לראות באיור, ε גבוה מדי יכול להוביל להתבדרות.

נסכם: רגרסיה ליניארית היא מודל פשוט ביותר, אך שימושי. ראינו כי ניתן להתאים אותו באמצעות פתרון אנליטי, אך ראינו כי ניתן לעשות זאת גם בשיטות נומריות כלליות יותר, שלא מניחות הנחות חזקות במיוחד על הפונקציה אותה אנו מנסים להתאים. לרגרסיה הליניארית ישנו חיסרון בולט: ובכן, היא ליניארית. הנחת הליניאריות אמנם מפשטת בעיות רבות, אך פעמים רבות היא פשוט אינה מתאימה.

1.3 רגרסיה לוגיסיטית

רגרסיה לוגיסטית דומה במובנים רבים לרגרסיה הליניארית, אך היא באה לטפל בתחום מסוים של בעיות: בפרט, בבעיות חיזוי של תוצאה בינארית. בעיות כאלה נמדל לרוב כאשר התיוג מקבל ערכים של 1 או 0. ברור לכן מדוע רגרסיה ליניארית לא תיטיב לתאר בעיות כאלה: הטווח שלה כלל אינו חסום, והיא עשויה לקבל מגוון רחב מדי של ערכים.

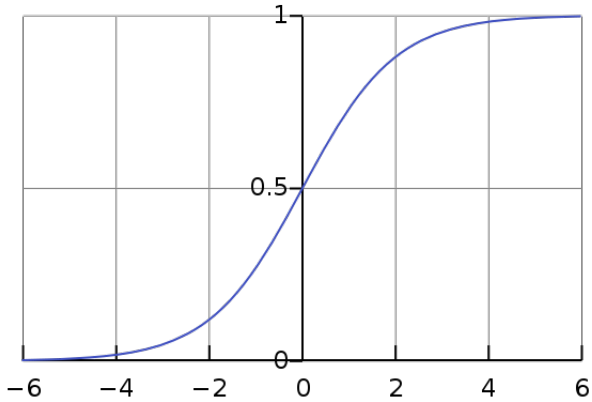
הפונקציה במודל הרגרסיה הלוגיסטית, אותה אנו מנסים להתאים, היא

$$f(x) = \frac{1}{1 + e^{-(a \cdot x + b)}}$$

מודל זה הוא הרכבה של פונקציה ליניארית עם הפונקציה הלוגיסטית, שנקראת גם סיגמואיד,

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

לפונקציה זו יש כמה תכונות שהופכות אותה לשימושית מאוד:



איור 4:

הערכים שלה חסומים בין 0 ו־1.

ל־σ יש תכונה מועילה נוספת:

$$\frac{d\sigma}{dz}(z) = \frac{e^{-z}}{(1+e^{-z})^2} = \sigma(z) \cdot \frac{e^{-z}}{1+e^{-z}} = \sigma(z) \cdot \frac{(1+e^{-z})-1}{1+e^{-z}} = \sigma(z) \cdot (1-\sigma(z))$$

לכן, מרגע שחישבנו את $\sigma(z_0)$, יהיה לנו קל מאוד לחשב את $\frac{d\sigma}{dz}(z_0)$.

יש לנו כעת את המבנה של הפונקציה f, ועלינו להתאים אותה כך שf תקרב את הפונקציה בצורה הטובה ביותר. עלינו לשאול את עצמנו כעת, באיזו פונקציית Loss נשתמש? נוכל להשתמש בפונקציית mse שראינו קודם לכן. אולם בשימוש ברגרסיה לוגיסטית נהוג להשתמש בפונקציית Loss אחרת, הנובעת מתוך פרשנות אחרת לתוצאות:

נוכל לחשוב על חיזוי המודל כעל גודל הקובע את הסיכוי לקבל תוצאה חיובית (כלומר 1). נרצה שבמקרים בהם קיבלנו 1, הסיכוי יהיה קרוב כמה שיותר ל־1, ולהיפך: במקרים בהם קיבלנו 0 נרצה שהמודל יחזה סיכוי קרוב ככל האפשר ל־0. אם נחשוב לכן גם על התיוג המקורי בתוך סיכוי לתוצאה חיובית, נוכל להשתמש במדד cross entropy לחישוב מרחק בין ההתפלגויות, באמצעות פונקציית ה־Loss הבאה:

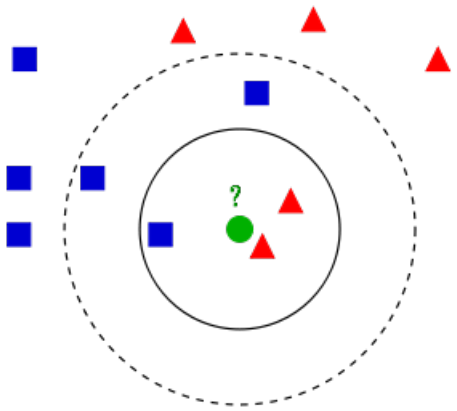
$$L(f(x_i), y_i) = -\frac{1}{m}\sum y_i \cdot log(f(x_i)) + (1-y_i)log(1-f(x_i))$$

כדי לכוונן את המודל נוכל להשתמש ב־gradient decsent כפי שראינו קודם.

1.4 K Nearest Neighbors

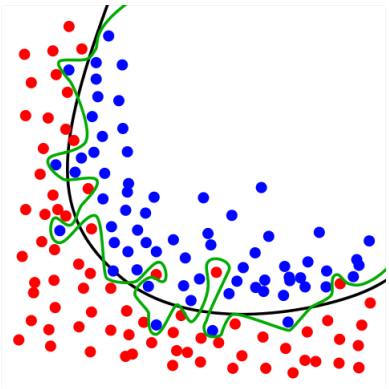
ראינו שני מודלים פשוטים המתאימים לחיזוי ערך רציף או לחיזוי בדיד (או קטגוריאלי). נראה כעת מודל פשוט יותר במובנים רבים, המתאים לשתי המשימות: בהינתן המידע עליו אנו מתאמנים, X, לכל מידע חדש נוכל לשאול, מיהן k הנקודות בסט האימון שקרובות אליו

ביותר? נשתמש בתיוגים שלהן כדי לחזות ערך לנקודה החדשה. במקרה של חיזוי ערך רציף נשתמש בממוצע הערכים שלהן, ובמקרה של חיזוי ערך בדיד נשתמש בערך הנפוץ ביותר.
נשים לב כי המודל לא מצריך כיוונון של פרמטרים. בפרט, זהו איננו מודל פרמטרי. זהו מודל בסיסי מאוד, אך יש לו ערך, כפי שנראה בהמשך הקורס, הנובע, בין השאר, מהיותו שונה מהותית מהמודלים הפשוטים האחרים שראינו ושנראה בהמשך. יש למודל זה חסרון בולט - כדי לקבל תחזיות אנו נדרשים לשמור את כל סט האימון, ותחזיות יעשו ב־$O(|data\ set|)$, כאשר סט האימון עשוי להיות גדול מאוד.



איור 5:

נשים לב לעובדה מעניינת נוספת: KNN הוא המודל הראשון שראינו בו, בהיעדר הפרדה בין train ו־test, ישנו פוטנציאל להגיע להתאמת יתר מושלמת, over fit: אכן, אנו יכולים לבנות מודל שיחזה במדויק על סט האימון, באמצעות בחירת $k = 1$. אולם זהו מודל גרוע למדי, שכן הוא צפוי שלא להצליח להכליל חיזויים לשאר המידע. באופן כללי,over fit הוא מצב בו התחזית שאנו נותנים משמעותית טובה יותר על סט האימון מאשר על סט המבחן (test set, validation set). במצב שכזה המודלים שלנו לא "מכלילים" ידע שהם למדו באימון לשימוש בסט המבחן, אלא "משננים" ידע מסט האימון, שאין כל ערובה שיסייע להם בסט המבחן. בתמונה ניתן לראות דוגמא ל־over fit: הקו השחור מתאר הפרדה טובה בין שני class־ים, בעוד הקו הירוק מתאר הפרדה בהתאמת יתר לסט האימון. אם נדגום נקודות חדשות לסט המבחן נקבל כישלון בחיזוי בסביבות הקו הירוק.



איור 6:

1.5 הערה לגבי מימד הפלט

בתיאור שלנו את הרגרסיה הליניארית והרגרסיה הלוגיסטית הנחנו כי החיזוי הוא ערכים $y \in \mathbb{R}$. אולם מובן כי נוכל להגדיר מודלים דומים עבור $y \in \mathbb{R}^n$ ל־n כלשהו: זאת כיוון שאנו יודעים להגדיר פונקציות loss ב־n מימדים. למשל, mse עבור וקטורים מחושב ע"י

$$mse(\hat{y}, y) = |\hat{y} - y|^2 = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

למעשה, במודלים אלו, כמה פלטים פירוש הדבר כמה מודלים שונים: החיזוי לכל פלט בנפרד אינו תלוי בחיזוי לפלטים האחרים. זאת כיוון שכל פלט נחזה באמצעות המשקולות המתאימות לו. נעיר כי זהו אינו בהכרח המצב במודלים אחרים, בהם חיזוי כמה פלטים שונה מבנייה של כמה מודלים שונים.

לחיזוי של כמה פלטים דווקא ישנה חשיבות רבה בבעיות מעשיות: כפי שראינו בתחילת השיעור, בבעיה בה נרצה לסווג כל דוגמא לאחת מכמה קטגוריות, יתכן ונרצה לחזות לכל דוגמא וקטור, שיקבע באיזו הסתברות אנו חושבים שהיא שייכת לכל אחת מהקטגוריות. נראה בעתיד מודלים שמאפשרים חיזוי שכזה בצורה נוחה.

1.6 הערה לגבי התפלגות המידע

עד כה התייחסנו אל המידע כאל וקטורים n־מימדיים, ואל מודל כאל פונקציה הפועלת עליהם. את הערכת המודל מבצעים באמצעות פונקציית Loss. נדגיש כי הערכת המודל צריכה להיעשות, באיזשהו אופן, על כמות המייצגת את **כל** המידע, ולא רק על דוגמאות ספציפיות. בפרט, היינו רוצים להתאים כמה שיותר טוב להתפלגות המקורית של המידע. אמנם טענו שהמידע מיוצג כוקטורים n־מימדיים, אך האם כל הוקטורים הללו נפוצים באותה המידה, ומתקבלים ב־dataset שלנו באותה הסתברות? מובן שלא. למשל, אם אנחנו מתארים אוכלוסיה באמצעות הגובה שלה, אנו מקבלים אוסף וקטורים חד־מימדיים

שמתפלגים נורמלית סביב ממוצע כלשהו.

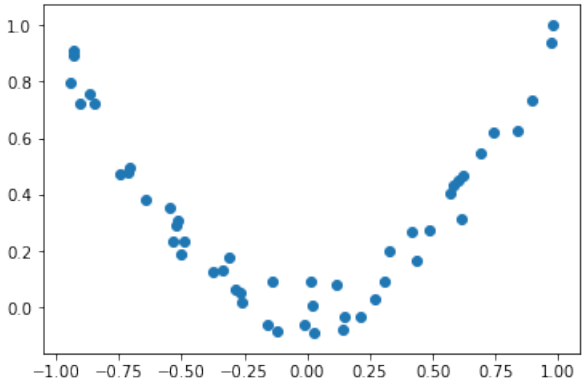
כעת נניח כי אני רוצה לבנות מודל שיחזה, בהינתן גובה של אדם, את המשקל שלו. כיוון שרוב האנשים הם בגובה $1.6 - 1.8$ מטרים, חשוב לי להצליח במיוחד בתחום זה של האוכלוסיה, כדי לקבל מודל שיצליח בכמה שיותר מן המקרים. אולם אם הדבר חשוב לי, מוטב שהוא יבוא לידי ביטוי בפונקציית Loss. האם הדבר קורה?

כן. לא ציינו זאת מפורשות, אבל הגדרנו את ה־Loss כפועלת על אוסף נקודות במדגם. כיצד נדגמות הנקודות באמצעותן אנו מעריכים את המודל? לפי ההתפלגות המקורית של המידע.

ההתפלגות של המידע חשובה, ורלוונטית תמיד לאופן בו אנו מעריכים את המודל שלנו.

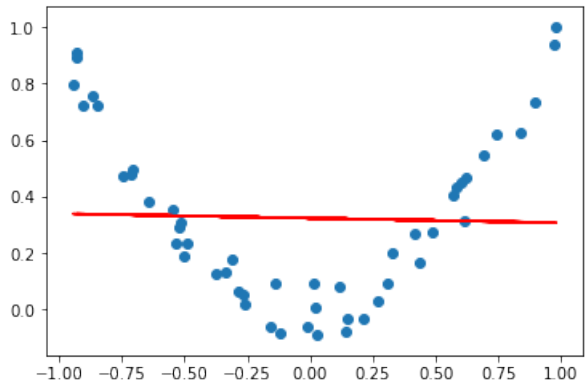
1.7 שימוש ברגרסיה ליניארית ליצירת מודלים לא ליניאריים ורגולריזציה

הכרנו את הרגרסיה הליניארית בתור מודל ליניארי פשוט וקל לאימון, בעל חסרון משמעותי: היותו ליניארי. נניח שנתון לנו המידע הבא:



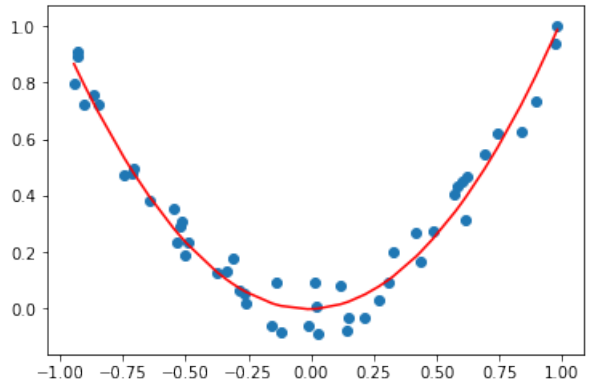
איור 7:

התאמה ליניארית פשוטה אליו לא תתאר אותו היטב:



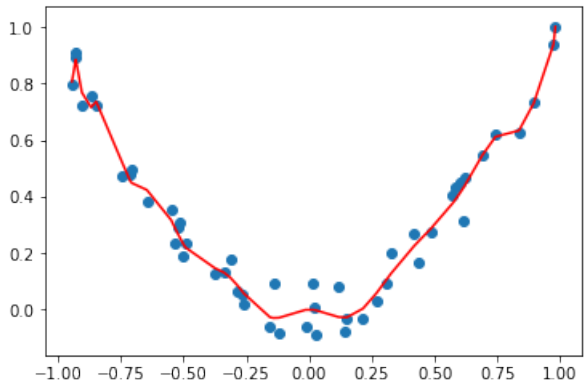
איור 8:

אכן, ברור כי פרבולה תיטיב לתאר את המידע. כיצד נוכל לתאר פרבולה ברגרסיה ליניארית? נוכל להוסיף עוד פיצ'רים למידע: במקום להשתמש בוקטורים מאורך 1, נוכל להשתמש בוקטורים מאורך 2 עם התכונות (x, x^2). נוכל לקבל מודל ליניארי ב־x^2, שמשום כך יהיה מודל לא ליניארי ב־x.



איור 9:

אולם אנו עשויים להיתקל בבעיה: אנו עשויים אמנם לרצות להתאים פולינומים, אך כעת עלינו לוודא שאיננו מתאימים פולינום ממעלה גבוהה מדי למידע שלנו. כך נראית התאמה של פולינום ממעלה 20 למידע:



איור 10:

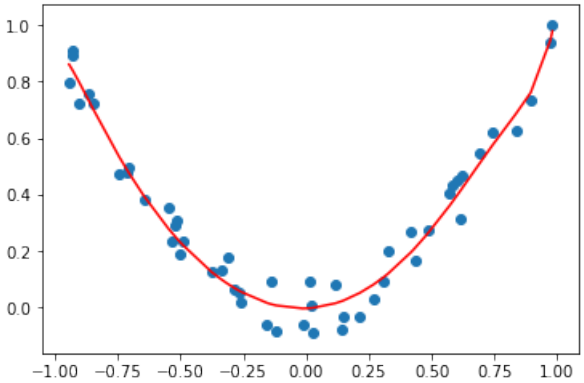
זוהי התאמת יתר על סט האימון: אכן, אם נוסיף נקודות חדשות נקבל כי המודל לא מתאר אותן בצורה טובה. כדי להתמודד עם הבעיה ניתן להשתמש ברגולריזציה: המטרה שלנו היא להתאים מודל מורכב, אך לרסן את המורכבות שלו באופן כלשהו. נוכל לעשות זאת באמצעות אילוץ כלשהו על המשקולות, הפרמטרים, של המודל. למשל, נוכל לדרוש שהמשקולות לא תהיינה `גדולות מדי`. איך עושים זאת? באמצעות הוספת ביטוי לפונקציית ה־Loss. נכתוב את פונקציית ה־Loss כך:

$$L(\theta) = \sum_i (\omega \cdot x_i + b - y_i)^2$$

כאשר θ הוא שם כולל למשקולות במודל. רגולריזציה מסוג L_2 נראית כך:

$$L(\theta) = \sum_i (\omega \cdot x_i + b - y_i)^2 + \lambda \cdot \sum_j |\theta_j|^2$$

כלומר, אנו מוסיפים עוד ביטוי לפונקציית ה־Loss, שערכו קטן ככל שסכום הריבועים של המשקולות קטן. בכך אנו מעודדים את המשקולות להיות קטנות יותר. λ הוא היפר פרמטר של המודל: פרמטר שאנו בוחרים מראש, ושאינו נלמד במסגרת האימון. במקרה זה, λ הוא הפרמטר השולט בעוצמת הרגולריזציה: ככל ש־λ גדול יותר, יש ביטוי גדול יותר לרכיב הרגולריזציה בפונקציית ה־Loss. נתאים שוב פולינום ממעלה גבוהה, הפעם עם רגולריזציה מתאימה:



איור 11:

קיבלנו מודל טוב יותר, שכן הוא אינו סובל מהתאמת יתר. באיזה מובן הוא טוב יותר? הרי ברור שהמודל הקודם שהתאמנו מתאים לסט האימון טוב יותר. הוא אמנם מתאים לסט האימון טוב יותר, אך לא נצליח להכליל אותו בצורה טובה למידע חדש. כלומר, אם נרצה לקבל ממנו תחזיות לנקודות אחרות נקבל שגיאה גדולה.

לסיכום, רגולריזציה מאפשרת לנו לרסן תופעות של התאמת יתר של מודלים באופן יעיל. הדבר נעשה באמצעות משחק עם פונקציית ה־Loss.