אוניברסיטת בן-גוריון בנגב

הפקולטה למדעי הטבע

המחלקה למדעי המחשב

דו"ח מסכם

נושאים בחזית מדעי המחשב
לסטודנטים מצטיינים

מגישים:

יונתן ששוני 205916265

יוסי כרמלי 204752406

1. מבוא

פרופ' משה זיפר העביר במהלך הקורס הרצאה בנושאים אלגוריתמים אבולוציונים ולמידת מכונה.

אנחנו בחרנו להתמקד בתחום הלמידה המונחית בנושא למידת מכונה.

למידת מונחית היא למידה שלומדת מניסיון, כלומר בהינתן dataset מתוייג (שניתן לחשוב עליו כעל אוסף של שאלות ותשובות מהן ניתן ללמוד) יוצרים מסווג שביכולתו לבצע פרדיקציה/הכללה לשאלות חדשות.

לאחר השלמה של סדרת הרצאות שהתבקשנו לצפות בהן, בחרנו לחקור יותר לעומק מספר אלגוריתמי למידה שונים, ולממש חלק מהם.

לאחר מכן בעזרת dataset בשם MNIST המכיל תמונות של כל הספרות 0-9, ערכנו בדיקות לטיב המסווגים שקיבלנו מהאלגוריתמים השונים.

בנוסף, בתחילת הקורס הוסבר כי ניתן להרחיב את היקף הפרויקט על מנת לקבל תוספת לציון.

לכן בחרנו להרחיב את היקף הפרויקט וללמוד גם נושא מתחום הלמידה הלא מונחית, $clustering$.

כל אחד מהחלקים הבאים יציג אלגוריתם למידה מסויים ואת תוצאות הבדיקות שערכנו על המסווג שהתקבל.

.2 *k-Nearest Neighbors*

אלגוריתם השכנים הקרובים הוא אלגוריתם מאוד אינטואיטיבי, בהינתן מדגם S של דוגמאות ותויות ודוגמא חדשה x הוא יבחר את k השכנים הדומים ביותר ל x מתוך S, ויחזיר את תווית הרוב מבין השכנים הקרובים.
באופן פורמלי: נקבל כקלט לאלגוריתם מדגם S ושלם $k \geq 1$, ונחזיר כפלט את המסווג h_S^{knn}.
המסווג h_S^{knn} חוזה עבור כל דוגמה x במרחב הדוגמאות את התווית לפי הכלל הבא:

$$h_S^{knn}(x) \coloneqq The\ majority\ label\ among\ \{y_{\pi_1(x)}, \dots, y_{\pi_k(x)}\}$$

כאשר $\pi_i(x)$ הוא האינדקס במדגם של השכן ה i הקרוב ביותר לדוגמא x, ו- $y_{\pi_i(x)}$ היא תוויות של שכן זה.

כאשר נרצה לעשות שימוש באלגוריתם השכנים הקרובים עולות שתי שאלות עיקריות:

- כיצד מוגדר המרחק בין שתי דוגמאות במרחב הדוגמאות?
  אנחנו בחרנו לעבוד עם dataset שמרחב הדוגמאות שלו כאמור הוא תמונות של ספרות, כלומר $x \in \mathbb{R}^{k \times k}$.
  בהינתן מטריצה המייצגת תמונה נשטח אותה ונציג אותה כוקטור $x \in \mathbb{R}^{k^2 \times 1}$. כעת בהינתן שני וקטורים נגדיר את המרחק בניהם באמצעות הנורמה האוקלידית: $\rho(x, x') = \|x - x'\|_2 = \sqrt{\sum_{i=1}^{d} \left(x(i) - x'(i)\right)^2}$.

- איזה ערך נבחר עבור הפרמטר k?
  לשם כך פנינו לאלגוריתם **cross validation** שיעזור לנו להעריך את הערך הטוב ביותר עבור k מתוך קבוצת ערכים אפשריים. האלגוריתם מחלק את המדגם לשני חלקים, בראשון הוא משתמש כמדגם כדי לבצע למידה. בשני הוא משתמש כדי להעריך את השגיאה של המסווג שנלמד, מדגם כזה נקרא מדגם ולידציה.
  <u>הקלט לאלגוריתם:</u>
  - מדגם מתוייג S
  - אלגוריתם למידת $\mathcal{A}$ - במקרה שלנו knn
  - Ψ קבוצת ערכים סופית עבור הפרמטר אותו נרצה לשערך – במקרה שלנו קבוצת ערכים עבור k
  - p – במהלך האלגוריתם נחלק את המדגם ל p חלקים שווים.

  <u>הפלט:</u> מסווג של האלגוריתם הנתון לאחר ביצוע למידה עם הערך הפרמטר הטוב ביותר.

  <u>שלבי האלגוריתם:</u>
  1. חלק את המדגם S ל p חלקים שווים בגודלם $S_1, \dots, S_p$.
  2. לכל $\alpha \in \Psi$ בצע:
     - 2.1 לכל $i \in \{1, \dots, p\}$
       - 2.1.1 $V = S_i$ (מדגם הולידציה)
       - 2.1.2 $S' = S \setminus S_i$
       - 2.1.3 $h_i = \mathcal{A}(S', \alpha)$ (במקרה שלנו $\mathcal{A}(S', \alpha) = h_S^{\alpha-nn}$)
       - 2.1.4 $\epsilon_i = err(h_i, V)$ (השגיאה היא אחוז הטעויות שביצע h_i על הדוגמאות מ V)
     - 2.2 $\epsilon_\alpha = \frac{1}{p} \cdot \sum_{i=1}^{p} \epsilon_i$
  3. $\alpha^* = \underset{\alpha \in \Psi}{\operatorname{argmin}}\, \epsilon_\alpha$
  4. $h^* = \mathcal{A}(S, \alpha^*)$
  5. החזר את h^*.

<u>תוצאות בדיקת המסווג:</u>
עבור מדגם בגודל 2000 המכיל תמונות עם הספרות 0-9 ועבור קבוצת הערכים אופציונלים $\{1,2,3,4,5,6,7,8,9,10\}$ עבור הפרמטר k, וחלוקה של המדגם ל-5 חלקים.
קיבלנו שהערך האופטימלי שהחזיר אלגוריתם cross validation הינו $k = 5$.
השגיאה שהתקבלה על מדגם מבחן בגודל 500 הינה: 9%.

3. עצי החלטה – *ID3*

עץ החלטה הוא מודל שבא לתאר תהליך החלטה טבעי. קל להסביר כיצד המודל עובר גם לאנשים מחוץ לתחום מדעי המחשב.
המסווג מתואר על ידי עץ החלטה המבצע חיזוי על ידי טיול משורש לעלה בעץ. כאשר בכל צומת פנימי בעץ מופיע מבחן עבור אחת התכונות של הדוגמא הנתונה, ההתקדמות לצד ימין או שמאל נעשית בהתאם לתוצאת המבחן בצומת הנוכחי. בעלי העץ נמצאות תוויות אפשריות ממרחב התוויות, וכאשר נגיע לסוף הטיול בעץ ונגיע לעלה נחזיר את התווית שהוא מייצג.
למען פשטות נניח כי מרחב הדוגמאות הוא $\{0,1\}^d$ ומרחב התוויות הוא $\{0,1\}$.
ID3 הוא אלגוריתם חמדן, אשר בכל שלב בוחר את התכונה שתפריד את המדגם בצורה הטובה ביותר, בעזרת פונקציית $Gain$.

<u>קלט האלגוריתם:</u> מדגם S ותת קבוצה של פיצ'רים $A \subseteq \{1, \dots, d\}$.
<u>פלט האלגוריתם:</u> עץ עבור S עם פיצ'רים מתוך A.
<u>שלבי האלגוריתם:</u>

- אם כל הדוגמאות ב S מתוייגות באותה תווית y, החזר עלה עם תווית זו.
- אם $A = \emptyset$ החזר עלה מתוייג עם תווית הרוב ב S.
- $j = \underset{i \in A}{\operatorname{argmax}}\, Gain(S, i)$
- $S_0 = \{(x, y) \in S \mid x(j) = 0\}$ (כאשר $x(j)$ היא הקורדינטה ה j בוקטור x)
- $S_1 = \{(x, y) \in S \mid x(j) = 1\}$
- החזר עץ בעל שורש עם תכונה j, בן שמאלי המתקבל על ידי $ID3(S_0, A\backslash\{j\})$, ובן ימני המתקבל על ידי $ID3(S_1, A\backslash\{j\})$.

<u>מהי הפונקציה $Gain$?</u>
היא פונקציה בה אנו עושים שימוש כדי לבחור את הפיצ'ר עליו משתלם לנו לשאול בשלב הנוכחי, ובמילים אחרות את הפיצ'ר שמפריד את המדגם שלנו בצורה הטובה ביותר. נסמן ב- q את השבר המתאר את אחוז הדוגמאות במדגם עם תווית $y = 1$, ואת שאר הדוגמאות עם תווית $y = 0$ ב $1 - q$. אם בשלב הנוכחי נעצור וניצור עלה נקבל שהשגיאה במסלול זה הינה $err(q) = \min(q, 1 - q)$. לפיכך נגדיר $\mathrm{err}_{\mathrm{before(S)}} = \mathrm{err(q)}$.
נניח כי נרצה לפצל את המסלול עם תכונה i כלומר נשאל $x(i) = 1$? ונסמן:

- q_i^0 שבר של אחוז הדוגמאות המתויגות $y = 1$ בצד השמאלי, כלומר עבור דוגמאות המקיימות $x(i) = 0$.
- q_i^1 שבר של אחוז הדוגמאות המתויגות $y = 1$ בצד הימני, כלומר עבור דוגמאות המקיימות $x(i) = 1$.
- p_i^0 אחוז הדוגמאות שהלכו שמאלה, כלומר דוגמאות המקיימות $x(i) = 0$.

כעת נשים לב כי השגיאה לאחר המבחן $x(i) = 1$? הינה: $err_{after}(S, i) = p_i^0 \cdot err(q_i^0) + (1 - p_i^0) \cdot err(q_i^1)$.
לכן נגדיר $Gain(S, i) = err_{before}(S) - err_{after}(S, i)$, כלומר הפונקציה בוחרת את המבחן אשר יקטין את השגיאה בערך הגדול ביותר.

לא ניתן לבצע למידה בצורה טובה כאשר נשקול להחזיר עץ החלטה מתוך קבוצה כל עצי ההחלטה האפשריים.
ניסיון כזה יגרום לתופעת $overfitting$, כלומר השגיאה על המדגם תהיה קטנה, אך השגיאה על דוגמאות חדשות תהיה גבוהה.

שתי דרכים לפתור את הבעיה הן:

- <u>הגבלת מספר הקודקודים בעץ</u> – ניתן לממש את הגבלה זאת על ידי סיום האלגוריתם כאשר הגענו למספר קודקודים מסויים, או לחלופין לגזום את העץ לאחר בנייתו. הרעיון בגזימה הוא הורדה של תתי עצים שלא פוגעים באופן משמעותי בשגיאה. נניח $f(T)$ פונקציה המקרבת את השגיאה של עץ T. בהינתן עץ T לכל קודקוד פנימי j בעץ, החל מהשיכבה התחתונה ביותר, בדוק את 5 הערכים הבאים ובחר באופציה עם הערך המינימלי עבור f: השאר את המצב הקיים, החלף את קודקוד j ב-עלה עם ערך 0/1, החלף את קודקוד j עם תת עץ שמאלי/ימני.

- <u>*Random Forest*:</u> יער הוא אוסף של כמה עצי החלטה, שמחזיר פרדיקציה לפי החלטת הרוב. כעת נשאלת השאלה כיצד נייצר מספר רב של עצים שונים? דרך אחת לעשות זאת היא לבצע למידה של כל עץ בעזרת מדגם שונה וקבוצת פיצ'רים מוגרלת.

4. *Naïve bayes*

נניח כי מרחב הדוגמאות שלנו מעל $\{-1,1\}^n$ ונחשוב על דוגמה כעל סדרת המלצות של n מומחים, שכל אחד מנחש את התווית הנכונה לדעתו מתוך $\{-1,1\}$.

ההנחה המתבצעת בשיטה זו היא **אי תלות מותנת** בין המומחים, כלומר לכל שני מומחים i, j כאשר אנחנו יודעים מהי התשובה הנכונה $y \in \{-1,1\}$, אזי עצם הידיעה של הניחוש של מומחה i לא עוזרת להבין את הניחוש של מומחה j.
באופן פורמלי: $\mathbb{P}[X = x|Y = y] = \prod_{i=1}^{n} \mathbb{P}[X(i) = x(i)|Y = y]$.

נסמן את ההסתברות שמומחה i צודק בניחוש שלו ב p_i.

תחת הנחת האי תלות מותנת המסווג האופטימלי הוא: $h^*(x) = sign(\sum_{i=1}^{n} w_i \cdot x_i)$, כאשר $w_i = \log\left(\frac{p_i}{1-p_i}\right)$.

האינטואיציה למקדמים הללו היא כדלקמן:

- עבור מומחה i שכמעט תמיד צודק $(p_i \to 1)$ המשקל w_i שהוא יקבל ישאף ל ∞ ולכן התשובה תהיה בהתאם להמלצתו x_i.
- עבור מומחה i שכמעט תמיד טועה $(p_i \to 0)$ המשקל w_i שהוא יקבל ישאף ל $-\infty$ ולכן התשובה תהיה הפוכה להמלצתו.
- עברו מומחה i שצודק ב 50% (כלומר $p_i = \frac{1}{2}$) המשקל w_i שהוא יקבל שווה ל $log\left(\frac{p_i}{1-p_i}\right) = log(1) = 0$ ולכן התשובה של המסווג לא תיקח בחשבון את המלצתו.

<u>כעת נשאלת השאלה כיצד נוכל למצוא את ערכם של p_i עבור כל אחד מן המומחים?</u>

הדרך לעשות זאת היא לקרב את ערכם באמצעות מדגם S.

נסמן ב k_i את מספר הדוגמאות מתוך S בהן צדק המומחה ה i ונחשב $p_i = \frac{k_i}{|S|}$.

<u>תוצאות בדיקת המסווג:</u>
עבור מדגם בגודל *2000* המכיל תמונות עם הספרות 0,1.
השגיאה שהתקבלה על מדגם מבחן בגודל 500 הינה: 1%.

עבור מדגם המכיל תמונות עם הספרות 3,5, השגיאה שהתקבלה הינה: 10%. לדענו השגיאה עבור תמונות של הספרות *3,5* גדולה יותר משום שספרות אלה דומות יותר אחת לשנייה בהשוואה לספרות *0,1*, ולכן מסווג קשה יותר ללמוד להבדיל בניהן.

5. SVM

דרך נוספת לבצע למידה היא באמצעות מפרידים לינארים.

כיצד מפרידים לינארים מסווגים דוגמא חדשה?
מפריד (היפר-מישור) כלשהו מחלק את המרחב לשני חלקים, אם הדוגמא החדשה ממוקמת מעל המפריד היא תקבל תוית מסויימת ואם היא מתחתיו היא תקבלת את התווית ההפוכה.

קיימים אלגוריתמים שונים המאפשרים ללמוד מפריד לינארי בצורה יעילה, כמו: תכנון לינארי, $Perceptron$ ו- $Hard-SVM$.
כל אלה מניחים כי המדגם S הינו מדגם פריד, כלומר שאכן קיים מפריד לינארי שכל הדוגמאות עם תווית 1 נמצאות בצד אחד של אותו מפריד, וכל הדוגמאות עם תוית -1 נמצאות בצד השני של המפריד.
אנחנו פנינו לאלגוריתם $Soft-SVM$ אשר מוותר על הנחה זו, ומוצא מפריד לינארי גם עבור מדגם שאינו פריד.

תחילה נסביר את משמעות המושג $margin$ של מפריד לינארי, ה $margin$ הוא המרחק של המפריד מהנקודה הקרובה ביותר אליו מתוך הנקודות במדגם. ככל שה $margin$ גדול יותר אפשר לחשוב שיש למפריד יותר מרווח לטעות בו, לכן נשאף למצוא מפריד עם $margin$ גדול ככל הניתן.

כמו כן עבור מפריד לינארי w ומדגם $S=\{(x_1,y_1),\dots,\{x_m,y_m)\}$ נגדיר פונקצית הפסד, hinge loss, אותה נשאף למזער:
$\ell^h\big(w,(x_i,y_i)\big)=\max\{0,1-y_i\langle w,x_i\rangle\}$, ועל המדגם: $\ell^h(w,S)=\frac{1}{m}\cdot\sum_{i=1}^{m}\ell^h(x,(x_i,y_i))$.

נשים לב שאפילו עבור מדגם פריד ו- α מספיק גדול נקבל $\ell^h(\alpha\cdot w,S)=0$, ול- $\alpha\cdot w$ נורמה גודלה יותר משל w, אך $margin$ קטן יותר.
על מנת לאזן את הרצון מצד אחד למזער את פונקצית ההפסד ומצד שני להגדיל את ה $margin$ נעשה שימוש ברגולטור λ: כאשר עבור λ גדול נעודד נורמה קטנה כלומר $margin$ גדול, ועבור λ קטן נעודד הפסד $\ell^h(w,S)$ קטן.
$Soft-SVM$ מצמצם את האובייקט הבא: $\lambda\|w\|^2+\ell^h(w,S)$.

את אלגוריתם $Soft-SV$ ניתן לממש באמצעות תכנון ריבועי.
תכנון ריבועי הוא אלגוריתם הממזער את האובייקט $\min_{z\in\mathbb{R}^d}\frac{1}{2}\cdot z^THz+\langle u,z\rangle$ תחת האילוצים $Az\geq v$.
נוכל להגדיר ולפתור את $Soft-SVM$ עבור הערכים הבאים של H,u,A,v כקלט לתוכנית ריבועית:

- $u=\left(0,\dots,0,\frac{1}{m},\dots,\frac{1}{m}\right)\in\mathbb{R}^{d+m}$ כאשר ישנם d קורדינטות שערכן 0, ועוד m קורדינטות שערכן $\frac{1}{m}$.
- $H=2\lambda\cdot\begin{pmatrix}I_d & 0_{d\times m}\\ 0_{m\times d} & 0_{m\times m}\end{pmatrix}\in\mathbb{R}^{(d+m)\times(d+m)}$.
- $v=(0,\dots 0,1,\dots 1,)\in\mathbb{R}^{2m\times 1}$ כאשר ישנם m קורדינטות שערכן 0, ואחריהן m קורדינטות שערכן 1.
- $A=\begin{pmatrix}0_{m\times d} & I_m\\ Y_iX_{i_{m\times d}} & I_m\end{pmatrix}\in\mathbb{R}^{(2m)\times(d+m)}$.
- וקטור התוצאה שלנו יהיה $z=(w_1,\dots,w_d,\xi_1,\dots,\xi_m)$, כאשר $w_1,\dots,w_d$ מבטאים את המפריד הלינארי שהתקבל מאלגוריתם $Soft-SVM$.

בהינתן מפריד לינארי w ודוגמא x נחזה את התווית באופן הבא: $y=sign(\langle w,x\rangle)$.

<u>תוצאות בדיקת המסווג:</u>
עבור מדגם בגודל 2000 המכיל תמונות עם הספרות 0,1 ועבור קבוצת הערכים אופציונלים $\{1,10,100\}$ עבור הרגולטור λ, וחלוקה של המדגם ל-5 חלקים.
קיבלנו שהערך האופטימלי שהחזיר אלגוריתם cross validation הינו $\lambda=1$.
השגיאה שהתקבלה על מדגם מבחן בגודל 500 הינה: 0%, כלומר המדגם היה פריד ומצאנו מפריד שהצליח להפריד אותו לחלוטין.
עבור מדגם המכיל תמונות עם הספרות 3,5, השגיאה שהתקבלה הינה: 6%.

6. למידת לא מונחית - *Clustering* (בונוס)

למידה לא מונחית היא למידה בה נקבל כקלט מדגם **לא** מתויג.

ניתן לבצע למידה לא מונחית כשלב מקדים ללמידה מונחית, על מנת לכווץ את הייצוג של הדוגמאות - PCA.

כמו כן, ניתן לעשות בה שימוש על מנת לארגן את המידע שיש בידנו – $clustering$.

אנחנו בחרנו להתמקד ב $clustering$, שיטה זו מארגנת את המידע שיש בידנו, לצורך העניין את המדגם S, לקבוצות שנקראות $clusters$. דוגמה לשימוש היא חלוקה של מאגר תמונות לאלבומים שונים.

במקרה שלנו נרצה לחלק את התמונות של הספרות מ MNIST לעשר קבוצות, כך שכל קבוצה מייצגת ספרה שונה.

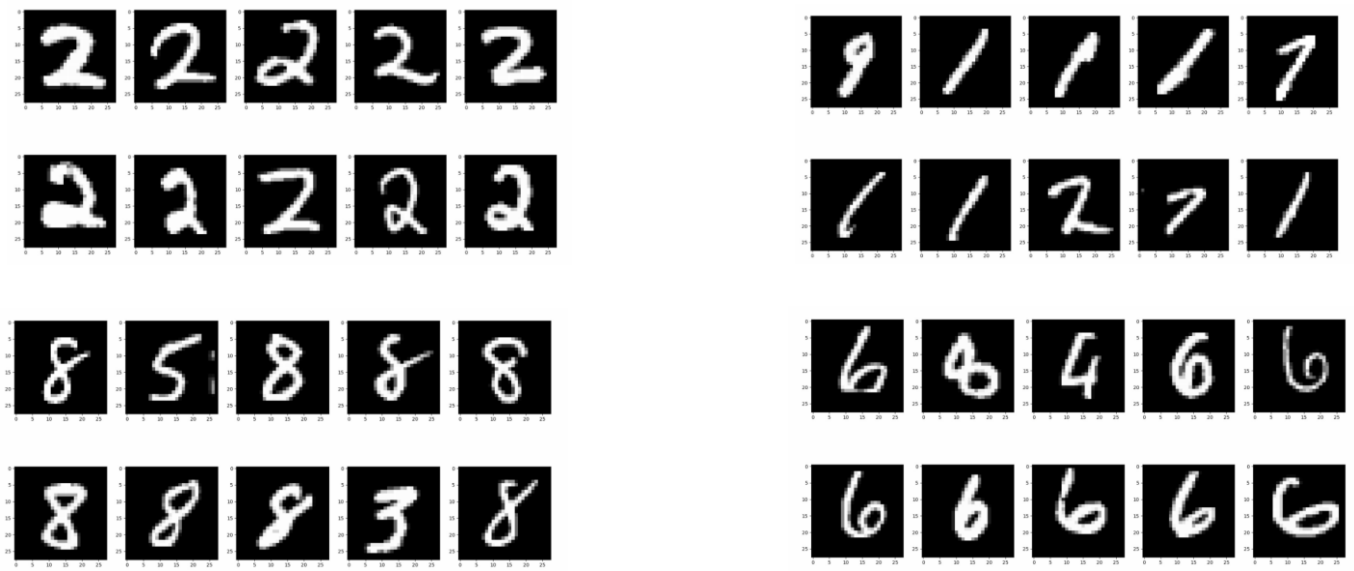
כדי לבצע את פעולת החלוקה לקבוצות נצטרך לעשות שימוש בפונקצית מרחק, אנו נעשה שימוש בפונקצית המרחק האוקלידי, כלומר $\|\cdot\|_2$.

האלגוריתם שלמדנו על מנת לבצע $clustering$ נקרא $K-means$.
על מנת להציג אותו נסמן ב C_i את הקלסטר ה- i (כלומר קבוצה של דוגמאות), ונסמן ב μ_i את מרכז ה- i (מרכז הוא איבר ממרחב הדוגמאות).
אלגוריתם $K-means$ הוא אלגוריתם איטרטיבי אשר עושה מעבר בין קלסטרים למרכזים ובכל שלב מעדכן את ערכיהם על מנת להשיג את המטרה.
האלגוריתם מתבסס על העבודה שניתן בקלות בעזרת מרכז μ לעבור להגדרה של קלסטר C, ולהיפך.

<u>קלט האלגוריתם:</u> מדגם לא מתויג S, ומספר הקלסטרים המבוקש k.
<u>פלט האלגוריתם:</u> k מרכזים $\mu_1, \dots, \mu_k$ (המגדירים k קלסטרים).
<u>שלבי האלגוריתם:</u>

- איתחול רנדומלי של המרכזים $\mu_1, \dots, \mu_k$.
- כל עוד האלגוריתם לא התכנס (כלומר כל עוד הרכבי הקלסטרים משתנה מסיבוב לסיבוב) בצע:
  - לכל $i \in \{1, \dots, k\}$ הגדר קלסטר $C_i := \{x \in S \mid i = \underset{1 \le j \le k}{\operatorname{argmin}} \|x - \mu_j\|\}$.
  - לכל $i \in \{1, \dots, k\}$ הגדר את המרכז של הקלסטר C_i להיות $\mu_i := \frac{1}{|C_i|} \cdot \sum_{x \in C_i} x$.
- החזר $\mu_1, \dots, \mu_k$.

<u>תוצאות:</u> דגמנו 10 דוגמאות מקלסטרים שונים, בציפייה למצוא בכל קלסטר רוב של דוגמאות מסוג כלשהו.



7. מימוש האלגוריתמים

בחרנו לממש את האלגוריתמים $knn, cross\ validation, naive\ bayes, soft - SVM, kmeans$.

ביצענו בדיקות על כל אחד מהמסווגים שיצרנו, והתוצאות היו משביעות רצון.

את הקוד כולו נצרף יחד עם דו"ח זה.