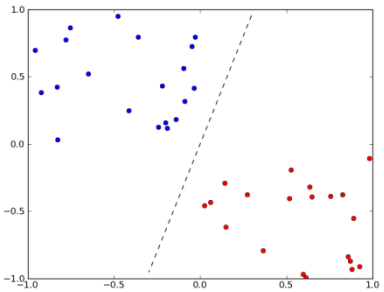
תרגיל 5 בקורס מבוא למדעי המחשב

אלגוריתם הפרספטרון



למדנו בכיתה את אלגוריתם הפרספטרון הלקוח מתורת הלמידה החישובית. באמצעות הפרספטרון אנו יכולים למצוא מפריד לינארי שיבצע עבורנו סיווג (classification) של נתונים. כלומר, לחזות תוצאות עתידיות על בסיס דוגמאות מתוייגות שניתנו לאלגוריתם בשלב האימון.

*בתמונה בצד - מפריד לינארי בין נקודות "מתויגות" כאדומות וכחולות.

בתרגיל זה תיישמו בעצמכם את אלגוריתם הפרספטרון בפייתון. לאחר היישום תעשו שימוש באלגוריתם בכדי לתייג נתונים. תתנסו הן בהפרדת נתונים דו מימדיים (דאטא הניתן להצגה במישור) והן בלמידת הפרדה במרחב ממימד גבוה יותר.

התרגיל מורכב משלושה חלקים. בחלק הראשון (משימות 0-2) תיישמו את אלגוריתם הפרספטרון ופונקציה הבודקת את הצלחתו על נתוני מבחן. בחלק השני תשתמשו בפונקציות אלו בכדי להפריד דאטא רב מימדי – תמונות בכתב יד של המספרים 4 ו– 7 (משימות 3-5). במידה והצלחתם בחלק א, תוכנתכם תצליח להבדיל בין תמונות אלו. החלק השלישי (משימה 6) הינו משימת בונוס (השווה 10 נק' נוספות).

שימו לב: חלק א הינו החלק הארוך והמשמעותי של התרגיל. אולם בכדי לקבל ציון מלא עליכם להותיר זמן גם לחלק השני של התרגיל.

עליכם ליישם את כל הפונקציות בקובץ perceptron.py המופיע באתר. חתימות הפונקציות כבר מופיעות בקובץ, אל תשנו אותן. אפשר וכדאי להוסיף פונקציות עזר בקוד.

ספקנו לכם קובץ נוסף intro2cs_ex5.py המכיל מספר פונקציות ויזואליזציה (פרטים נוספים למטה) וכן פונקצית עזר לקריאת קבצים. לצרכי בדיקה עצמית והתרשמות אתם יכולים להשתמש בקובץ זה במהלך כתיבת התרגיל. בכדי לעשות זאת ייבאו את הקובץ (import intro2cs_ex5).

בעת הגשת התרגיל אל תעשו import לקובץ ואל תקראו לאף אחת מהפונקציות שלו.

את הקובץ הנ"ל כמו גם את קבצי הטקסט בהמשך התוכנה עליכם לחלץ מתוך ex5.zip (הורידו מאתר הקורס) בתיקיית העבודה שלכם.

הנכם רשאים להניח בכל מקום בתרגיל כי הקלט הוא חוקי ותקין.

שימו לב כי intro2cs_ex5 עושה שימוש בשתי ספריות: numpy & matplotlib – לאלו מבינכם המשתמשים בסביבה שונה מcs מוצעת כאן אפשרות פשוטה להתקנת ספריות אלו.

חלק א

משימה 0: מכפלה פנימית בין וקטורים.

עליכם לממש פונקציה שתקרא **dot** שתממש מכפלה פנימית - פעולה המקבלת שני וקטורים ותוצאתה היא סקלאר (מספר ממשי). (http://en.wikipedia.org/wiki/Dot_product).

פרמטרים:

- **A** – רשימה באורך n של איברים מספריים (float or int) – שתייצג וקטור בR^n.
- **B** – רשימה נוספת באורך n של איברים מספריים, וקטור נוסף בR^n.
  ניתן להניח שהרשימות באותו אורך ושהן אינן ריקות.

ערך ההחזרה:

- הערך המתמטי של המכפלה הסקלארית של A·B (A dot B). הנוסחה לחישוב A·B היא : $\sum_{i=0}^{n} A_i * B_i$

משימה 1: הפרספטרון.

עליכם לממש פונקציה שתקרא perceptron שתממש את אלגוריתם הפרספטרון. במשימה זו נקבל רשימה באורך כלשהו של ווקטורים ב R^n - ותיוגים לווקטורים אלה על פיהם נחשב מפריד לדאטא.

פרמטרים:

- **data** – רשימה של רשימות. נגדיר כי גודל הרשימה החיצונית הינו m וכל אחד מאיבריה הינו רשימה בגודל n (וקטור בR^n). הנכם יכולים להניח שהקלט חוקי (כל איבריה של הרשימה החיצונית הינם רשימות בגודל אחיד של איברים מספריים).
- **labels** – רשימה באורך m של מספרים, כך שהאיבר במקום ה - i ברשימה הוא התיוג של השורה ה - i ב - data (תת הרשימה של - data במקום ה - i). ערך התיוג הוא חיובי (1) או שלילי (-1). גם כאן הנכם רשאים להניח כי הקלט חוקי. התיוגים ייוצגו באמצעות באמצעות int – 1 או באמצעות float – 1.0.

ערכי ההחזרה:

- הפלט:
  פלט הפונקציה יהיה רשימה - w באורך - n ומשתנה b – bias שהינם הפלט המוחזר על ידי אלגוריתם הפרספטרון לאחר שהגיע להפרדה מלאה של הדאטא. במידה ולא נמצאה הפרדה תוך מספר העדכונים המותר (פרטים למטה) עליכם להחזיר צמד Nones. החזירו את w ו - b כ – tuple:

  return (w, b).

  במידה ולא נמצאה הפרדה החזירו tuple המכיל שני None:

  return (None, None)

את אלגוריתם הפרספטרון למדנו בשיעור. כזכור לכם בסופו, במידה וקיימת הפרדה, הכפלה של w בשורה ה - i ב - data והפחתת הסכום ב - b תתן תוצאה חיובית אם התיוג ה - i הינו חיובי ולהפך עבור תיוג שלילי : $sign(\vec{w} \cdot \overrightarrow{data}^i - b) = label^i$.

האלגוריתם מתקדם כך שבכל איטרציה אנחנו מתמקדים בשורה אחרת ב - data ומכפילים אותה ב - w (באמצעות מכפלה פנימית). במידה וסימנה של תוצאת הכפל (פחות ה - bias) זהה לתיוג המתאים ב - labels אנו ממשיכים לשורה הבאה ב - data.

אחרת (במידה וסימנה של שורת הכפל שונה מהתיוג): נעדכן את - w כך ש - w החדש יהיה שווה ל - w הישן ועוד מכפלתם של השורה ה - i ב - data כפול התיוג שלו, ו - b יהיה שווה ל - b הישן פחות התיוג ה – i (על פי המשוואות במצגות הקורס). לאחר מעבר על כל השורות של data ועדכונים מתאימים של w וb, נחזור שוב לשורה הראשונה ונחל את הריצה מחדש.

נסיים את ריצת האלגוריתם (נאמר שישנה הפרדה) לאחר שעבור כל השורות בdata מתקיים: $sign(\vec{w} \cdot \overrightarrow{data}^i - b) = y^i$, כלומר מצאנו מפריד לינארי.

כיצד נמנע מריצה אינסופית במידה ואין הפרדה? עליכם לספור את מספר הפעמים בהם עדכנתם את w. במידה ובריצתכם עדכנתם את w יותר מ - 10*m פעמים, נניח כי לא קיימת הפרדה ותדרשו להחזיר צמד Nones.

ביכולתכם לבדוק את התקדמות האלגוריתם. נשים לב כי אם data הינו דו מימדי – כלומר רשימה של רשימות באורך 2 ביכולתנו להציג אותו במרחב. לצורך בדיקה עצמית - קראו במהלך ריצת האלגוריתם, **לאחר כל עדכון של w**, לפונקציה שיישמנו עבורכם בקובץ intro2cs_ex5 באופן הבא: intro2cs_ex5.show_perceptron(data,labels,w,b), כאשר שני הארגומנטים הראשונים הם אלו שקבלתם כקלט. קריאה לפונקציה זו תציג לכם הן כיצד נראה המרחב והן את הצגתם הגראפית של w וb. אם ישמתם את האלגוריתם באופן מוצלח תוכלו לראות את הישר המסווג שלכם מתקדם לעבר מיקומו הנכון. **הפונקציה show_perceptron מניחה כי data הינו דו מימדי ועל כן לא תעבוד על תתי רשימות באורך שונה מ2. כמו כן מניחה הפונקציה קלטים המתאימים לdata מסוג זה (w באורך 2). מיד יוצגו עבורכם מקרים כאלו (דו מימדיים) ועבורם ביכולתכם לבדוק את הצלחתכם. במידה וקראתם ל-show_perceptron יפתח חלון המציג את data ואת w וb הנוכחים. בכדי להמשיך בריצת הקוד על המשתמש יהיה לסגור את האיור שנפתח.**

שימו לב בעת הגשת התרגיל אין לייבא את הקובץ intro2cs_ex5 ואין לקרוא לאף אחת משיטותיו. על כן ודאו כי בקובץ המוגש אין קריאה לshow_perceptron (אך אנו ממליצים מאוד להשתמש באופשרות זו במהלך הכנת התרגיל)

כעת ביכולתכם לבדוק את עצמכם:

- השתמשו בקבצים: data.txt, labels_AND.txt, labels_XOR.txt בכדי לבדוק את הצלחת האלגוריתם. הורידו את קובץ הzip מאתר הקורס וחלצו אותו בתיקייה בה אתם עובדים.
  ביכולתכם לקרוא קבצים באמצעות הפונקציה loadtxt הנמצאת בקובץ intro2cs_ex5.
- לאחר ייבואו שימוש בפונקציה באופן הבא:
  data = intro2cs_ex5.loadtxt('data.txt')
  יקרא את התוכן של הקובץ data.txt לתוך רשימה של רשימות, כך שאורכה של הרשימה החיצונית הינו 4 ואורכה של כל רשימה פנימית הינו 2.

  קראו באופן דומה את קבצי התיוג האפשריים labels_AND.txt, labels_XOR.txt:
  labels = intro2cs_ex5.loadtxt('labels_AND.txt') (תתקבל רשימה אחת באורך 4).

- כעת הריצו את אלגוריתם הפרספטרון פעם אחת עם התיוג labels_AND ופעם אחת עם התיוג labels_XOR . במידה וישמתם נכונה את האלגוריתם תצליחו למצוא הפרדה עבור labels_AND אך לא עבור labels_XOR.

- data הינו משתנה דו מימדי ועל כן אנו ממליצים לבדוק את התקדמות האלגוריתם באופן גראפי באמצעות קריאה ל- show_perceptron לאחר כל עדכון של w.
- שימו לב כי גם במקרה זה אין לייבא את Intro2cs_ex5 במהלך הגשת התרגיל.

- ביכולתכם לבדוק את הצלחת האלגוריתם גם על קבצי נתונים גדולים יותר. הקבצים data_2D.txt וקבצי התיוגים המתאימים labels_2D_sep.txt ו- labels_2D_no_sep.txt מכילים מידע דו מימדי עבור 889 נקודות במישור ושני קבצי תיוגים כך שהראשון מאפשר הפרדה והשני לא. קראו קבצים אלו באופן דומה ובמידה ויישמתם באופן מוצלח את פונקצית הפרספטרון תוכלו לראות את התקדמות האלגוריתם עד להצלחתו במקרה הניתן להפרדה וכשלונו במקרה השני. גם נתונים אלו הינם דו מימדים וביכולתכם לחזות בהתקדמות האלגוריתם באמצעות show_perceptron

<u>משימה 2: בדיקת התיוג.</u>

במשימה זו אנו בוחנים את הצלחתו של הפרספטרון. באמצעות פונקציה זו נוכל לקבוע עד כמה הצליח האלגוריתם ללמוד את הדאטא – לקבוע באמת את חוקיו. ניתן להשתמש בפונקציה כזו בכדי לאמן את הפרספטרון על נתונים שנקרא להם נתוני אימון (train data) ולבחון את הצלחתו על נתונים שלא שמשו לצורך בנייתו של w – נתוני מבחן (test data).

עליכם לכתוב פוקנציה בשם generalization_error. פונקציה זו תקבל כקלט רשימה של רשימות בגודל m X n (רשימה של - m רשימות שאורך כל אחת הוא - n), את וקטור התיוגים שלה; ווקטור מפריד אפשרי וbias. תפקידה של הפונקציה הזו הוא לבדוק האם עבור כל רשימה בקלט הוקטור המפריד אכן מצליח למצוא את התיוג הנכון.

פרמטרים:

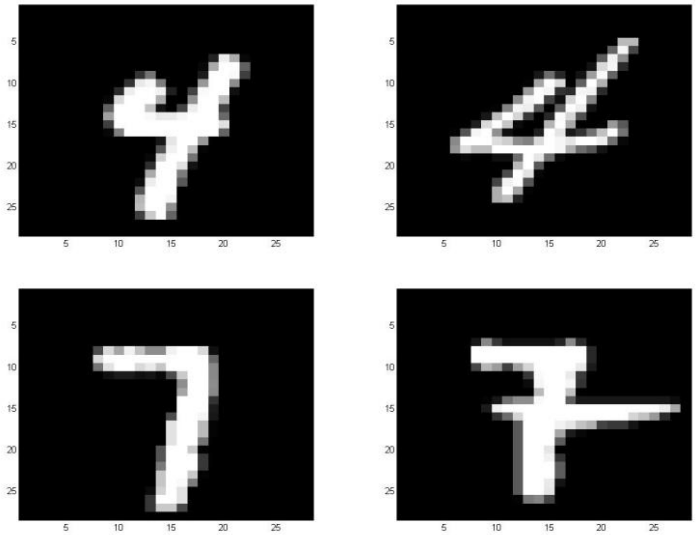
data – רשימה של רשימות בגודל m X n, הנכם רשאים להניח כי הקלט תקין – כל האיברים ברשימה החיצונית הינם רשימות וכל הרשימות הפנימיות הינן באורך אחיד.
labels – רשימה של תיוגים בגודל m. כמו מקודם תיוגים יהיו בעלי ערך חיובי (1) או שלילי (-1).
- **w** – רשימה בגודל n שהינה פלט אפשרי של פונקצית הפרספטרון – כלומר וקטור הפרדה אפשרי (מוצלח או לא) לdata.
- **b** משתנה מסוג float או int שהינו ה - bias שהוחזר על ידי הפרספטרון.

ערך ההחזרה: רשימה בגודל m שאיבריה מטיפוס - int כך שבמקום ה- i יהיה 0 במידה ו-w ו-b חזו נכון את התיוג ה - i ב - labels (עבור השורה ה - i ב – data) או 1 במידה וחזו באופן שגוי.

חלק ב

זיהוי כתב יד באמצעים אוטומאטיים היא משימה אלגוריתמית מסובכת כיוון וקיימת שונות רבה מאוד בין פרטים שצריכים להיות מסווגים באופן דומה. בחלק זה תשתמשו בפונקציות שיישמתם בחלק א בכדי להפריד מידע המיוצג ע"י וקטור רב מימדי – תמונות של המספרים 4 ו-7 בכתב יד. התמונות שונות אחת מהשנייה, וכל יצוג של המספר 4 שונה במקצת מהאחר. משימתכם תהייה להשתמש באלגוריתם הפרספטרון כדי "לקרוא" את כתב היד ולהבדיל בצורה טובה ככל הניתן בין הספרות 4 ו-7.

תמונות אפשריות של המספרים 4 ו - 7 בכתב יד הלקוחים מהנתונים עליהם תעבדו בתרגיל



<u>משימה 3: המרת הנתונים</u>

ממשו את הפונקציה vector_to_matrix. פוקנציה זו מקבלת רשימה באורך 784 ויוצרת ממנה רשימה של רשימות באורכים 28*28 המייצגת תמונה. כל איבר ברשימת הרשימות ייצג פיקסל למסך וכך נקבל תמונה. פיקסל הינו איבר אי שלילי אשר מייצג רמת בהירות בין שחור (0) ללבן (1). אוסף של פיקסלים ומיקומים במישור יתן לנו תמונה כמו אלו המוצגות לעיל.

פרמטרים:

- **vec** - רשימה בגודל 784 של איברים מסוג float (גם כאן אתם רשאים להניח כי הקלט תקין).

ערך ההחזרה: matrix - רשימה של רשימות בגודל 28 .

שימו לב כי עליכם ליצור את הפלט כך ש:

- האיבר ה0 ברשימה ה0 של matrix יהיה האיבר ה0 בvec.
- האיבר ה27 ברשימה ה0 יהיה האיבר ה27 בvec.
- האיבר ה0 ברשימה ה1 יהיה האיבר ה28 בvec.

חשבו: האם ביכולתכם לבנות את הפונקציה כך שתתאים גם למקרה הכללי בו נרצה להפוך וקטור (בגדלים שונים) למטריצה (ריבועית). בנו את הפונקציה בצורה זו.

בדקו את עצמכם:

- הקובץ data_47.txt מכיל מטריצה מגודל 784 X 1000 כך שכל שורה בגודל 784 הינה תמונה של הספרה 4 או 7. קראו את תוכן הקובץ כפי שהודגם במשימה 2:
data_47 = intro2cs_ex5.loadtxt('data_47.txt')

- יישמנו עבורכם פונקציה המקבלת מטריצה ומציגה אותה כתמונה. בכדי להשתמש בה ייבאו את הקובץ intro2cs_ex5 וקראו לפונקציה
intro2cs_ex5.show_number(matrix), כך ש - matrix הינו הפלט של vector_to_matrix.
- **כעת ביכולתכם לבדוק את עצמכם** – הפכו את אחת הרשימות הפנימיות של data_47 למטריצה והפעילו את show_number עליה. במידה ויישמתם נכונה את המשימה תתקבל תמונה של המספר 4 או המספר 7 בכתב יד (בדומה לתמונות לעיל).
- **שוב, שימו לב כי אין לייבא את הקובץ הנתון או לקרוא לshow_number בשום שלב של ריצת הקוד המוגש! (כלומר אין לקרוא לפונקציה במהלך ישומן של אחת מהפונקציות המוגשות). ביכולתכם להשתמש בקובץ זה רק לצרכי בדיקה עצמית. מחקו קריאות אלו לפני הגשת התרגיל.**

<u>משימה 4: למידת הספרות 4,7</u>

כתבו פונקציה הנקראת classifier_4_7 המקבלת כקלט רשימה של רשימות, כך שכל רשימה פנימית מייצגת את המספר 4 או 7 ורשימה נוספת של תיוגים והחזירו את הפלט של אלגוריתם הפרספטרון.

פרמטרים:

- data – רשימה של רשימות מגודל 784 x 1000. כל איבריה של הרשימה החיצונית הינם רשימות וכל איברי הרשימות הפנימיות הינם איברים מסוג float.
- labels – רשימה של תיוגים – כך שערכים חיובים ייצגו את המספר 7 וערכים שלילים ייצגו את המספר 4.

ערך ההחזרה:

<u>Tuple בעל שני איברים :</u>

- **w** – רשימה בגודל 784 במידה ונמצאה הפרדה באלגוריתם הפרספטרון או רשימה ריקה במידה ולא (הפלט של פונקציה perceptron).
- **b –** משתנה מסוג float או int שהינו ה - bias שהוחזר על ידי פונקצית הפרספטרון.

שימו לב שכעת כל איבר בדאטא שלנו הוא בR^{784}, על כן לא ניתן ליצגו במישור דו מימדי ואין ביכולתכם להשתמש בshow_perceptron בכדי לצפות בהתקדמות האלגוריתם.

הקובץ labels_47.txt מכיל רשימה נכונה של תיוגים עבור הוקטורים בdata_47.txt . יישום נכון של הקוד יאפשר לכם למצוא מסווג בין תמונות המייצגות את המספר 4 לתמונות המייצגות את המספר 7.

<u>משימה 5: בדיקת המסווג על מידע חדש.</u>

בחלק זה תקבעו את הצלחתו של הפרספטרון בלמידת החוקים מאחורי הפרדת תמונות 4 ו - 7.
כזכור במשימה 2 יישמתם את הפונקציה generalization_error המשתמשת בוקטור מפריד נתון בכדי לבדוק את הצלחתו על נתוני מבחן. כמו כן במשימה 3 הורדתם טבלה המייצגת תמונות של המספרים 4 ו - 7 והשתמשתם בפרספטרון כדי ליצור רשימה w וbias b המפרידים בין התמונות.
כעת נשתמש ב - w וb אלו בכדי לנסות לקבוע עבור תמונות חדשות האם הן מייצגות את המספר 4 או 7.

יישמו את הפונקציה test_4_7 המקבלת כקלט נתוני אימון ונתוני מבחן ומחזירה רשימה, שאורכה כאורך נתוני המבחן, הקובעת על אילו מנתוני המבחן האלגוריתם הצליח ועל אילו לא.

פרמטרים:

- **train_data** - רשימה של רשימות בגודל m X n.
- **train_labels** – רשימת תיוגים בגודל m.
- **test_data** – רשימה של רשימות בגודל k X n. הנכם רשאים להניח כי הקלט תקין - כלומר כל איבר בtest_data הינו רשימה – וכל רשימה כזו מכילה מספר זהה של איברים לרשימות הפנימיות בtrain_data.
- **test_labels** – רשימת תיוגים בגודל k

ערך ההחזרה:

עליכם להחזיר tuple המכיל:

- **w** – המפריד שנמצא על ידי הפרספטרון על נתוני האימון
- **b –** biasה שנמצא בהתאמה
- **errors** – רשימה של מספרים שלמים (0 או 1) – הפלט של הפונקציה generalization_error כשזו נקראת על נתוני המבחן עם w וb שנמצאו על ידי פונקצית הפרספטרון.

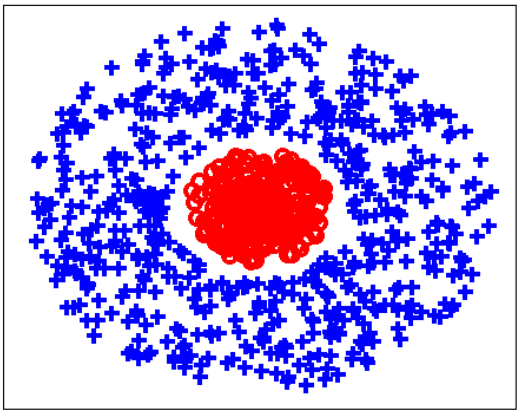
  החזירו את הtuple באופן הבא:
  return (w,b,errors)
- במידה והפרספטרון לא זהה מסווג (כלומר w וb הוחזרו כNone) החזירו שלשת Nones: retrun(None,None,None)

<u>משימה 5.א</u>

קראו את תוכן data_47.txt , labels_47.txt והשתמשו בהם כנתוני האימון.
קראו את תוכן test_data_47.txt, test_labels_47.txt והשתמשו בהם כנתוני המבחן.
כתבו ב**README** – האם הצלחתם להפריד את נתוני המבחן באמצעות ה - w שהושג על נתוני האימון? היכן נכשלתם? האם אתם יכולים לזהות מדוע נכשל האלגוריתם במקומות בו נכשל (פתחו את התמונות של האיברים שלא צלחו).

חלק ג - בונוס (10 נק')

א. קראו את התוכן של data_bonus.txt ו- labels_bonus.txt. זהו דאטא דו מימדי הנראה כך.



ב. כתבו בREADME תחת סעיף 6 – מהי הבעייה? מדוע האלגוריתם כפי שאנו מכירים אותו כעת אינו יכול להפריד נתונים אלו. כיצד ניתן לדעתכם לשפר את האלגוריתם בכדי שיוכל להחזיר מפריד גם עבור מקרה זה?

הוראות הגשה:

- עליכם להגיש את הקובץ ex5.tar
- קובץ זה לא יכיל דבר מלבד קובץ README ואת הקובץ perceptron.py.
- ודאו כי בקובץ המוגש אין ייבוא של intro2cs_ex5 ואין קריאה לאף אחת מהפונקציות בו. (כלומר אל תגישו תרגיל הכולל קריאה לloadtxt, לshow_perceptron או לshow_number.
- מועד אחרון להגשת התרגיל הינו **4.12.2014 בשעה 21:00**
- שימו לב כי כפי שצוין לעיל אין לכלול פונקצית main בקובץ perceptron.py. פונקציות אחרות בנוסף לאלו הנדרשתם אליהם מותרות כרצונכם.
- קובץ הREADME צריך להיות על פי התבנית המופיעה באתר הקורס בנוסף למשימות ב5א והבונוס במשימה ב6.

בהצלחה!