זיהוי תמונות של ספרות בכתב-יד

מבוא

בתרגיל זה נכתוב, בפייתון, פונקציות שתאפשרנה לממש מערכת שתקרא תמונות של ספרות (0 עד 9) ותקבע מה הספרה שבתמונה. לא נדון בכתיבת פונקציית main ש"תחבר" את המרכיבים ביחד. אנחנו משאירים לכן משימה זו.

בתרגיל נשתמש באוסף התמונות הנמצא ב: http://yann.lecun.com/exdb/mnist/. עליכן להוריד מהאתר (ולבצע unzip) את ארבעת הקבצים:

train-images-idx3-ubyte.gz
train-labels-idx1-ubyte.gz
t10k-images-idx3-ubyte.gz
t10k-labels-idx1-ubyte.gz

בהמשך נסביר מה יש בקבצים אלה.

אנחנו נתקדם שלב שלב. ובכל שלב נסביר כיצד לכתוב פונקציה לביצוע משימה מסוימת. הסברים על פקודות או פונקציות שאינכן מכירות עליכן לחפש ברשת. חלק מהקוד נמצא בקובץ digits.py ועליכן יהיה להשלים את מה שחסר.

קריאת ה-dataset

בארבעת הקבצים שהורדתן יש נתונים של 70000 תמונות של ספרות (0 עד 9) הכתובות בכתב-יד. לכל תמונה נתון מהי הספרה המצולמת בתמונה. הפונקציות שנכתוב תשתמשנה בנתונים שבקבצים ובאלגוריתם השכן הקרוב (שיוסבר בהמשך) כדי לקבוע מהי הספרה הכתובה בתמונות שאנחנו נצלם.

עליכן להבין, קודם כל, את המידע שיש בקבצים אלה. ההסבר נמצא באתר ממנו הורדתן את התמונות. עיקרו בפסקאות הראשונות (עד הפסקה שמתחילה ב: With some classification methods), ובפרק FILE FORMATS FOR THE MNIST DATABASE.

המשימה הראשונה היא כתיבת פונקציה שתיצור את ה-dataset. ה-dataset היא רשימה (list) של רשימות. כל רשימה מכילה את הנתונים של אחת התמונות. לכל תמונה 785 מספרים שלמים: 784 מייצגים את כל הפיקסלים שבתמונה, שורה אחרי שורה (הערך של כל אחד מהם בין 0 ל 255), ומספר נוסף המייצג את הספרה שבתמונה.

הפונקציה שיוצרת את ה-dataset היא הפונקציה idx2list. היא מקבלת את השמות של שני קבצים: images – קובץ נתוני התמונות ו-labels – קובץ התוויות המגדיר מה הספרה בכל תמונה. שלוש השורות הראשונות של הפונקציה:

```
lst = []  #initializes the returned list
fimages = open(images,"rb")#"rb" is "read bytes"
flabels = open(labels,"rb")
```

lst היא הרשימה שתכיל את ה-dataset, בסופו של דבר (שימו לב שהפקודה האחרונה בפונקציה מחזירה את הערך של lst). ערכה מאותחל לרשימה ריקה.

שתי השורות הבאות פותחות את הקבצים לקריאה של בייטים (כפי שקראתם, המידע בקבצים מאוחסן בבייטים).

כעת יש לקדם את הסמן בקבצים (שנמצא כרגע על בייט 0 – הביט הראשון בקובץ) לבייט ה"מעניין" הראשון. עליכן ללמוד מהו הבייט ה"מעניין" הראשון מתוך האתר ממנו הורדתן את הקבצים, ולברר (אם אינכן יודעות) איזו פונקציה מקדמת את הסמן בקובץ למיקום מסוים. עליכן להשלים את הקוד שעושה זאת.

לולאת ה-while קוראת את כל המידע שבקבצים עד סופם, בייט אחרי בייט. כאשר הסמן מגיע לסוף הקובץ, ערכו של הבייט הנקרא הוא ""b. לכן מבנה הלולאה לקריאה מקובץ הוא:

```
x = fimages.read(1)#Reads the first byte
while x != b"": #While not end of file
     #Process the data
     x = fimages.read(1) #Read the next byte
```

השורה הראשונה בתוך הלולאה:

```
line = [ord(x)]
```

מציבה ב-line את המספר שבבייט הראשון (ord הופך את ערך הבייט שנקרא למספר שלם). כעת עליכן להשלים את הלולאה שתקרא את שאר 783 הבייטים ותצרף את ערכם ל-line. מכיוון שמספר הפיקסלים (784) ידוע מראש, אפשר להשתמש בלולאת for.

לאחר קריאת כל ערכי הפיקסלים לתוך line, השורה:

```
line += [ord(flabels.read(1))]
```

מצרפת ל-line את הערך מתוך קובץ התוויות המציין איזו ספרה מצולמת בתמונה.

השורה:

```
lst += [line]
```

מצרפת את הנתונים של התמונה, שנקראה באיטרציה זו של לולאת ה-while, ל-lst. השורה האחרונה בלולאה, קוראת את הבייט הבא, כפי שתואר קודם.

שלוש השורות האחרונות של הפונקציה מובנות מאליהן, וכל שנותר לכן לעשות הוא להשלים את הקוד החסר. כיצד תוודאו שהקוד שכתבתן נכון? בדוקנה שמספר הרשימות ב-lst הוא כמספר התמונות, שאורך כל רשימה ב-lst הוא 785, שערכי הפיקסלים בין 0 ל-255 ושערכי התוויות בין 0 ל-9. בהמשך נראה איך לראות את התמונות שב-dataset.

הערה: הפונקציה נכתבה, כאילו, מספר התמונות בקובץ אינו ידוע. זה לא נכון. מספר התמונות כתוב בתוך הקובץ וניתן היה לקרוא אותו. יותר נוח וקל להתעלם מכך מאשר "לפענח" את מספר התמונות מתוך הקובץ.

הצגת התמונות

הפונקציה data2img מקבלת רשימה, lst, המכילה שורה מתוך ה-dataset. הפונקציה מחזירה את התמונה הנוצרת מהפיקסלים שבשורה. הפונקציה נעזרת בפונקציה Image.fromarray, מתוך החבילה pillow, כדי לבצע זאת. הפונקציה מקבלת מערך דו-מימדי ומחזירה את התמונה שהוא מכיל. מערכים ניתן ליצור בעזרת החבילה numpy (חבילה שימושית מאוד, שכדאי שתכירו).

קודם כל יוצרת הפונקציה רשימה של רשימות בשם lst2d. כל רשימה, ב-lst2d, מכילה את הפיקסלים של שורה בתמונה. lst2 נוצרת בשורות הבאות:

```
lst2d = []
for i in range(ROWS):
  lst2d += [lst[COLS*i  :  COLS*(i+1)]
```

בכל איטרציה של הלולאה מתווספת שורה ל-lst2d. קל להבין זאת אם מציגים בטבלה את האינדקסים בהם נחתכת lst. זכרו ש: ROWS=COLS=28, וש-lst[x:y] יוצרת רשימה שהאיבר האחרון שלה הוא lst[y-1].

i = 0	COLS*i = 0	COLS*(i+1) = 28
i = 2	COLS*i = 28	COLS*(i+1) = 56
i = 3	COLS*i = 56	COLS*(i+1) = 84
. . .	. . .	. . .
i = 27	COLS*i = 756	COLS*(i+1) = 784

אחרי יצירת lst2d, numpy.array יוצרת ממנה מערך דו-מימדי, numpy.unit8 ממירה את המספרים במערך לשלמים ללא סימן המאוחסנים בבייט יחיד, ו-Image.fromarray יוצרת תמונה מהנתונים שבמערך. תוכלו ללמוד יותר על פונקציות על-ידי חיפוש ברשת.

הפונקציה data2image יוצרת תמונה אבל לא מציגה אותה. הפונקציה display_image מקבלת את ה-dataset, ds, ואת השלם i, ומציגה את התמונה שבשורה ה-i ב-ds (אם יש שורה כזאת). עליכן להשלים את השורה שמציגה את התמונה שב-im. חפשו בתיעוד של המחלקה Image את הפונקציה שמציגה תמונה.

אחרי שתשלימו את הפונקציה. תוכלו, בתכנית הראשית, להציג תמונות מתוך ה-dataset. האם התמונות מכילות ספרות בשחור על רקע לבן או תמונות בלבן על רקע שחור?

אלגוריתם השכן הקרוב (Nearest neighbor)

אחרי שטענו את ה-dataset וראינו אילו תמונות יש ב-dataset. נוכל לכתוב פונקציה, שמקבלת תמונה ומעריכה איזו ספרה מצולמת בה. זאת נעשה בעזרת אלגוריתם השכן הקרוב. בהינתן תמונה X, האלגוריתם עובר על כל התמונות ב-dataset, ומוצא את התמונה שהכי קרובה ל-X (השכן הקרוב ל-X). האלגוריתם יעריך שהספרה המצולמת ב-X היא הספרה המצולמת בשכן הקרוב ביותר שלה.

כיצד מודדים קירבה בין שתי תמונות? כל תמונה, מבחינת האלגוריתם, היא רשימה של 784 מספרים, בין 0 ל-255. הוצעו הרבה שיטות לחישוב מרחק בין שתי רשימות כאלה, למשל: מרחק מינקובסקי, מרחק צ'בישב ומרחק המינג. תוכלו למצוא תיאורים של שיטות אלה ברשת. אנחנו נבחר במרחק איקלידי (Euclidean distance). המרחק האיקלידי בין שתי רשימות באורך n, X ו-Y, הוא:

$$\sqrt{(x_0-y_0)^2+(x_1-y_1)^2+\cdots+(x_{n-1}-y_{n-1})^2}$$

הפונקציה distance מקבלת שתי רשימות, a ו-b, ומחזירה את המרחק האיקלידי ביניהן. עליכן להשלים את הקוד של הפונקציה distance.

כעת תוכלו להשלים את הקוד של הפונקציה nearest_neighbour. הפונקציה מקבלת את ה-dataset, ds, ואת התמונה instance, ומחזירה את הספרה שבתמונה, לפי הערכת אלגוריתם השכן הקרוב.

רמז: הפונקציה צריכה לעבור, בלולאה, על כל השורות ב-ds, ולמצוא את השורה שמרחקה מ-instance מינימלי. הפונקציה תחזיר את הספרה המצולמת בתמונה זו.

האם האלגוריתם מצליח לגלות את הספרה שבתמונה?

האם אלגוריתם השכן הקרוב מסווג נכון תמונות של ספרות? כדי לבדוק זאת נכתוב פונקציה שתסווג תמונות מתוך ה-dataset של MNIST, שסיווגן ידוע, ונשווה את הסיווג של האלגוריתם לסיווג האמתי.

הפונקציה test מקבלת dataset, ds, ושני אינדקסים train, ו-test (ההנחה היא ש-test+train לא גדול ממספר השורות ב-ds). לכל אחד מ-test השורות האחרונות ב-ds, הפונקציה תקרא לפונקציה nearest_neighbour שתעריך איזו תמונה בשורה זו. ה-dataset שיועבר ל-nearest_neighbour יכיל רק את train השורות הראשונות ב-ds (למה לחלק את ה-dataset לשני חלקים: train השורות הראשונות ו-test השורות האחרונות?). הפונקציה תחשב, ותחזיר, את היחס בין מספר התמונות שסווגו נכון לבין מספר התמונות שסווגו (test).

עליכן לכתוב את הלולאה שקוראת ל-nearest_neighbour, וצוברת, ב-count, את מספר התמונות שסווגו נכון. הריצו את הפונקציה ובדקו את איכות הסיווג של האלגוריתם.

סיווג תמונות של ספרות שאתן צילמתן

כפי שראיתן, האלגוריתם מסווג מצוין תמונות מתוך ה-dataset של MNIST. מה לגבי תמונות של ספרות שתציירנה בעצמכן?

צורנה תמונות (מסוג jpg) של ספרות שכתבתן בעצמכן ואחסנו את כולן בתיקייה אחת. השתדלנה שהתמונות תהיינה ריבועיות (ולא מלבניות), שהספרות תצוירנה בטוש שחור על נייר לבן, ושהספרות תתפרסנה על כל שטח הנייר (בדומה לספרות ב-dataset).

נכתוב פונקציה בשם img2data המקבלת שם של קובץ של תמונה (ומשתנה neg, שתפקידו יוסבר בהמשך) ומחזירה רשימה של ערכי 784 הפיקסלים שבתמונה. הפונקציה משתמשת במחלקה Image. הקוד פשוט ומתועד. מידע נוסף על הפונקציות במחלקה Image תוכלו למצוא ברשת.

עליכן להוסיף לפונקציה בדיקה האם הערך של הפרמטר neg הוא True, ואם כן להפוך את im לנגטיב של עצמה (למה זה חשוב? הזכרנה איך נראות התמונות ב-dataset של MNIST). בדקו את הפונקציה על התמונות שציירתן – הפכו את התמונה לרשימה, הפכו את הרשימה חזרה לתמונה (כתבתן פונקציה שעושה זאת), והציגו את התמונה שהתקבלה.

לסיום נתונה הפונקציה classify_images המקבלת dataset, ds, ושם הנתיב (path) של התיקייה בה נתונות התמונות. הפונקציה קוראת את כל קבצי ה-jpg בתיקייה, ולכל קובץ

מדפיסה את שמו ואת הספרה שמוצגת בו (לפי הערכת אלגוריתם השכן הקרוב).הפונקציה פשוטה למדי ומתועדת. נציין רק, שהפונקציה listdir היא חלק מהחבילה os.

מהם ביצועי האלגוריתם על הספרות שציירתן?

כיצד ניתן לשפר את ביצועי האלגוריתם?

כפי שראיתן, ביצועי האלגוריתם על התמונות שציירתן אינם מצוינים. ניתן לשפרו במספר דרכים שתוכלנה לחקור וליישם בעצמכן:

- האלגוריתם קובע את סיווג התמונה לפי השכן הקרוב. יכול להיות שהביצועים ישתפר עם האלגוריתם ימצא מספר שכנים קרובים, ויקבע על פי רובם את הסיווג. אלגוריתם זה נקרא k nearest neighbour או knn בקיצור. תוכלו למצוא עליו מידע רב ברשת. כדי ליישם אותו עליכן למצוא דרך יעילה לדעת מיהם k השכנים הקרובים.
- האלגוריתם מחשב מרחק איקלידי בין תמונות. אולי דרך אחרת לחישוב מרחק תביא לסיווג מדויק יותר?
- ה-dataset מכיל 10000 דוגמאות. האם צריך את כולן? נסו לבדוק את רמת הדיוק של האלגוריתם כשמשתמשים רק ב-1000, 2000, 3000 או 4000 דוגמאות. האם רמת הדיוק של האלגוריתם נפגעת? שימוש ב-dataset קטן יותר יוביל לאלגוריתם מהיר יותר.
- בכל תמונה יש 784 פיקסלים. האם כולם שימושיים לסיווג? הגיוני לחשוב שהשורה העליונה והשורה התחתונה בתמונה לא מכילות הרבה מידע. למעשה, ניתן להראות, שניתן להגיע לרמת דיוק גבוהה של סיווג בעזרת חמישית מהפיקסלים בלבד. מספר קטן של פיקסלים יוביל לאלגוריתם מהיר יותר.
- מדוע להשתמש בתמונות בגודל של 28 על 28 פיקסלים. אולי מספיק להשתמש ברזולוציה נמוכה יותר? למשל, בתמונות בגודל 14 על 14? מספר קטן של פיקסלים יוביל לאלגוריתם מהיר יותר.