מיני פרוייקט באנליזה של נתונים.202.1.4511
ד"ר סיון סבתו sabatos@cs.bgu.ac.il
סתיו 2017/8

בפרוייקט זה נממש אלגוריתם הלומד לזהות ספרות כתובות בכתב יד על פי תמונות. ניתן לבצע את הפרוייקט בזוגות או ביחידים.

אלגוריתם לומד

האלגוריתם הלומד מקבל כקלט **דוגמאות examples** של תמונות, כאשר לכל תמונה מוצמדת הספרה האמיתית הכתובה בה. זוהי ה **תוית label** של הדוגמא. לקלט של האלגוריתם, שהוא קבוצת דוגמאות שלכל אחת מוצמדת התוית האמיתית שלה, קוראים **מדגם אימון training set**.
האלגוריתם משתמש במדגם האימון כדי לבנות **מודל חיזוי predictor**. מודל חיזוי הוא פונקציה הממפה כל דוגמא אפשרית לתוית המתאימה לה לדעת האלגוריתם. המטרה היא שהמודל יהיה מוצלח, כלומר שבקבלת דוגמא חדשה, המודל בדרך כלל יתן את התוית המתאימה ביותר לדוגמא. בדרך כלל אי אפשר להגיע ל 100% הצלחה, אך מקווים לאחוזי הצלחה כמה שיותר גבוהים.
כדי למדוד את אחוזי ההצלחה של האלגוריתם, נשתמש **במדגם בדיקה test set.** מדגם זה מכיל דוגמאות שהתיוג שלהם ידוע לנו אך אינו ידוע לאלגוריתם. אלגוריתם החיזוי יקבל כקלט את מדגם הבדיקה, ויציע תוית לכל אחת מהדוגמאות במדגם, על פי מודל החיזוי שבנה אלגוריתם הלמידה. אחוז ההצלחה של האלגוריתם הוא מספר הדוגמאות במדגם הבדיקה שעבורן התויות שהציע היא אכן התוית הנכונה.

האלגוריתם בוחר את המודל שנראה לו המתאים ביותר מתוך קבוצה של מודלים אפשריים (בפרוייקט זה נשתמש בקבוצת המודלים שנקראת **עצי החלטה**, המתוארים בהמשך). כדי להחליט איזה מודל הוא המתאים ביותר, האלגוריתם משתמש במדגם האימון: העקרון הבסיסי הוא שאם מדגם האימון הוא קבוצה מייצגת של דוגמאות, אז מודל שעובד טוב על מדגם האימון (כלומר חוזה נכון הרבה מהתויות של דוגמאות במדגם זה) יצליח לחזות היטב גם תוית של דוגמאות אחרות שמעולם לא ראה.

חשוב להזהר עם עקרון זה, כיון שאם מודל החיזוי מתייחס באופן ספציפי מדי למדגם האימון, הוא עלול להיות גרוע מאד על דוגמאות חדשות למרות שהוא מצליח מאד על מדגם האימון. דוגמא: מודל חיזוי שלוקח את רשימת הדוגמאות ממדגם האימון ופשוט זוכר את התויות המתאימה לכל דוגמא שראה, יצליח מאד על מדגם האימון אך ייכשל כמעט לחלוטין על דוגמאות חדשות שמעולם לא ראה. לתופעה זאת קוראים **התאמת יתר over-fitting.** אנחנו נטפל גם בבבעיה זו במימוש האלגוריתם.

קבוצת מודלי החיזוי: עצי החלטה decision trees

האלגוריתם הלומד שתממשו ילמד מודל חיזוי מסוג עץ החלטה. מודל חיזוי כזה הוא עץ בינארי, אשר בכל צומת פנימי שלו יש תנאי, ובכל אחד מהעלים שלו יש אחד מהערכים האפשריים לתויות. לכל צומת יש שני ילדים: אחד עבור המקרה שהתנאי מתקיים ואחד עבור המקרה שהתנאי אינו מתקיים.
כדי לסווג דוגמא על פי עץ החלטה נתון, מתחילים משורש העץ ובודקים את התנאי על התמונה. לפי קיום או אי קיום התנאי, ממשיכים לילד המתאים. אם גם בצומת זה מופיע תנאי, בודקים גם אותו, וכן הלאה עד שמגיעים לעלה. כאשר מגיעים לעלה, מחזירים את התוית הכתובה בו. זו התחזית של המודל עבור הדוגמא.
פרט חשוב במימוש של אלגוריתם למידה של עץ החלטה הוא קביעת קבוצת התנאים שמתוכן האלגוריתם יבחר את אלו שכדאי להשתמש בהם בעץ. נסמן קבוצה זו באות C. כל תנאי בקבוצה מחזיר "כן" או "לא" לפי התמונה שהתקבלה כקלט. למשל, אם התנאי הוא "האם הפיקסל בפינה הימנית העליונה הוא בעוצמה גבוהה מ 230", התנאי יחזיר "כן" על פי ערך הפיקסל בתמונה אותה העץ מנסה לסווג כעת.

כיצד האלגוריתם בוחר עץ החלטה

האלגוריתם ישתמש בדוגמאות האימון על מנת לבנות עץ החלטה באופן הבא:
מתחילים מעץ שעשוי מעלה אחד. בעלה זה תופיע הספרה הנפוצה ביותר במדגם האימון. כעת, עובדים בסיבובים: בכל סיבוב מוחקים את אחד העלים בעץ, ומחליפים אותו בצומת שלו יש שני ילדים שהם עלים. הערך בכל עלה חדש תמיד יהיה הספרה הנפוצה ביותר במדגם האימון עבור תמונות שמתאימות לעלה זה. מספר הסיבובים (כלומר, מספר הצמתים הפנימיים שיהיו לעץ) ייקבע לפי פרמטר קלט T.

כדי להחליט איזה עלה להחליף ומה לשים במקומו, נשתמש במדד שנקרא **שיפור האינפורמציה information gain.** מדד זה מודד את השינוי ב**אנתרופיה entropy** של העץ. אנתרופיה היא מספר המודד את מידת אי הוודאות בשיוך דוגמא לתוית על פי העץ הנוכחי.

לכל עלה בעץ, אפשר לחשב את האנתרופיה entropy של העלה על מדגם האימון כך:
נסמן ב N(L) את מספר הדוגמאות במדגם האימון שמתאימות לעלה זה (כלומר מתאימות לתנאים במסלול שמוביל אל העלה). מתוך הדוגמאות האלו, נסמן ב Ni(L) את מספר הדוגמאות שהתוית עליהן היא i.
האנתרופיה של העלה L על מדגם האימון, כאשר K הוא מספר התויות, היא:

$$H(L) := \sum_{i=1}^{K} \frac{N_i(L)}{N(L)} \log(N(L)/N_i(L))$$

אם כל הדוגמאות בעלה הן בעלות אותה תוית, נקבל H(L)=0, כלומר אין כלל אי וודאות. לעומת זאת, אם הדוגמאות בעלה מחולקות שווה בשווה בין כל התויות, נקבל H(L)=log(K). זה הערך המקסימלי של אי הודאות האפשרי.
האנתרופיה הכוללת של העץ היא סכום משוקלל של האנתרופיה בכל העלים. מטרתנו בבנית העץ היא להקטין את האנתרופיה של הסיווג. לכן בכל סיבוב נגדיל את העץ באופן שיקטין את האנתרופיה הכוללת כמה שיותר, באמצעות החישוב הבא.

נניח שמחליפים עלה L בתנאי כלשהו מתוך הקבוצה C. לצומת התנאי יווצרו שני עלים ילדים, עבור שתי התוצאות האפשריות של התנאי. נסמן את העלים האלו ב La, Lb. מתוך N דוגמאות במדגם האימון שמתאימות ל L, נסמן ב N(La) את הדוגמאות שמתאימות לעלה החדש וב N(Lb) את הדוגמאות שמתאימות לעלה החדש Lb.
האנתרופיה המשוקללת של שני העלים החדשים ביחד היא $H(X) = \frac{N(L_a)}{N(L)} H(L_a) + \frac{N(L_b)}{N(L)} H(L_b)$.
היא תמיד תהיה קטנה יותר מ H(L). שיפור האינפורמציה של תנאי X על עלה L הוא:

$$IG(X, L) := H(L) - H(X).$$

בכל סיבוב האלגוריתם יבחר את X ואת L שעבורם N(L)*IG(X,L) הוא מקסימלי, כאשר X נבחר מתוך כל התנאים בקבוצה C ו L נבחר מתוך כל העלים הנוכחיים בעץ.
שימו לב שאת השיפור מכפילים במספר הדוגמאות N שמתאימות לעלה L, כדי שהאלגוריתם יעדיף שיפור שמשפיע על דוגמאות רבות ולא כזה שמשפיע על מעטות.

אחרי T סיבובים האלגוריתם עוצר כאשר בידיו עץ החלטה עם T צמתים פנימיים.

כיצד לבחור את גודל העץ (ערך הפרמטר T)

גודל העץ הוא פרמטר חשוב עבור האלגוריתם הלומד. גודל עץ קטן מדי, יוביל לכך שיהיה קשה לזהות לאיזה תוית הדוגמא מתאימה, כיוון שלא נבדקו מספיק תנאים שיכולים להשפיע על הבחירה. גודל עץ גדול מדי, יכול להוביל לכך שהעץ מאד מתאים לדוגמאות שבמדגם האימון, אבל לא מתאים במיוחד לדוגמאות

חדשות. זאת בשל התופעה של התאמת יתר over-fitting.

אנו נשתמש ב**מדגם ולידציה validation set** לבחירת הערך של T:
קובעים ערכים אפשריים ל T. לצורך המימוש שלנו נבדוק רק ערכי T שהם חזקות של 2, מ-1 ועד 2 בחזקת L, כאשר L הוא פרמטר קלט (למשל אם L = 3 ייבדקו הגדלים 1,2,4,8). כמו כן קובעים אחוז מדוגמאות הקלט שישמשו למדגם הולידציה, נסמנו P.
מתוך הקלט לאלגוריתם בוחרים באופן רנדומלי P% מהדוגמאות עם התויות שלהן. דוגמאות אלו יהיו מדגם הולידציה, ו**לא יהיו חלק ממדגם האימון**. שאר הדוגמאות שהתקבלו כקלט ישמשו כמדגם אימון.
עבור כל אחד מהערכים האפשריים של T, מריצים את אלגוריתם בניית העץ על מדגם האימון עבור T.
מחשבים את הצלחת האלגוריתם המשוערת על ידי הפעלת מודל החיזוי על הדוגמאות שבמדגם הולידציה, וחישוב אחוז הדוגמאות שעבורן מודל החיזוי של האלגוריתם מתאים לתוית האמיתית של הדוגמא.

בסוף המעבר על כל הערכים של T, בוחרים את הערך של T שעבורו ההצלחה על מדגם הולידציה היתה הגבוהה ביותר. העץ שהאלגוריתם למד עבור ערך זה יהיה מודל החיזוי הנבחר. הצלחתו הסופית של אלגוריתם הלמידה תיבדק על מדגם הבדיקה.

פורמט ה data-set

אנו נריץ את האלגוריתם על data-set בשם MNIST. ניתן לקרוא עליו כאן:
http://yann.lecun.com/exdb/mnist

MNIST מכיל תמונות של ספרות בכתב יד בגודל 28x28, כאשר כל פיקסל בתמונה מיוצג על ידי ערך בין 0 ל 255 הקובע את דרגת האפור שלו. לכן כל תמונה מיוצגת על ידי וקטור באורך 748 של מספרים בין 0 ל 255. הקלט לאלגוריתם יהיה קובץ המתאר מדגם אימון, כלומר תמונות ולכל תמונה מה הספרה המופיעה בה. אנו נשתמש ב data-set בפורמט CSV המתואר כאן:

https://pjreddie.com/projects/mnist-in-csv/

ניתן להוריד את מדגם האימון ואת מדגם הבדיקה המלאים בפורמט הזה של MNIST מאותו לינק.
אלגוריתם הלמידה יקבל מדגם אימון בפורמט המתואר (לא בהכרח מאותו data-set או באותו גודל), וייצור קובץ המתאר את העץ שלמד. לאחר מכן אלגוריתם החיזוי יקבל את הקובץ המתאר את העץ ומדגם בדיקה באותו פורמט (במדגם הבדיקה יש להתעלם מהתוית המופיעה לצד כל תמונה) ויחזיר את רשימת התויות החזויות עבור הדוגמאות במדגם הבדיקה על פי העץ שבקובץ.

למידת עץ ההחלטה על data-set של תמונות מתוך MNIST

כדי לממש אלגוריתם למידת עצים על צדגם מתוך MNIST, צריך לקבוע את קבוצת התנאים C שממנה האלגוריתם יכול לבחור. בפרויקט תממשו שתי גרסאות של אלגוריתם הלמידה, שההבדל ביניהן הוא בקבוצת התנאים C.

1. **גרסא ראשונה:** קבוצת התנאים C תכיל את כל התנאים מהצורה: האם ערך הפיקסל במיקום (x,y) בתמונה הוא יותר מ 128?
2. **גרסא שניה:** בגרסא זו עליכם **להציע קבוצת תנאים C משלכם** (שיכולה לכלול גם את התנאים הקודמים לפי שיקול דעתכם). המטרה היא להציע קבוצת תנאים שישפרו את יכולת החיזוי של העץ הנלמד עד כמה שניתן. שימו לב: יש להציע תנאים כלליים מספיק כדי שיתאימו לסוגים שונים של ספרות ואותיות. למשל, אין להציע תנאי שעוזר באופן ספציפי להבדלה בין הספרות 0 ו-3. התנאים יכולים להיות פשוטים או מורכבים, על פי החלטתכם, אך כדאי שיהיה הגיון מאחורי בחירתם.

דוח הפרוייקט

בדו"ח הפרוייקט ענו על השאלות הבאות.

1. תארו את קבוצת התנאים שהשתמשת בהם בגרסא השניה של המימוש. הסבירו מדוע בחרתם אותם וכיצד הם מומשו. אם ניסיתם מספר אפשרויות שונות, פרטו לגביהן והסבירו כיצד בחרתם את האפשרות שהגשתם.
2. פרטו את אופן המימוש שלכם של שתי הגרסאות. הסבירו כיצד בחרתם את מבני הנתונים והאלגוריתמים שבהם השתמשתם למימוש. פרטו לגבי אתגרים שנתקלתם בהם וכיצד התמודדתם איתם.
3. הריצו ניסוי שבו אתם משווים את תוצאות ההרצה של שתי גרסאות האלגוריתם. בניסוי עליכם לבדוק את הדיוק של האלגוריתם על מדגם הבדיקה המלא של MNIST כתלות בגודל מדגם האימון. עבור גודל נתון של מדגם האימון, עליכם לייצר מדגם בגודל זה על ידי בחירה אקראית של דוגמאות מתוך מדגם האימון המלא של MNIST.
   a) הציגו בדו"ח גרף המראה את השגיאה על מדגם הבדיקה של שתי הגרסאות של האלגוריתם, כתלות בגודל המדגם. ודאו כי אתם בודקים גדלים בתחום רחב החל מעשרות דוגמאות, ועד כל מדגם האימון של MNIST. אם התוצאות משתנות באופן משמעותי מניסוי לניסוי על אותו גודל מדגם, הריצו מספר פעמים את אותו גודל עם מדגמים אקראיים שונים, ודווחו בגרף על ממוצע התוצאות ועל הטווח שלהם (על ידי error bars). פרטו כמה פעמים הרצתם כל גודל של המדגם.
   b) תארו את ההבדלים בין התוצאות של שתי הגרסאות של האלגוריתמים כפי שהן משתקפות בגרף מהסעיף הקודם, ואת התלות שלהם בגודל המדגם.
   c) נסו להסביר מדוע קיבלתם את ההבדלים הללו ואת התלות שלהם בגודל המדגם.
4. העתיקו את מדגם האימון ומדגם הבדיקה של MNIST, והוסיפו להם רעש באופן הבא: עבור רמת רעש alpha בין 0 ל-1, כל פיקסל בכל תמונה יתהפך (כלומר המספר יהפוך למשלים שלו ל 255) בסיכוי alpha. הריצו ניסוי דומה לזה שהרצתם בסעיף 3, אך הפעם על המדגמים הרועשים. בחרו מספר רמות של רעש שבהן ניתן לראות הבדל משמעותי בתוצאות.
   a) הציגו בדו"ח גרף המראה את תוצאות הניסויים כתלות בגודל המדגם, עבור שתי הגרסאות וכל רמות הרעש שבדקתם.
   b) תארו את ההבדלים בהתנהגות הגרסאות, כתלות ברמות הרעש ובגודל המדגם.
   c) נסו להסביר את ההבדלים הללו וכיצד רמת הרעש משפיעה עליהם.
5. עבור הניסויים שביצעתם בסעיפים 3, 4, בדקו את גדלי העצים שקיבלתם בכל ניסוי.
   a) דווחו על גדלי העצים כתלות ברמת הרעש וגודל המדגם בגרף/ים מתאימים.
   b) תארו את התצפיות המרכזיות שניתן להבחין בהם בגרפים שהצגתם
   c) נסו להסביר את התצפיות.

הרצה והגשה

יש להגיש את הקוד של האלגוריתם, ודו"ח פרוייקט. נא לעקוב אחר ההוראות להלן באופן מדויק. חריגה מההוראות תגרום להורדה בניקוד.

Command line for running the algorithms

You can write your code in any (reasonable) language, e.g., Python, Java, Matlab, C++. If you want to use a different language email me to make sure it's OK.

Your submission should include the code as well as an executable (or a script, e.g. can be a script that runs Java with the jar file), for the learning algorithm that produces the tree, and for the prediction algorithm that predicts labels based on the learned tree.

The learning algorithm

The command line to run the learning algorithm should be in the following format:

```
learntree <1/2> <P> <L> <trainingset_filename> <outputtree_filename>
```

Note: this command line *exactly* should work, without change. If your code has, e.g. an extension py, write a script that is called learntree (without extensions) which runs it. The input parameters are:

- 1/2 - which version of the learning algorithm to run. 1 - the first version with the given conditions, 2 - the second version, with your new conditions.
- P - the percent of the training set that should be used for validation
- L - the maximal power of 2 for tree sizes T
- `trainingset_filename` - the name/path of the training set file. The format of this file is the CSV format mentioned above. Each line represents a single example. The file can have any number of lines depending on the size of the input training set. The examples in the file can be from the MNIST dataset but can also be others (in the same format).
- `outputtree_filename` - the name/path of the file into which the algorithm will output the decision tree that it learned based on the training set. The format of the file is up to you, since only your prediction algorithm will read it. If the file exists you should delete it and write a new file instead of it.

You should identify any errors related to the input parameters such as files that don't open, etc. If there are errors, report them via standard error and exit.

The output of "learntree" is the tree file. In addition, you should print to the standard output the following information:

```
num: <number of training examples>
error: <error of the learned tree on the training set (in whole percents,
between 0 and 100, no % sign)>
size: <size of the learned tree>
```

For example, the output can look like this:

```
num: 1200
error: 34
size: 32
```

There should be **no other prints, and no format changes**. A new line must follow the number in each line.

The prediction algorithm

The command line to run the prediction algorithm should be in the following format:

```
predict <tree_filename> <testset_filename>
```

Note: this command line *exactly* should work, without change. If your code has, e.g. an extension py, write a script that is called predict (without extensions) which runs it.

The input parameters are:

- `tree_filename` - the name/path of the file that describes the tree to use for prediction. This will be a file that was output by running you "learntree", with the first version or the second version of the algorithm.
- `testset_filename` - the name/path of the file that includes the test examples which you should predict, in the CSV format mentioned above. Note that the format also includes a label for each example, you should ignore it.

You should identify any errors related to the input parameters such as files that don't open, etc. If there are errors, report them via standard error and exit.

The output of "predict" is a list of labels which should be printed to the standard output, each label in its own line and ending with a newline. **No other characters/spaces/prints are allowed**. For instance, if the test file has 3 examples, the output of "predict" can be:

2
1
1

Meaning that it predicts that the label of the first test example is 2, and the label of the other examples is 1.

Submission instructions

You should submit a zip file named after your id number, e.g. 012345678_123456789.zip. Include the following files in the root of the submission archive file. **No changes to the file names are allowed, that includes extensions.**

- `learntree` - the exectuable or script that runs the learning algorithm.
- `predict` - the exectuable or script that runs the prediction algorithm.
- `report.pdf` - the project report

- your source code - can be in a subfolder

All files/folders should have alphanumeric names: no spaces, no hebrew letters.

Running your code

- You code will be executed on **vmcicero3**, possibly several times.
- Make sure your code, with the command lines described above, works on this machine after unzipping the submission file, and from the same location as the zip file. Better to try unzipping and running it in someone else's account to make sure there are no account dependencies.
- If you need a package that is not installed on vmcicero3, please email me. **Note: it could take a while to install packages, so don't do this last minute before the deadline!**
- Do not create files during the run of your algorithm except for those required by the instructions above.