אופיר בן שוהם

למידה חישובית - פרויקט סיום

בחרתי את המאמר:

A two-stage gene selection scheme utilizing MRMR filter and GA wrapper

Ali El Akadi · Aouatif Amine · Abdeljalil El Ouardighi · Driss Aboutajdine

תוכן עניינים:

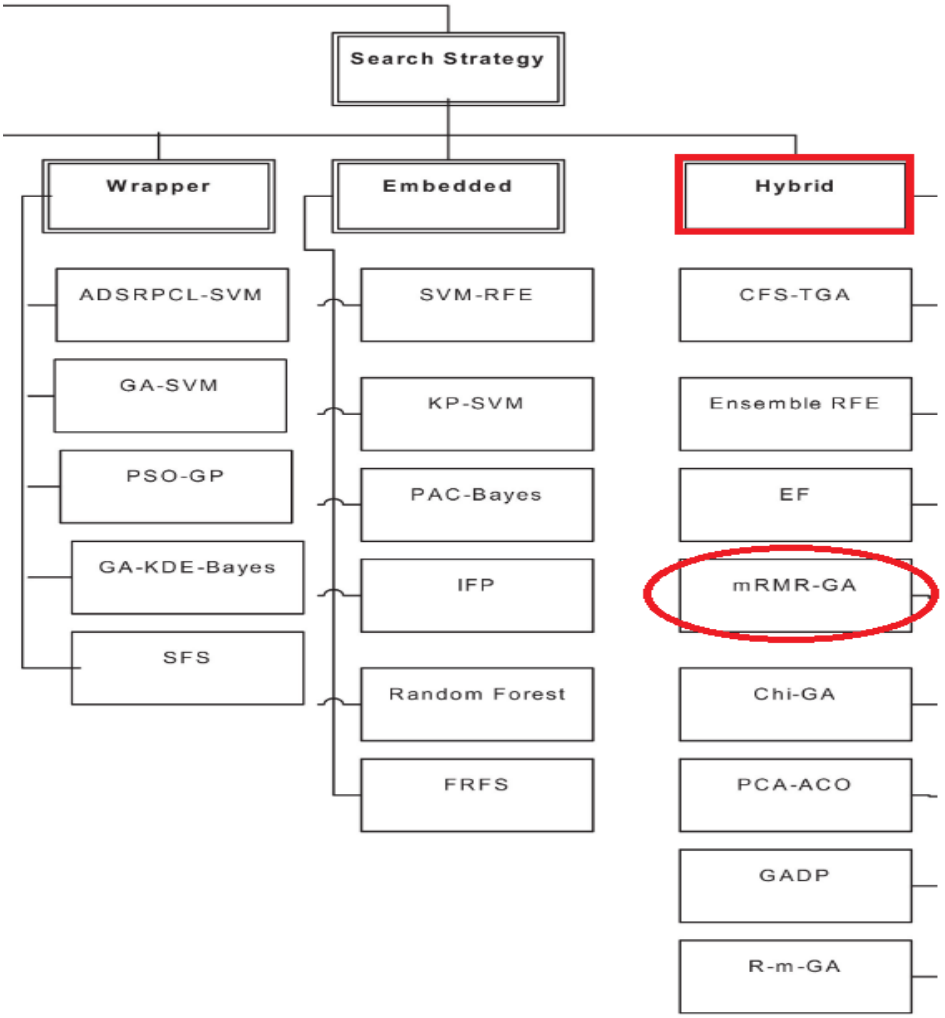
מטרת האלגוריתם

מטרת האלגוריתם היא לבצע feature selection על מנת להוריד את מספר הפיצ'רים הנדרשים לייצוג, תוך פגיעה מינימלית במדדי המודל. האלגוריתם משלב MRMR בתור פילטור ראשוני ולאחר מכן מפעיל אלגוריתם גנטי, כך שה-fitness function שלו היא estimator בפני עצמו, כפי שאציג בפירוט בהמשך.

תיאור המאמר

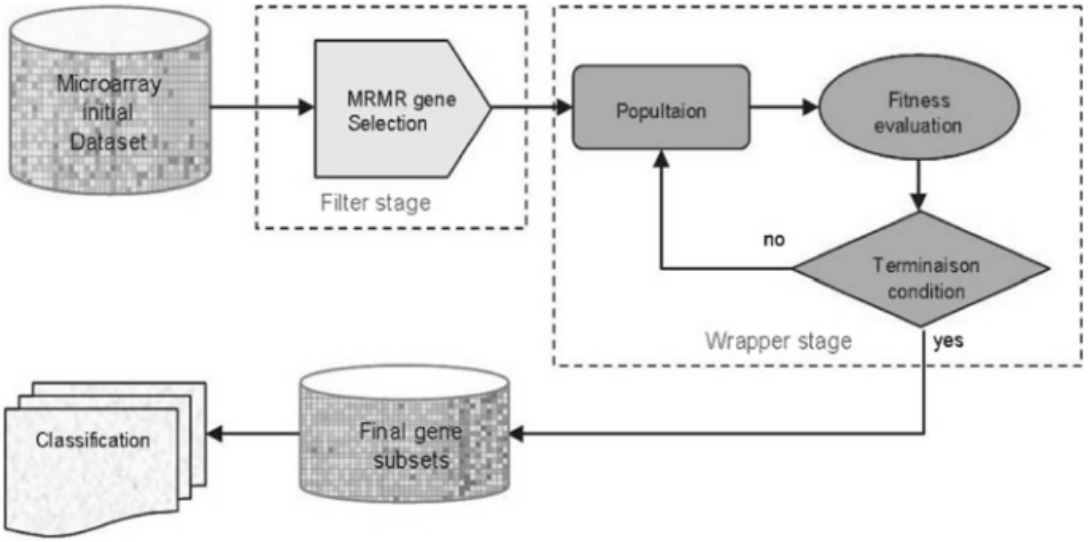
מבחינת טקסטונומיה, המאמר משתייך לקבוצת ה- hybrid methods כי הוא משלב MRMR בתור פילטר ועל זה אלגוריתם גנטי. ה-MRMR מסיר פיצרים לא רלוונטיים ואז האלגוריתם הגנטי פועל אחריו ובוחר את הפיצרים הכי חשובים.

ההיררכיה שלו מסומנת באדום:



למעשה במאמר יש שיטת hybrid שמשלבת filter שהוא MRMR ו-Wrapper שהוא GA SVM.

ארכיטקטורת האלגוריתם



פאסודו קוד:

1. בצע preprocessing.
2. הרץ אלגוריתם MRMR על הדאטא.
3. הרץ אלגוריתם גנטי עם SVM/NB בתור ה-fitness function עד אשר הגיע לתנאי עצירה (התכנסות או מקסימום חזרות).
4. החזר את קבוצת הגנים שחזרה מ3.

תיאור ופירוט על האלגוריתם:

למעשה, במאמר הם הציעו אלגוריתם עם שני שלבים אחד אחרי השני. השלב הראשון הוא ביצוע MRMR filter שבוחר את הגנים שיש להם את הרלוונטיות הגבוהה ביותר למחלקת ה-target וגם שיש להם את ההבדל המקסימלי אחד מהשני (כי אין לנו מה לקחת גנים דומים אחד לשני - לא נרוויח ככה עוד מידע).
לאחר מכן מפעילים GA (אלגוריתם גנטי), שמשלב בתוכו אלגוריתם למידה ומשמש בתור wrapper - די מוצלח אבל יקר מבחינת זמן ריצה. בעצם הMRMR מסיר מידע לא רלוונטי ועובד כפי שפירטתי מקודם, ואז מוצא את הקבוצת מועמדים האופציונאלית. ואז GA בוחר את הקבוצה האופטימלית מהרשימת מועמדים האופציונאלית.

מבחינת ה-preprocessing - לכל גן יש zero mean value and unit variance והם המירו את הנתונים הרציפים לקבוצה סופית של מרווחים, כדי להפחית רעשים וגם NB מעדיף דאטא קטגוריאלי.

פרמטרים:

1. האלגוריתם MRMR עוצר אחרי שהוא בוחר 100 גנים.
2. משתמשים ב-Leave One out cross validation על קבוצת גנים כדי להתגבר על overfitting. כלומר כל פעם מאמנים על n-1 דוגמאות ובוחנים על זו ששמנו בחוץ. עושים את זה עבור N דוגמאות שיש לנו ומקבלים N פעמים תוצאת מדדים (כל מדד מחושב על בסיס LOOCV).

3. לאחר מכן עבור כל דוגמת אימון הערך fitnetss של הGA מחושב ע"י ה-LOOCV על ה-classifier (כמו NB או SVM במאמר, ה-NB עם ביצועים טובים יותר מSVM כאן).

פרמטרים לאימון האלגוריתם הגנטי:

GA parameters

Parameters	Value
Size of population	100
Number of generations	20
Crossover rate	0.8
Mutation rate	0.1

ה-SVM הוגדר עם C=100 בתור normalization parameter. ו-poly kernel עם d=degree=1.

יתרונות האלגוריתם שהוצע במאמר:

1. נותן תוצאות טובות יותר (במדד accuracy) מאלגוריתם גנטי בנפרד, וגם תוצאות טובות יותר מאלגוריתם MRMR בנפרד.
2. מאפשר feature selection תוך הקטנת מספר המשתנים.
3. אלגוריתם ה-MRMR שמשתמשים בו כאן, מסיר רעשים וכאמור בוחר את הגנים שיש להם את הרלוונטיות הגבוהה ביותר למחלקת ה-target וגם שיש להם את ההבדל המקסימלי אחד מהשני (כי אין לנו מה לקחת גנים דומים אחד לשני - לא נרוויח ככה עוד מידע).

חסרונות האלגוריתם:

1. דורש הרצת שני אלגוריתמים זה אחר זה. פחות יעיל מבחינת זמן ריצה מאשר הרצה של אחד מהם.
2. בהמשך ל1, האלגוריתם משתמש ב-NB בתור fitness function, דבר שמוסיף זמן ריצה לאלגריתם.
3. דורש יותר זמן פיתוח מאשר כל אחד מהאלגוריתמים האלו בנפרד.

לא ראיתי שפורסם הקוד של האלגוריתם המתואר (מה גם שהקוד שלא פורסם הוא ב-matlab) במאמר אז מימשתי אותו בעצמי. ניתן לראות את הקוד ב-github המצורף, בפונקציה paper_algo.
להלן פירוט 3 הרצות על toy example dataset:
כפי שניתן לראות במחברת, הגדרתי את האלגוריתם והרצתי אותו 3 פעמים עבור ה-toy dataset. פעם אחת עבור k=1, פעם שנייה עם k=2 ופעם שלישית עם k=3. המשמעות של k היא מספר הפיצרים שהאלגוריתם יבחר. ז"א שאם k=3, מתוך כל הפיצרים האלגוריתם יבחר 3 פיצרים לכל דוגמה.

תוצאות עבור toy dataset:
עבור k=1 האלגוריתם בחר את הפיצר ה- 26. עבור k=2 האלגוריתם בחר את פיצר 26 ואת פיצר 30. עבור k=5 האלגוריתם בחר 3 פיצרים: 26, 30, 44.

איסוף הנתונים:

דאטאסטים שנבחרו:

מתוך Datamicroarray:

1. gravier
2. yeoh
3. singh
4. christensen
5. sorlie

מתוך scikit-feature:

1. warpAR10P
2. orlraws10P
3. Carcinom
4. arcene
5. Yale

מתוך mAML_benchmark_datasets:

1. Morgan2012_IBD
2. Gevers2014_IBD_ileum
3. Costello2009_Subject
4. Ravel2011_Vaginal
5. Gevers2014_IBD_rectum

מתוך microbiome_data:

1. david
2. claesson
3. kostic
4. bushman_cafe
5. turnbaugh

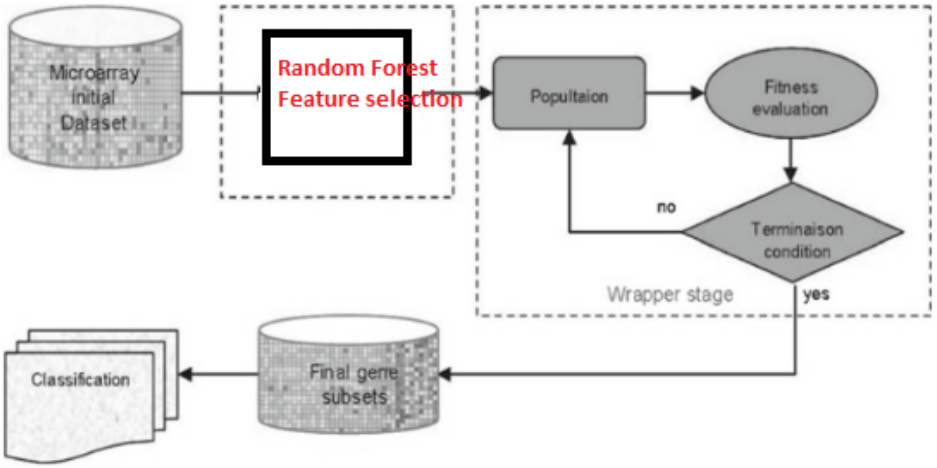
מבחינת preprocessing, ניתן לראות במחברת הזו את ה-preprocessing שבוצע. לאור שיקולי זמן ריצה ומשאבי אימון, צמצמתי את מספר הפיצרים בדאטאסטים המשתייכים לקבוצה microbiome_data.

עבור כל הדאטאסטים, ה-preprocessing הכיל את השלבים הבאים:

1. ביצוע השלמה למידע חסר ע"י SimpleImputer על בסיס mean strategy.
2. ביצוע Variance Threshold עם שימוש ב-threshold הדיפולטי שערכו 0.
3. ביצוע Power Transformer עם ערכים דיפולטיים של sklearn.
4. מספור הפיצ'רים על בסיס סדרם.
5. שינוי שם העמודה שמכילה את ה-class לשם y.

תיאור האלגוריתם המשופר

ארכיטקטורת האלגוריתם



למעשה, הסרתי את ה-feature selection על בסיס mrmr והוספתי feature selection על בסיס אימון Random Forest והסתכלות על ה-feature importance של הפיצרים בעץ שנבנה. לקחתי את ה-200 פיצרים שעבורם יש feature importance הכי טוב, והעברתי אותם להמשך הכולל את ה-Genetic Algorithm שמשתמש ב-Naive Bayes בתור ה-fitness function שלו באימון.
בהמשך אציג את התוצאות של האלגוריתם המשופר, ואציג בפרט את ההבדלים בינו לבין האלגוריתם שמימשתי מהמאמר.

תוצאות האלגוריתם והשוואה לאלגוריתמים דומים:

<u>פאסודו קוד עבור הוצאת המדדים:</u>

- עבור על כל הדאטאסטים:
  - עבור על כל הפולדים:
    - עבור על כל ה-fs methods:
    - עבור על כל ה-K-ים:
      - סנן את הדאטא של הפולד ע"פ ה-fs, k.
      - עבור כל ה-classifiers:
        - אמן מודל, בדוק מדדים וזמנים - ע"י cross_validate.
        - שמור תוצאות.

להלן התוצאות:
מבחן פרידמן - השערת האפס נדחתה. ההפרשים במדדי ה-AUC מובהקים סטטיסטית, יצא p-val ששואף לאפס.

תוצאות מבחן Post-Hoc-nemenyi:

names	f_classif	paper_algo	paper_algo_improved	relieff	mrmr	rfe
f_classif	1.000	0.001000	0.001	0.900	0.001	0.001000
paper_algo	0.001	1.000000	0.001	0.001	0.001	0.897369
paper_algo_improved	0.001	0.001000	1.000	0.001	0.001	0.001000
relieff	0.900	0.001000	0.001	1.000	0.001	0.001000
mrmr	0.001	0.001000	0.001	0.001	1.000	0.001000
rfe	0.001	0.897369	0.001	0.001	0.001	1.000000

ניתוח התוצאות:

האלגוריתם paper_algo הוא האלגוריתם של המאמר אשר מימשתי את הקוד שלו. האלגוריתם paper_algo_improved הוא האלגוריתם המאמר לאחר ששיפרתי אותו (פירוט נרחב בהמשך). בנוסף יש את האלגוריתמים המוכרים f_classif, relieff, mrmr, rfe שאנחנו משתמשים בהם לשם השוואה והערכת ביצועים של האלגוריתם שלנו על פני אלגוריתמים מוכרים אחרים.

בטבלה הנ"ל ניתן את ערכי ה-p-value השונים בין כל שני אלגוריתמים וכך ניתן להבין מהי המשמעות הסטטיסטית ביניהם.

עבור f_classif, אנחנו רואים שבינו לבין paper_algo (האלגוריתם שמימשתי מהמאמר) יש מובהקות סטטיסטית, וגם בינו לבין האלגוריתם המשופר שלי (paper_algo_improved), מטבע הדברים. כמו כן, גם בינו לבין mrmr ו- rfe יש מובהקות סטטיסטית. יחד עם זאת, בינו לבין rellieff אין מובהקות סטטיסטית, למעשה ה-p-value הוא 0.9, די גבוה שמראה שאין מובהקות סטטיסטית בין השניים.

כעת נסתכל על השורה שמכילה את paper_algo, האלגוריתם שמימשתי של המאמר. ניתן לראות שיש מובהקות סטטיסטית בינו לבין האלגוריתם המשופר, שהרי ה-p-value ביניהם הוא 0.05>0.001. בנוסף ישנה מובהקות סטטיסטית בינו לבין relieff, mrmr אך לא ביחס ל-rfe.

בנוגע לאלגוריתם המשופר שמימשתי, ניתן לראות שיש לו מובהקות סטטיסטית ביחס לכולם (פרט אליו עם עצמו כמובן).

עבור relieff, מעבר להתייחסויות אליו קודם, ניתן לציין שבינו לבין mrmr ו-rfe ישנה מובהקות סטטיסטית. כמו כן, יש מובהקות סטטיסטית בין mrmr ל-rfe, עם p-value שערכו 0.001.

על מנת להשוות בצורה יותר טובה בין האלגוריתמים השונים, ניתחתי את ה-ACC הממוצע עבור כל אחד מהאלגוריתמים, המיצוע הוא עבור כל הדאטאסטים. להלן התוצאות:

filtering algorithm	ACC
f_classif	0.576746
mrmr	0.620289
paper_algo	0.694622
paper_algo_improvement	0.771194
relieff	0.617844
rfe	0.692581

ניתן לשים לב בתוצאות הנ"ל לכמה דברים מעניינים. האלגוריתם f_classif עם ה-ACC הנמוך ביותר (אחוזי דיוק של 0.57). אחריו relieff (עם 0.618), ואז mrmr עם אחוזי דיוק של 0.62.

לאחר מכן, האלגוריתמים rfe והאלגוריתם של המאמר שמימשתי (paper_algo) עם תוצאות מאוד דומות 0.69, עם הפרש קצת לטובת האלגוריתם מהמאמר. ניתן לשים לב שיש הבדל מאוד משמעותי ביניהם לבין f_classif, relieff ו-mrmr באחוזי דיוק הממוצעים.

עוד משהו מעניין שניתן לראות בטבלה המצורפת, הוא שהאלגוריתם של המאמר עם השיפור שלי, משיג תוצאות מאוד גבוהות ביחס לשאר - אחוזי דיוק של 77%, לעומת 69% דיוק שראינו באלגוריתם של המאמר וב-rfe.

בנוסף לכך, מעבר להשוואה על סמך ה-ACC, נרצה להשוות גם על סמך מדד AUC:

filtering algorithm	AUC
f_classif	0.695983
mrmr	0.677230
paper_algo	0.726115
paper_algo_improvement	0.738618
relieff	0.694900
rfe	0.720909

בדומה לניתוח הקודם עבור ה-ACC, גם כאן ניתן לראות שהאלגוריתם המשופר שמימשתי הוא בעל הביצועים הטובים ביותר, גם במדד ה-AUC בנוסף למדד ה-ACC שראינו קודם, ואחריו האלגוריתם של המאמר ו-rfe. כאן ניתן לראות ש-relieff עקף את mrmr במדד ה-AUC בניגוד לקודם לכן, ש-mrmr עקף אותו בפער מאוד קטן במדד ה-ACC.

מסקנות:
האלגוריתם של המאמר השיג תוצאות טובות יותר מ- mrmr, relieff, mrmr ותוצאות די שקולות ל-rfe.
כמו כן, שימוש ב-Random Forest במקום mrmr בשלב ה-feature selection של האלגוריתם במאמר, שיפר את הביצועים כפי שראינו בתוצאות שהצגתי עבור האלגוריתם המשופר ביחס לשאר האלגוריתמים.
בנוסף לכך, ראינו ע"י מבחן פרידמן שהשערת האפס נדחתה. כלומר, ההפרשים במדדי ה-AUC מובהקים סטטיסטית.
לאחר מכן, ביצענו מבחן סטטיסטי כדי להבין בין איזה אלגוריתמים יש מובהקות סטטיסטית. ראינו בין האלגוריתם של המאמר יש מובהקות סטטיסטית ביחס לשאר פרט ל-rfe.
לבסוף, ראינו שעבור האלגוריתם המשופר יש מובהקות סטטיסטית ביחס לכל שאר האלגוריתמים.

לסיכום, כדאי להשתמש באלגוריתם במאמר (ובפרט באלגוריתם המשופר שהצעתי) כאשר אנו רוצים למקסם את התוצאות תוך ביצוע feature selection. לעומת זאת, כאשר נרצה אלגוריתם שירוץ כמה שיותר מהר, עדיף לא להשתמש בו אלא באלטרנטיבות מהירות יותר (ופחות טובות) כגון f_classif.