גרסה 1.00 – 1.6.2009

שימוש בכלי Weka לצורך בחינת אלגוריתמי למידה

ניר אדר

1. מבוא

Weka זהו אוסף של אלגוריתמי למידה, שמטרתו העיקרית היא לשמש כלי למטרות data mining.
ניתן להשתמש בממשק המערכת או להשתמש בקוד המוכן של האלגוריתמים. Weka זמינה כפרוייקט Open Source תחת רשיון GPL, וניתנת להורדה בכתובת
/http://www.cs.waikato.ac.nz/ml/weka

הדגש במסמך זה יהיה על הצגת הבסיס של השימוש ב-Weka, ולאחר מכן – הסבר איך מוסיפים אלגוריתמי למידה משלנו למערכת, ובודקים אותם. ל-Weka יש עומק רב שלא מוסבר במסמך זה.

חשוב לשים לב שכאשר עובדים על דוגמאות קטנות אפשר להשתמש בממשק הויזואלי של Weka. כאשר מריצים על כמות גדולה של נתונים סביר להניח שיהיה צורך לעבור ל-Command Line עקב בעיות זכרון.

אותה נקודה נכונה גם לגבי הידור של קוד Weka – יש צורך במערכת עם זכרון רב על מנת לבצע הידור של הקוד בהצלחה דרך Eclipse, למשל. הידור דרך Command Line יפתור בעיה זו.

מסמך זה מניח ידע קודם בתחום המערכות הלומדות.

2. מושגי יסוד

2.1. אוסף הנתונים

- **Dataset** – קבוצה של פריטי מידע. ניתן להסתכל על Dataset גם בתור טבלה של פריטי מידע. (כל פריט מידע בה הוא שורה אחת). Dataset ממומש ב-Weka במחלקה Weka.core.Instances.
- כל **פריט מידע** בודד ממומש ע"י Weka.core.Instance. לפריט יש מספר תכונות, כאשר כל תכונה יכולה להיות:
  - nominal – יכולה לקבל ערך מתוך קבוצה קטנה של ערכים.
  - numeric – יכולה לקבל מספר שלם או ממשי.
  - string – מחרוזת – קבוצה של תווים מוקפת במרכאות.
  - סוגים נוספים, שפחות רלוונטיים לנו.
- ARFF
  - ה-Dataset בו משתמשת Weka נשמר בקבצים חיצוניים – קבצי ARFF.
  - מבנה הקובץ: header המגדיר את שם ה-Dataset ואת התכונות השונות, ולאחר מכן רשימת הפריטים (comma separated).
  - התכונה האחרונה כברירת מחדל מוגדרת להיות **תכונת המטרה**,
  - דוגמא לקובץ ARFF קטן:

```
@relation audiology
@attribute age_gt_60 { f, t}
@attribute air { mild, moderate, normal, profound, severe}
@attribute airBoneGap { f, t}
@data
t,mild,f
t,mild,t
t,normal,f
f,moderate,t
```

שם ה-Dataset הוא audiology. כל התכונות הן נומינליות, ותכונת המטרה היא airBoneGap.

2.2. מסווגים

- כל אלגוריתם למידה ב-Weka נורש מהמחלקה המופשטת weka.classifiers.Classifier.
- weka.classifiers.Classifier:
  - buildClassifier() - יצירת מסווג מאוסף דוגמאות האימון.
  - clasifyInstance() – הפעלת המסווג שנוצר על דוגמאות לא מסומנות.
  - distributionForInstance() – יצירת פילוג הסתברויות לכל ערך שפונקצית המטרה יכולה לקבל.
- **מסווג** הוא מיפוי מכל התכונות (למעט אחת) אל תכונת היעד.
- **מסווגים נבחרים:**
  - **ZeroR –** weka.classifiers.rules.ZeroR – מסווג שמקבל תמיד את הערך הנפוץ ביותר בקבוצת האימון. מסווג זה פשוט ביותר, אבל בעל חשיבות – הוא משמש כחסם תחתון לביצועי מסווגים על dataset מסויים.
  - **J48 –** weka.classifiers.trees.J48 - הגירסה של Weka לאלגוריתם C4.5.
  - **IBk –** weka.lazy.IBk – מסווג מסוג kNN.
- **מדידת ביצועי מסווג:** הדרך הפשוטה ביותר לבדוק את איכות המסווג היא ע"י בדיקת **הדיוק של המסווג**. בהנתן קבוצה לא מסומנת, כמה פריטים בה המסווג מסווג נכון. סימונים בספרות לדיוק הם accuracy או 1-ErrorRate. מספר דרכים לבצע את המדידה:
  - חלוקת הדוגמאות ל-2 קבוצות זרות: training set ו-test set. הנחה סמויה: ההתפלגות של 2 הקבוצות הזרות זהה.
  - Cross-Validation – מחלקים את קבוצת הנתונים ל-n חלקים. בכל פעם משתמשים ב-(n-1) חלקים בתור קבוצת האימון, והקבוצה הנותרת משמשת כקבוצת הבדיקה. היעילות היא הממוצע של n הבדיקות.

2.3. מסננים

מסננים מאפשרים לנתח את הנתונים תוך התעלמות מתכונה, דוגמאות וכו'.

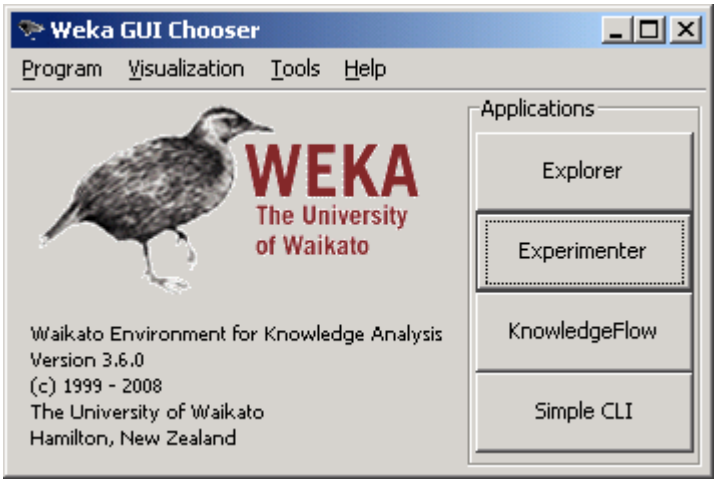
המסננים נמצאים ב-weka.filters – לא נרחיב עליהם במסמך זה.

3. מסווגים

פקודות Command-Line עבור מסווגים, ממדריך Weka:

- **-t**
  specifies the training file (ARFF format)
- **-T**
  specifies the test file in (ARFF format). If this parameter is missing, a crossvalidation will be performed (default: ten-fold cv)
- **-x**
  This parameter determines the number of folds for the crossvalidation. A cv will only be performed if -T is missing.
- **-d**
  The model after training can be saved via this parameter. Each classifier has a different binary format for the model, so it can only be read back by the exact same classifier on a compatible dataset. Only the model on the training set is saved, not the multiple models generated via cross-validation.
- **-l**
  Loads a previously saved model, usually for testing on new, previously unseen data. In that case, a compatible test file should be specified, i.e. the same attributes in the same order.
- **-p #**
  If a test file is specified, this parameter shows you the predictions and one attribute (0 for none) for all test instances.
- **-i**
  A more detailed performance description via precision, recall, true- and false positive rate is additionally output with this parameter. All these values can also be computed from the confusion matrix.
- **-o**
  This parameter switches the human-readable output of the model description off. In case of support vector machines or NaiveBayes, this makes some sense unless you want to parse and visualize a lot of information.

4. הממשק הגרפי של Weka



הממשק הראשי

- Explorer – פלטפורמה לניתוח נתונים.
- Experimenter – סביבה לביצוע ניסויים והשוואות בין מסווגים שונים.
- KnowledgeFlow – מספקת פונקציונליות דומה לזו של Explorer.
- Command Line – Simple CLI שמאפשר להריץ פקודות WEKA.

ה-Explorer שימושי יותר ל-Data Mining ואנשים המשתמשים בפועל בתוכנה.

ה-Experimenter שימושי יותר למטרות למידה של תחום המערכות הלומדות.

4.1. Explorer

ל-Explorer יש פונקציונליות רבה שלא נתעסק בה. הנושא שמעניין אותנו יותר הוא בבדיקת מסווגים על גבי ה-Explorer.

צורת העבודה:

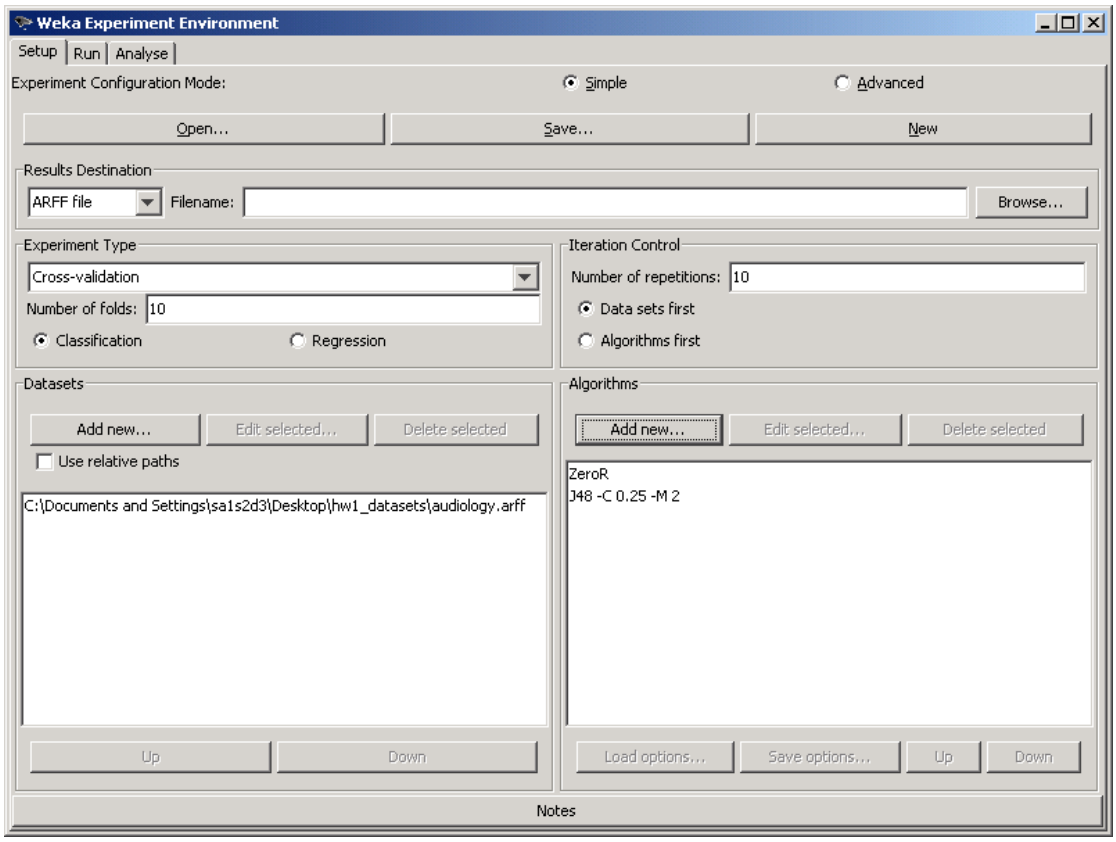
1. טעינת ARFF ע"י "Open File"
2. בטאב ה-classify אפשר להפעיל מסווגים שונים על הנתונים.
3. More Options הוא האפשרות המעניינת ביותר. ניתן לקבל מידע על המסווג הנוצר (למשל – לקבל את עץ ההחלטה הנוצר כאשר משתמשים ב-J48) על ידי האפשרויות הנוספות בכפתור זה.

4.2. Experimenter

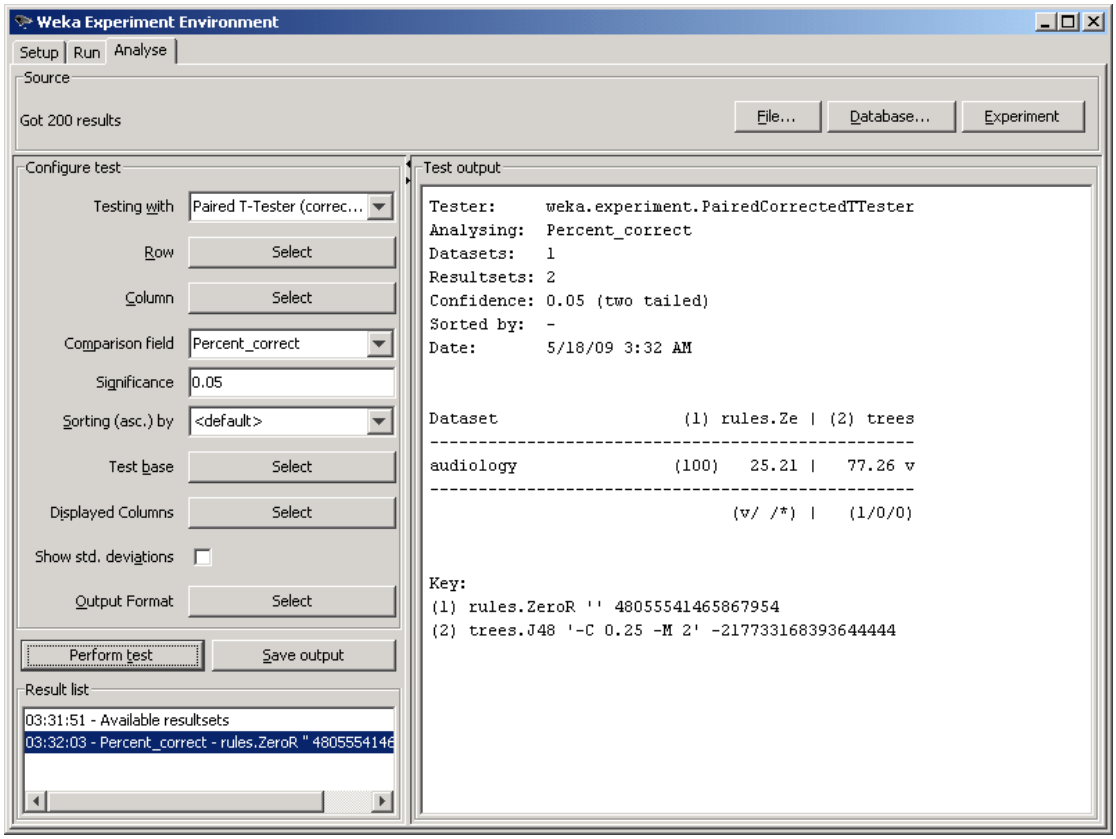
סביבה המאפשרת לבצע ניסויים על מספר מסווגים במקביל. כמו שאר חלקי התוכנית – גם את ה-Experimenter ניתן להפעיל באמצעות Command Line. במדריך Weka ניתן למצוא את הפרמטרים המתאימים.

הסביבה תומכת בביצוע החישובים על מחשב אחד או פיצולם על מספר מחשבים ברשת לחסכון בזמן עיבוד.

הממשק הבסיסי פשוט. לוחצים "New" כדי להתחיל ניסוי חדש, מוסיפים את האלגוריתמים השונים שרוצים לבחון לאיזור ה-Algorithms, את הנתונים ל-Dataset ומתחילים בניסוי.



ב-Tab של Run אנחנו מתחילים את הניסוי. ב-Tab של Analyze אנחנו רואים תוצאות אחרי שהניסוי הסתיים.



דוגמא לתוצאות ניסוי

אנחנו רואים שעבור הנתונים שהבאנו, ZeroR צדק ב-25.21 מהמקרים, ואילו J48 צדק ב-77.26% מהמקרים.

השדה המעניין ביותר במסך הזה – Comparsion Field – מאפשר להשוות אספקטים שונים בין האלגוריתמים.

5. הוספת מסווגים ל-Weka

הוספת מסווגים – נקודות:

- כדי להוסיף מסווגים יש לקמפל מחדש את Weka עם המסווג החדש.
- הקוד של Weka נמצא ב-weka-src.jar שמגיע ביחד עם התקנת Weka. כל תוכנת כיווץ סטנדרטית יודעת לפתוח קובץ זה.

יש ליצור את המסווג בהורשה מהמחלקה weka.classifiers.Classifier. המימוש כמובן כרצוננו.

אחרי שיצרנו את ה-class של המסווג, יש להוסיף אותו לקובץ הבא על מנת שיופיע בממשק המשתמש:

weka-src\src\main\java\weka\gui\GenericObjectEditor.props

כדי לראות איפה בדיוק צריך להוסיף אותו – ניתן לבצע חיפוש למסווג דומה. (למשל אם יצרנו גרסה של עץ החלטה, נחפש J48) ונראה את המקום בקוד.

הקובץ עצמו הוא פשוט אוסף רשימות של מחלקות – שמוצגות בחלקים שונים בממשק.

אחרי ההוספה יש לקמפל מחדש את הקוד, וניתן להשתמש במסווג החדש.

התעסקות בקוד של Weka עם Eclipse – ניתן למצוא הוראות בקישור הזה.
(http://weka.wiki.sourceforge.net/Eclipse+3.4.x+(weka-src.jar

לא בכל מחשב ניתן לבצע זאת – במספר מחשבים הסביבה מודיעה שאין מספיק זכרון.

אם ישנן בעיות – ניתן להשתמש ב-ANT להידור נוח.

6. מקורות

1. Weka 3.6.0 Manual
2. http://weka.wiki.sourceforge.net/GenericObjectEditor+(developer+version)