Getting started with Weka

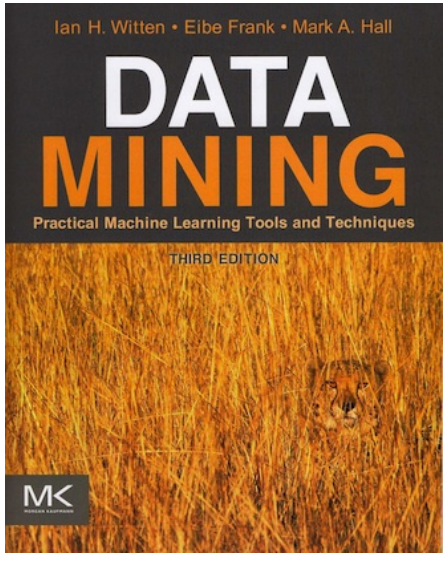
Simon Šuster, University of Groningen

Course *Learning from data*
November 18, 2013

Weka toolkit

- collection of Machine learning algorithms and data-preprocessing tools
- in Java, free under GNU GPL
- allows one to quickly try out several methods
- ease of use (especially GUIs)

- coverage of some NLP algorithms is limited: structured prediction
- book:



Using Weka I

Through:

- Command-line
- GUIs (Explorer, Knowledge Flow, Experimenter, Command-line interface)

= Choice is yours!

Use from `/net/aps/64/src/weka-3-6-10`
Or:
Download from
`http://www.cs.waikato.ac.nz/ml/weka/downloading.html`
and unzip

For command-line use, add the following to your `.bashrc` file:

- `export WEKAINSTALL=/path/to/weka-3-6-10`
- `export CLASSPATH=$WEKAINSTALL/weka.jar:$CLASSPATH`

Using Weka II

Running Weka

- GUIs: `java -jar weka.jar`
- Following slides concentrate on running Weka from the command line

For FAQ about Weka, see `http://weka.wikispaces.com`

Preparing the data I

Weka uses the **ARFF** (Attribute Relation File Format) format, consisting of lines with:

- descriptors (`@...`)
- comments (`%`)
- learning data

Precisely:

- `@relation name` : name/description of the dataset
- `@attribute name type/set_of_values` : attribute (feature) with its type
  - most common types: `numeric` (integer or real), `nominal` (included in curly braces), `string`
- `@data` : marks beginning of the learning data
- 1 instance per line, comma-separated
- class label usually the last element, sometimes first

Preparing the data II

String attributes

- Without data transformation, one learning instance would be associated with a potentially very long string (e.g. a document)
- Not very useful, it would terribly overfit the training data
- Strings must be quoted
- To use strings in ML, we normally extract smaller units by converting them into numeric attributes
- Each word can then be associated with a count

```
@attribute   graag     numeric
@attribute   grapje    numeric
@attribute   grappig   numeric
@attribute   gratis    numeric
@attribute   great     numeric
@attribute   green     numeric
@attribute   groot     numeric
@attribute   grote     numeric
```

Preparing the data III

Sparse data

Word features are usually sparse, so the instance representation is also sparse, meaning:

- Only active attributes (words actually encountered) are included
- Words are refered to with integers
- You know it's "sparse" because of `{}`

Preparing the data IV

- Elements of an instance are comma separated
- First integer in an element refers to the *attribute index*
- Second number indicates *attribute presence* (as in example) or *count* (or some other measure):

```
@relation '_home_p262594_Datasets_twitter2_data_source-...
@attribute @@class@@ {nl,other}
@attribute # numeric
@attribute #megazinnen numeric
@attribute #mls numeric
@attribute #retweet numeric
/.../

@data
{6 1,9 1,48 1,114 1,440 1,459 1,548 1,688 1,824 1,976 1}
{238 1,252 1,897 1}
{0 other,7 1,134 1,312 1,432 2,615 1,631 1,734 1,920 1}
```

Learning from data I

- Check available classifiers:
  `jar tf weka.jar` , or see
  `http://weka.sourceforge.net/doc.dev/`
  (note: names of classifier groups inside `weka.classifiers` need some gettting used to)

- Get usage for a classifier:
  ```
  java weka.classifiers.bayes.NaiveBayes -h
  ```

- Get information about a dataset:
  ```
  java weka.core.Instances FILE.arff
  ```

- Run a classifer using (default) 10-fold cross validation:
  ```
  java weka.classifiers.bayes.NaiveBayes -t FILE.arff
  ```

Learning from data II

- To evaluate on a test set:
  - build the model:
    ```
    java weka.classifiers.bayes.NaiveBayes
    -t TRAIN_FILE.arff -d MODEL_FILE
    ```
  - evaluate:
    ```
    java weka.classifiers.bayes.NaiveBayes
    -T TEST_FILE.arff -l MODEL_FILE
    ```

- Some useful flags in combination with a classifier:
  - `-i` :
    output precision, recall, F-score and the like
  - `-o` :
    don't output the classifier
  - `-p 0` :
    obtain predictions

Learning from data III

- `-c`:
  class label position: the class label as the last attribute is assumed. Use `-c 1` specify the label as the first attribute
- `-x`:
  number of folds in cross validation; overwrite the default to speed up processing
- `-d`:
  save the model
- `-l`:
  use an existing model

- Miscellaneous
  `-Xmx1g`:
  solves memory issues by increasing the heap size (say, 1GB)

Further data preparation I

Many useful tools can be found in `weka.core.converters` and `weka.filters`:

- csv to arff conversion
- resampling
- attribute selection
- discretization (numeric to nominal)
- type conversion
- ...