Advanced Data Mining with Weka

Class 5 – Lesson 1

Invoking Python from Weka

Peter Reutemann

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 5.1: Invoking Python from Weka
Lesson 5.1: Invoking Python from Weka

Scripting

Pros

- script captures preprocessing, modeling, evaluation, etc.
- write script once, run multiple times
- easy to create variants to test theories
- no compilation involved like with Java

Cons

- programming involved
- need to familiarize yourself with APIs of libraries
- writing code is slower than clicking in the GUI
Invoking Python from Weka

Scripting languages

- **Jython** - https://docs.python.org/2/tutorial/
  - pure-Java implementation of Python 2.7
  - runs in JVM
  - access to all Java libraries on CLASSPATH
  - only pure-Python libraries can be used

- **Python**
  - invoking Weka from Python 2.7
  - access to full Python library ecosystem

- **Groovy** (briefly) - http://www.groovy-lang.org/documentation.html
  - Java-like syntax
  - runs in JVM
  - access to all Java libraries on CLASSPATH
## Invoking Python from Weka

### Java vs Python

<table>
<thead>
<tr>
<th>Java</th>
<th>Output</th>
</tr>
</thead>
</table>
| ```java
public class Blah {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            System.out.println((i+1) + ": Hello WekaMOOC!");
        }
    }
}
``` | 1: Hello WekaMOOC! |
| | 2: Hello WekaMOOC! |
| | 3: Hello WekaMOOC! |
| | 4: Hello WekaMOOC! |
| | 5: Hello WekaMOOC! |
| | 6: Hello WekaMOOC! |
| | 7: Hello WekaMOOC! |
| | 8: Hello WekaMOOC! |
| | 9: Hello WekaMOOC! |
| | 10: Hello WekaMOOC! |

<table>
<thead>
<tr>
<th>Python</th>
</tr>
</thead>
</table>
| ```python
for i in xrange(10):
    print("%i: Hello WekaMOOC!" % (i+1))
``` |
Invoking Python from Weka

Package manager

- start Package manager from the main GUI (from the Tools menu)
- install the following packages
  - tigerJython 1.0.0
    *GUI for writing/running Jython scripts*
  - jfreechartOffscreenRenderer 1.0.2
    *JFreeChart offers nice plots (used in Lesson 3)*
- after restarting Weka, you can start Jython GUI
  - Tools → Jython console

Note: I'm using Weka 3.7.13
**Invoking Python from Weka**

**TigerJython Interface**

- **Execute your script**
- **Write your script here**
- **Debug mode on/off**
- **Output/Errors**

**Preferences**
- Decrease font
- Add support for tabs
Invoking Python from Weka

Debugging your scripts

- Let’s re-use example from *Java vs Python* comparison
  ```python
  for i in xrange(10):
      print("%i: Hello WekaMOOC!" % (i+1))
  ```

- Select "Toggle debugger" from the "Run" menu
- Execute the script

![Diagram showing current execution pointer, output generated so far, and current state of variables with speed of execution options.]
Invoking Python from Weka

Information sources for Weka API

- Javadoc - detailed, per-class information
  - online (latest developer version)
  - http://weka.sourceforge.net/doc.dev/
  - Weka release/snapshot
  - see the doc directory of your Weka installation
- Example code
  - check the wekaexamples.zip archive of your Weka installation
- Weka Manual
  - check WekaManual.pdf of your Weka installation
  - Appendix → Using the API
Invoking Python from Weka

What we need...

- Weka
  - `weka.filters.Filter` - for filtering datasets
  - `weka.filters.unsupervised.attribute.Remove` - removes attributes
  - `weka.core.converters.ConverterUtils.DataSource` - loads data
- Environment variable
  - set `MOOC_DATA` to point to your datasets

In Windows:

Control panel ->
System and Security ->
System ->
Advanced system settings ->
Environment Variables ->
New
```python
import weka.filters.Filter as Filter
import weka.filters.unsupervised.attribute.Remove as Remove
import weka.core.converters.ConverterUtils.DataSource as DS
import os

data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "iris.arff")
rem = Remove()
rem.setOptions(['-R', 'last'])
dataNew = Filter.useFilter(data, rem)
print(dataNew)
```
Invoking Python from Weka

What we did...

- Installed tigerJython
- Seen that Python is easy to read and write
- Learned about API documentation resources
- Wrote our first Jython script
Advanced Data Mining with Weka

Class 5 – Lesson 2

Building models

Peter Reutemann

Department of Computer Science
University of Waikato
New Zealand
Lesson 5.2: Building models

Class 1  Time series forecasting

Class 2  Data stream mining in Weka and MOA

Class 3  Interfacing to R and other data mining packages

Class 4  Distributed processing with Apache Spark

Class 5  Scripting Weka in Python

Lesson 5.1 Python from Weka

Lesson 5.2 Building models

Lesson 5.3 Visualization

Lesson 5.4 Invoking Weka from Python

Lesson 5.5 A challenge, and some Groovy

Lesson 5.6 Course summary
Building models

What we need...

- **Weka**
  - `weka.classifiers.Evaluation` - for evaluating classifiers
  - `weka.classifiers.*` - some classifiers
  - `weka.filters.Filter` - for filtering datasets
  - `weka.filters.*` - some filters

- **Java**
  - `java.util.Random` - for randomization
Building models

Build J48 classifier

- Script: build_classifier.py
- Output

J48 pruned tree
------------------

hardness <= 70
  | strength <= 350
  |   | family = ?
  |   |   | surface-quality = ?
  |   |   |   condition = ?: 3 (68.0/1.0)
  |   |   |   condition = S
  |   |   |   | thick <= 0.75: 3 (5.0)
  |   |   |   | thick > 0.75
  |   |   |   |   | thick <= 2.501: 2 (81.0/1.0)
  |   |   |   |   | thick > 2.501: 3 (2.0)
  |   |   |   | condition = A: 2 (0.0)
  |   |   |   | condition = X: 2 (0.0)
  |   |   | surface-quality = D: 3 (55.0)
...

You can download the scripts and data files from the course page for this lesson

Hint: ensure that anneal.arff is in the directory indicated by your MOOC_DATA environment variable
Cross-validate J48

- **Script**: crossvalidate_classifier.py
- **Output**

```plaintext
=== J48 on anneal (stats) ===
Correctly Classified Instances 884  98.441 %
Incorrectly Classified Instances 14   1.559 %
Kappa statistic                0.9605
Mean absolute error            0.0056
Root mean squared error        0.0669
Relative absolute error        4.1865 %
Root relative squared error    25.9118 %
Coverage of cases (0.95 level) 98.7751 %
Mean rel. region size (0.95 level) 16.7223 %
Total Number of Instances     898

=== J48 on anneal (confusion matrix) ===
  a  b  c  d  e  f  <-- classified as
  5  0  3  0  0  0  |  a = 1
  0  99 0  0  0  0  |  b = 2
  0  2 680 0  0  2  |  c = 3
...```
Building models

Predict class labels

- Script: make_predictions-classifier.py
- Output

```python
array('d', [0.0, 0.0, 1.0, 0.0, 0.0, 0.0]) - 2.0 - 3
array('d', [0.021739130434782608, 0.0, 0.9782608695652174, 0.0, 0.0, 0.0]) - 2.0 - 3
array('d', [0.0, 0.0, 1.0, 0.0, 0.0, 0.0]) - 2.0 - 3
array('d', [0.0, 0.0, 1.0, 0.0, 0.0, 0.0]) - 2.0 - 3
array('d', [0.0, 0.0, 1.0, 0.0, 0.0, 0.0]) - 2.0 - 3
array('d', [0.0, 0.9811320754716981, 0.018867924528301886, 0.0, 0.0, 0.0]) - 1.0 - 2
array('d', [0.021739130434782608, 0.0, 0.9782608695652174, 0.0, 0.0, 0.0]) - 2.0 - 3
...```

Ensure that anneal_train.arff and anneal_unlbl.arff are in the appropriate directory.
Building models

What we did...

- built a classifier
- output statistics from cross-validation
- used built model to make predictions
Advanced Data Mining with Weka

Class 5 – Lesson 3

Visualization

Peter Reutemann

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 5.3: Visualization

Class 1
Time series forecasting

Class 2
Data stream mining in Weka and MOA

Class 3
Interfacing to R and other data mining packages

Class 4
Distributed processing with Apache Spark

Class 5
Scripting Weka in Python

Lesson 5.1 Invoking Python from Weka
Lesson 5.2 Building models
Lesson 5.3 Visualization
Lesson 5.4 Invoking Weka from Python
Lesson 5.5 A challenge, and some Groovy
Lesson 5.6 Course summary
Visualization

What we need...

- JFreeChart
  - easier to use than some of Weka's plotting
  - install the *jfreechartOffscreenRenderer* package
  - Javadoc
  - http://www.jfree.org/jfreechart/api/javadoc/
  - classes
    - `org.jfree.data.*` - some dataset classes
    - `org.jfree.chart.ChartFactory` - for creating plots
    - `org.jfree.chart.ChartPanel` - for displaying a plot
    - `weka.gui.*` - for tree/graph visualizations

- Java
  - `javafx.swing.JFrame` - window for displaying plot
Visualization

Classifier errors with size of error

- Script: crossvalidate_classifier-errors-bubbles.py
- Output

You can download the scripts and data files from the course page for this lesson.

Hint: ensure that bodyfat.arff is in the directory indicated by your MOOC_DATA environment variable.
Visualization

Multiple ROC

- Script: display_roc-multiple.py
- Output

Ensure that balance-scale.arff is in the appropriate directory
Visualization

Tree

- Script: display_tree.py
- Output

Ensure that iris.arff is in the appropriate directory
Visualization

Network graph

- Script: display_graph.py
- Output
Visualization

What we did...

- Used JFreeChart for plotting
  - classifier errors
  - ROC
- Displayed J48 decision tree
- Visualized BayesNet network graph
Advanced Data Mining with Weka

Class 5 – Lesson 4

Invoking Weka from Python

Peter Reutemann

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 5.4: Invoking Weka from Python

Class 1  Time series forecasting

Class 2  Data stream mining in Weka and MOA

Class 3  Interfacing to R and other data mining packages

Class 4  Distributed processing with Apache Spark

Class 5  Scripting Weka in Python

Lesson 5.1 Invoking Python from Weka

Lesson 5.2 Building models

Lesson 5.3 Visualization

Lesson 5.4 Invoking Weka from Python

Lesson 5.5 A challenge, and some Groovy

Lesson 5.6 Course summary
Invoking Weka from Python

Why the other way?

- Jython limits you to pure-Python or Java libraries
- Weka provides only modeling and some visualizations
- Python offers much more:
  - NumPy - e.g., efficient arrays and matrices
  - SciPy - e.g., linear algebra, optimization, integration
  - matplotlib - plotting library
  - more info: https://wiki.python.org/moin/NumericAndScientific
Invoking Weka from Python

What we need...

- Install Python 2.7
  - https://www.python.org/downloads/
  - Java and Python need the same “bitness” (either 32bit or 64bit)
- Set up environment for compiling libraries
  - on Linux a no-brainer
  - OSX and Windows quite a bit of work involved
- Install python-weka-wrapper library
  - https://pypi.python.org/pypi/python-weka-wrapper
- Instructions and videos for all this can be found here
Invoking Weka from Python

**python-weka-wrapper**

- fires up JVM in the background and communicates with JVM via JNI
- provides a thin wrapper around Weka's superclasses (classifiers, filters, ...)
- provides a more “pythonic” API - some examples:
  - Python properties instead of get/set-method pairs
    - `options` instead of `getOptions/setOptions`
  - lowercase + underscore instead of Java's camel case
    - `crossvalidate_model` instead of `crossValidateModel`
- convenience methods
  - `data.class_is_last()` instead of `data.setClassIndex(data.numAttributes()-1)`
- plotting is done by matplotlib
Invoking Weka from Python

Cross-validate J48

- Script: pww-crossvalidate_classifier.py
- Output

```
DEBUG:weka.core.jvm:Adding bundled jars
DEBUG:weka.core.jvm:Adding Weka packages
DEBUG:weka.core.jvm:package_dir=/home/fracpete/wekafiles/packages
...DEBUG:weka.core.jvm:MaxHeapSize=default
DEBUG:javabridge.jutil:Creating JVM object
DEBUG:javabridge.jutil:Signalling caller
...
=== J48 on anneal (stats) ===
Correctly Classified Instances         884               98.441  %
Incorrectly Classified Instances        14                1.559  %
Kappa statistic                          0.9605
Mean absolute error                      0.0056
Root mean squared error                  0.0669
Relative absolute error                  4.1865 %
Root relative squared error             25.9118 %
Coverage of cases (0.95 level)          98.7751 %
Mean rel. region size (0.95 level)      16.7223 %
Total Number of Instances              898
```

You can download the scripts and data files from the course page for this lesson

Hint: ensure that anneal.arff is in the directory indicated by your MOOC_DATA environment variable
Invoking Weka from Python

Classifier errors with size of error

- Script: pww-crossvalidate_classifier-errors-bubbles.py
- Output

Ensure that bodyfat.arff is in the appropriate directory
Invoking Weka from Python

Multiple ROC

- Script: pww-display_roc-multiple.py
- Output

Ensure that balance-scale.arff is in the appropriate directory
Invoking Weka from Python

What we did...

- Installed Python and additional modules via Python's pip
- Used Weka from within a “native” Python environment
Advanced Data Mining with Weka

Class 5 – Lesson 5
A challenge, and some Groovy

Peter Reutemann
Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 5.5: A challenge, and some Groovy

Class 1  Time series forecasting

Class 2  Data stream mining in Weka and MOA

Class 3  Interfacing to R and other data mining packages

Class 4  Distributed processing with Apache Spark

Class 5  Scripting Weka in Python

Lesson 5.1 Invoking Python from Weka
Lesson 5.2 Building models
Lesson 5.3 Visualization
Lesson 5.4 Invoking Weka from Python
Lesson 5.5 A challenge, and some Groovy
Lesson 5.6 Course summary
A challenge and some Groovy

The challenge

- Annual shoot-out of the Council for Near-Infrared Spectroscopy (CNIRS)
- Shoot-out process
  - Build data on training data ("calibration")
  - Validate model on separate dataset ("validation")
  - Generate and submit predictions ("test set")
- We use the 2014 datasets
  

You can download the file challenge.text from the course page for this lesson. It gives information about the challenge.
**A challenge and some Groovy**

**The challenge**

- **What to do?**
  - Download the CSV files for Dataset 1 and 2 (calibration/test)
  - Generate data for Weka, build (“calibration”) and evaluate models (“test”)
  - Class attribute is “reference value”
  - Don't include “sample #”

- **What to beat?**
  - Dataset 1
    - $CC = 0.8644$
    - $RMSE = 0.384$
  - Dataset 2
    - $CC = 0.9986$
    - $RMSE = 0.0026$
Install Groovy

- open Package manager (under Tools)
- scroll down and select kfGroovy
- click on Install
- after restarting Weka, open Groovy console (under Tools)
A challenge and some Groovy

Groovy basics

- Grammar is derived from Java (but no semicolons!)
- “def” defines a variable, no type required
- lists are similar to Python ones: [1, “a”, true]
- maps are similar to Python dictionaries: [red: '#FF0000', green: '#00FF00']
- Enhances Java syntax, e.g.:
  - multi-line strings using triple single quotes
  - string interpolation
  - default imports of commonly used packages
  - closures (not the same as Java 8 lambdas)
- Differences
A challenge and some Groovy

Groovy loops

- standard Java for-loop and while-loop
- using java.lang.Number object methods
  - `upto`
    ```groovy
    0.upto(10) {println(it)}
    prints numbers 0 to 10
    ```
  - `times`
    ```groovy
    5.times {println(it)}
    prints numbers 0 to 4
    ```
  - `step`
    ```groovy
    0.step(10, 2) {println(it)}
    prints numbers 0, 2, 4, 6, 8
    ```

A challenge and some Groovy

Make predictions

- Script: make_predictions-classifier.groovy
- Output

J48 pruned tree
------------------

hardness <= 70
  |    strength <= 350
  |    |    family = ?
  |    |    |    surface-quality = ?
  |    |    |    condition = ?: 3 (46.0/1.0)
  |    |    |    condition = S

...
A challenge and some Groovy

Multiple ROC

- Script: roc_multiple.groovy
- Output

Ensure that balance-scale.arff is in the appropriate directory
A challenge and some Groovy

What we did...

- Tried our hands at some real-world data modeling
- Learned another scripting language
Advanced Data Mining with Weka

Class 5 – Lesson 6

Course summary

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand
Lesson 5.6: Course summary

Class 1  Time series forecasting

Class 2  Data stream mining in Weka and MOA

Class 3  Interfacing to R and other data mining packages

Class 4  Distributed processing with Apache Spark

Class 5  Scripting Weka in Python

Lesson 5.1 Invoking Python from Weka

Lesson 5.2 Building models

Lesson 5.3 Visualization

Lesson 5.4 Invoking Weka from Python

Lesson 5.5 A challenge, and some Groovy

Lesson 5.6 Course summary
Summary

From “More Data Mining with Weka”

What have we missed?

- **Time series analysis** — Environment for time series forecasting
- **Stream-oriented algorithms** — MOA system for massive online analysis
- **Multi-instance learning** — Bags of instances labeled positive or negative, not single instances
- **One-class classification**
- **Interfaces to other data mining packages** — LibSVM, LibLinear, R
- **Distributed Weka with Hadoop**
- **Latent Semantic Analysis**

These are available as Weka “packages”
Summary

Advanced Data Mining with Weka

What did we do?

✔ Time series analysis — Environment for time series forecasting
✔ Stream-oriented algorithms — MOA system for massive online analysis
✔ Multi-instance learning — Bags of instances labeled positive or negative, not single instances
✔ One-class classification (Activity 3.1)
✔ Interfaces to other data mining packages — LibSVM, LibLinear, R
✔ Distributed Weka with Hadoop and SPARK
✔ Latent Semantic Analysis

✔ Scripting in Python and Groovy
✔ Applications

✔ These are available as Weka “packages”
Summary

Applications

- **Infrared data from soil samples**
  
  _Hard to achieve sufficiently good performance for practical application_
  
  _Need to investigate outliers, more classifier/filter tweaking_

- **Bioinformatics: signal peptide prediction**
  
  _Domain knowledge is vital: collaborate with experts!_
  
  _Accurate prediction vs explanatory model?_
  
  _Overfitting the data_

- **Functional MRI Neuroimaging data**
  
  _Enormous 4D data_
  
  _Amalgamating data from different runs?_
  
  _Combining data from different subjects_
  
  _In an early competition, demographic data alone did well!_

- **Image classification**
  
  _Specialist feature extraction techniques for different kinds of data_
Summary

More practical data mining: Kaggle competitions (https://www.kaggle.com/)

- **Featured competitions**
  *Win money!*

- **Recruitment competitions**
  *Get jobs!*

- **Interesting datasets/Playground**
  *Play around*

- **Getting started**
  *Educational*

- **Completed competitions**
  *Past solutions*
  *Kaggle blog*
  *Interviews with winners*
  *Descriptions of winner’s solution*
Summary

Ethics: don’t forget!

- “More than ever, knowingly or unknowingly, consumers disseminate personal data in daily activities”
- “As companies seek to capture data about consumer habits, privacy concerns have flared”
- “Data mining: where legality and ethics rarely meet”
- “Big data might be big business, but overzealous data mining can seriously destroy your brand”
- “What big data needs: A code of ethical practices”
Advanced Data Mining with Weka

Department of Computer Science
University of Waikato
New Zealand

Creative Commons Attribution 3.0 Unported License
creativecommons.org/licenses/by/3.0/

weka.waikato.ac.nz