CIS 730 Artificial Intelligence
CIS 530 Principles of Artificial Intelligence
Fall 2013

Homework 8 of 8: Machine Problem (MP8)
Reasoning and Learning, Part II:
Machine Learning Basics

Assigned: Friday 15 Nov 2013
Due: Fri 22 Nov 2013 (before midnight)

The purpose of this assignment is to help you exercise some basic concepts of machine learning.

This homework assignment is worth a total of 100%.
Include a copy of your solution *with* the final project submission before the due date.

Preliminaries:

- In your web browser, open the URL http://www.cs.waikato.ac.nz/~ml/weka/
- Download the *Waikato Environment for Knowledge Analysis (WEKA)* v3.6.9 (GUI version © 2013 I. H. Witten, E. Frank, *et al.*) to your local system (this can be a Windows, Unix, Mac, or other system, but the binaries are precompiled for ix86 Linux. Follow the instructions in the *WEKA3* manual for installing it into your home directory.
- Download the University of California Irvine (UCI) Machine Learning and KDD Repository data sets, available at http://prdownloads.sourceforge.net/weka/uci-20070111.tar.gz. (See also http://www.cs.waikato.ac.nz/~ml/weka/index_datasets.html for the complete list of data collections curated by the Waikato Machine Learning Group.)
- Create a subdirectory "UCI" of the "data" directory for your installed copy of WEKA and unpack the UCI ML/KDD data there.
- Launch WEKA and start the Explorer application from the Applications menu.
- Click "Open File" on the Preprocess tab. Load each of the specified data sets (in Windows, browse to your "Program Files (x86)/Weka-3-6/data" directory), one at a time. The data sets are:
  - Jergen's audiology data set from Baylor College of Medicine, donated by Porter (`audiology.arff`, http://bit.ly/150PsCc)
  - Quinlan's credit approval data set (`credit-a.arff`, http://bit.ly/150PnOU)
  - Nakai's *E. coli* data set from the Institute of Molecular and Cellular Biology at Osaka University, donated by Horton (`ecoli.arff`, http://bit.ly/150PlGS)
  - The Hayes-Roths' data set, donated by Aha (`hayes-roth_test.arff` and `hayes-roth_train.arff`, http://bit.ly/150Pjyl)
  - Schlimmer's U.S. Congressional voting record classification data set (`vote.arff`, http://bit.ly/150PxWK)
- Apply the specified *inducers* (inductive learning algorithms).
- Refer to the new Weka wiki (http://weka.wikispaces.com) for documentation.

1. **(40% for CIS 530, 30% for CIS 730) Supervised classification learning: running experiments and reporting results.** Create a table with one row per inducer and three columns per data set. The table should look like this:

Inducer	Data Set 1			Data Set 2		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
1						
2						
3						

After loading a data set using Open File in the Preprocess tab, click Choose in the Classify tab to select an inducer. If an inducer is not applicable to a particular data set, it will be shown in red. Indicate "N/A" for all results in such cases.

Report the mean accuracy, precision, and recall for the following inducers on each of the five data sets above.

- trees → J48 (Decision Trees)
- bayes → Naïve Bayes (Simple Bayes)
- functions → RBFNetwork (Radial Basis Function network)
- lazy → IBk (k-Nearest Neighbor)

2. **(40% for CIS 530, 30% for CIS 730) Interpreting results from classification learning.**

   a) **Accuracy, precision, and recall.** For each data set, which inducer achieved the highest accuracy? Did it also achieve the highest precision and recall? If not, what can you conclude from the performance measures?
   b) **Evaluating inducers.** Is one inducer giving you consistently superior results across <u>all</u> data sets? Why or why not?

3. **Experiment Replication Script (20% for CIS 530, 40% for CIS 730).** Write a script in any scripting language (e.g., Unix shell script, Tcl/Tk, Perl, Ruby, Python, Lua, or VB.net) to automate the collection of experimental results for your final project report. Alternatively, write a data generator or preprocessor using a scripting language or a high-level imperative language such as C++ or Java. Examples include:

   a) A Perl script to run *Angband* using different levels of training (the basic APWborg, trained with 1000 turns of combat data, trained with 2000, 4000, 8000, 16000)
   b) A Python driver for training a *TAC-SCM* agent and then bringing them into new games on a running server
   c) A Java program to generate test data for benchmarking the import module you write to map data from a particular protein database format into the unified protein interaction format. Measure the ontology reasoner and I/O costs separately.

   Turn in a standalone table of results produced by this program along with your source code.

Class participation (required). Think about the following question and post your answer to the class participation thread for MP8 in the KSOL messaage board by Fri 06 Dec 2013.

Collect a *baseline* result and compare it to your system's. Compute *percent decrease* (*or increase*) *versus baseline* for a chosen metric. This can be test set accuracy, test set precision/recall/*F*-score (*aka* F1 score or f-measure), or area under the ROC curve.

Report the baseline and new values, and the difference (incremental gain or loss due to your improvement) over one or more experimental *replications* – i.e., repetitions of the experiment under the same task specification. List results in your final project report.

References:
http://en.wikipedia.org/wiki/Precision_(information_retrieval)
http://en.wikipedia.org/wiki/Recall_(information_retrieval)
http://en.wikipedia.org/wiki/F1_Score
http://en.wikipedia.org/wiki/Sensitivity_and_specificity
http://en.wikipedia.org/wiki/ROC_Curve