An Easy Performance Evaluation Program for AQ Learning Programs

Q. Zhang R. S. Michalski

P 97-23
MLI 97-16

An Easy Performance Evaluation Program for AQ Learning Programs

Qi Zhang Ryszard S. Michalski*

Machine Learning and Inference Laboratory
School of Information Technology and Engineering
George Mason University
Fairfax, Virginia 22030-4444
{qzhang, michalsk}@aic.gmu.edu

*Also with GMU Department of Systems Engineering, and
the Institute of Computer Sciences, Polish Academy of Science

Publication No.
P 97-23

Reports of the Machine Learning and Inference Laboratory
MLI 97-16

December 1997

ABSTRACT

This paper describes a program EPE (Easy Performance Evaluation) designed for AQ learning programs which serves two purposes: firstly, it provides the user with an automatic tool for tesing the performance of AQ learning programs in terms of predicative accuracy using different experimentation methods over any number of runs on a given problem; secondly, it shows the user the performance improvment when more training examples are fed to an AQ program via a so-called multi-stage process.

Keyword: Evaluation of Machine Learning Algorithms.

ACKNOWLEDGMENTS

This research was conducted in the Machine Learning and Inference Laboratory at George Mason University. The Laboratory's activities are supported in part by the Defense Advanced Research Projects Agency under grant F49620-95-1-0462, administered by the Air Force Office of Scientific Research, in part by the National Science Foundation under grants DMI-9496192 and IRI-9020266, and in part by the Office of Naval Research under grant N00014-91-J-1351

1 INTRODUCTION

Inductive learning from examples has long since been an important research area in machine learning and many learning programs were invented, such as AQ (Michalsk et al., 1986), CN2 (Clark & Niblett, 1989), C4.5 (Quinlan, 1991), etc.

The performance of an inductive learning algorithm can be evaluated in aspects such as predicative accuracy, simplicity of induced knowledge. *Predicative accuracy* is the percentage of correctly predicated/classified unobserved examples by the learning program. As for simplicity, different measures have been adopted for different types of induced knowledge. For instance, the number of nodes is a commonly used measure for an induced decision tree; for rule learning programs, the number of testing condition in rules (selectors in the case of AQ programs) is often used to describe complexity of induced rules.

The performance of a learning program is evaluated by means of experiments. In machine leanring research, three evaluation methods, namely Hold-out, k-fold and Leave-one-out, are widely used in different problem settings.

It is also recognized that the performance of a learning programs cannot be reliably evaluated via one run of the program on a given problem. Results should be summarized over many runs (say 20) and should be statistically convicing. In addition it is useful to show the user of a learning program how its performance is improved with more examples avaialbe.

This paper describes a program EPE (Easy Performance Evaluation) designed for AQ learning programs which meets the above needs. In a nutshell, EPE serves two purposes: firstly, it provides the user with an automatic tool for tesing the performance of AQ learning programs in terms of predicative accuracy using different experimentation methods over any number of runs on a given problem; secondly, it shows the user the performance improvment when more training examples are fed to an AQ program via a so-called multi-stage process.

EPE is an extensive correction and expansion of the EDC (Experiment Design Component) system that was originally developed by John Doulamis. The current version is implemented in ANSI C.

2 EVALUATION

Corresponding to the amount of data available, three evaluation methods are commonly used by machine learning researchers.

Hold-out:: In this method, the avaiable data is split into two disjoint sets. One set is used for training to get the induced knowledge, and the other one is hold out until the training is completed and then is used to test the performance of the induced knowledge. This technique is very reasonable in case of plenty of data, say, about 1000 examples or more. Usually, two-thirds of the available data are taken for training and the remaining for testing.

k-fold: The available data is divided into mutually disjoint k subsets of equal size. Each set is used once for testing and all other sets for training. The average over the k train-test sessions is taken as the performance result. This technique is suitalbe when only a limited sample of data is avaiable.

Leave-one-out:: One example from the avaialbe data is taken out for testing and all others used for training. Repeat this process for each example. The average over all the train-test sessions is the desired result. This method is computationally expensive and it has often been reserved for problems where relatively small sample data is avaialbe. Actually Leave-one-out is a special example of k-fold method.

All the above methods are also called *cross-validation* method as unoberved examples are used to test or validate the performance of induced knowledge. (Weiss & Kulikowski 1990) is a good reference about experimention with regard to various learning programs.

Before going on to the usage of the program, some concepts need clarification. A *target concept* , T, is what a learning program is supposed to learn and usually can be specified as the set of examples this concept includes. The induced concept by a learning program is called *learned concept* , L, and it also represents a set of examples which meet conditions of the learned concept. By *error of omission* is meant an example which is covered by the target concept but is not covered (missed) by the learned concept. Its rate is defined as (|T| - |L|)/|T|. By *error of commission* is meant an example which is not covered by the definition of a target concept but is covered by the learned concept. Its rate is defined as (|C| - |T|) divided by the number of examples in the testing data which are not a member of the target concept.

To exhibit the performance improvement of a learning program when more examples are avialable, we use the term *stage*. A *s* stage train-test process is one in which the avaialbe training examples are divided into equal-sized *s* subsets and in the first stage, only the first subset is used for training and testing according to a selected evaluation method and with one more subset included in later stages.

To get a statistically solid results, usually a learning program is applied to the same problem over many times (i.e., *runs*) and the training and testing sets in each run are formed by randomly selecting examples .

3 USAGE OF EPE

This program automatically calls an AQ learning program (named as aq.run) and runs it over the given data according to user-specified methods. Be sure to make this AQ program is placed in the same directory as this program is.

The user has to put all the data in a AQ-style file (whichever name he/she likes) (Wnek et al., 1995). In order to get error rates of omission and commission, the test parameter in the input file must contain "mc" (Wnek et al., 1995).

The syntax of running this program is:

epe -f input_filename **[-rn1] [-mn2] [-pn3] [-sn4] [-o** output_filename]

All the arguments can be in any sequence. The meaning of them are the following:

input_filename: the file containing training examples.
n1: number of runs and the default is 1.
n2: a number representing evaluation method: 1, Hold-out; 2, k-fold; 3, Leave-one-out. The default is Holdout.
n3: a paramter used only in Hold-out and k-fold. In Hold-out, it is the percentage of examples for training; in k-fold, k value. The default is 70 for the Hold-out method.
n4: number of stages and the default is 1.
output_filename: the file containing a summary output from this program and the default file is epe.out.

During the runing of this program, the program genereates a series of files containing intermediate results corresponding to each application of AQ to each data division. The file names are in the format epeAQs**N1**n**N2**.out where N1 represents the number of stage and N2 the sequential number of running aq.run.

3.1 Illustration

We are going to use wind bracing data for illustration (Szczepanik et al., 1995). (See files ex.holdout, ex.kfold and ex.leave accampanying this software).

If a user wants to run aq.run over file ex.holdout in Hold-out mode for three runs, using 66% of the data for training and putting output in a file called ex1, he/she can type:

epe -f ex.holdout -r3 -m1 -p66 -s3 -o ex1.

The following will be output on screen:

```
═══════════ An Easy Performance Evaluation for AQ ═══════════

input file  = [ex.holdout]
#runs       = [3]
method      = [1, Holdout]
parameter   = [66% events for training]
#stages     = [3]
output file = [ex1]

═══════════════════ Run 1 ═══════════════════
Running AQ at stage 1 ...
Running AQ at stage 2 ...
Running AQ at stage 3 ...

═══════════════════ Run 2 ═══════════════════
Running AQ at stage 1 ...
Running AQ at stage 2 ...
Running AQ at stage 3 ...

═══════════════════ Run 3 ═══════════════════
Running AQ at stage 1 ...
Running AQ at stage 2 ...
Running AQ at stage 3 ...


═══════════════════════════════════════════════
═══════════ Summary of 3 Runs of AQ ═══════════
═══════════════════════════════════════════════
                                Error %
Stage              **  overall  commission  omission
  1                    17.000      6.920      20.148
  2                    13.333      5.357      29.774
  3                    12.333      4.894      24.517
```

Note in the above, "overall error" equals 100 minus the predcative accuracy.

In file ex1, a detailed summary containing the results of each run is stored:

```
═══════════ An Easy Performance Evaluation for AQ ═══════════

═══════════════════ Run 1 ═══════════════════
        ---------------------------------------
          3-stage Holdout results with 66% training
        ---------------------------------------
                                  Error %
Stage  #training  #testing  **  Overall  commission  omission
```

1	74	37	16.000	6.471	32.629
2	147	75	11.000	4.246	36.411
3	220	115	17.000	7.143	39.503

Run 2

3-stage Holdout results with 66% training

				Error %		
Stage	#training	#testing	**	Overall	commission	omission
1	74	37		11.000	4.065	9.412
2	147	75		16.000	6.475	38.850
3	220	115		10.000	3.874	9.519

Run 3

3-stage Holdout results with 66% training

				Error %		
Stage	#training	#testing	**	Overall	commission	omission
1	74	37		24.000	10.223	18.403
2	147	75		13.000	5.349	14.062
3	220	115		10.000	3.667	24.530

Summary of 3 Runs of AQ

		Error %		
Stage	**	Overall	commission	omission
1		17.000	6.920	20.148
2		13.333	5.357	29.774
3		12.333	4.894	24.517

To run aq.run over file ex.kfold over 3 runs in 4-fold mode and 2 stages and stroe the results in the default file epe.out, type:

epe -f ex.kfold -m2 -r3 -p4 -s2

On screen, the following is displayed

An Easy Performance Evaluation for AQ

input file = [ex.kfold]
#runs = [3]
method = [2, k-fold]
parameter = [4-fold]
#stages = [2]
output file = [epe.out]

Run 1

Running AQ at stage 1, fold No. 1 is taken as testing set...
Running AQ at stage 1, fold No. 2 is taken as testing set...
Running AQ at stage 1, fold No. 3 is taken as testing set...

```
Running AQ at stage 1, fold No. 4 is taken as testing set...
Running AQ at stage 2, fold No. 1 is taken as testing set...
Running AQ at stage 2, fold No. 2 is taken as testing set...
Running AQ at stage 2, fold No. 3 is taken as testing set...
Running AQ at stage 2, fold No. 4 is taken as testing set...

=========================== Run 2 ===========================
Running AQ at stage 1, fold No. 1 is taken as testing set...
Running AQ at stage 1, fold No. 2 is taken as testing set...
Running AQ at stage 1, fold No. 3 is taken as testing set...
Running AQ at stage 1, fold No. 4 is taken as testing set...
Running AQ at stage 2, fold No. 1 is taken as testing set...
Running AQ at stage 2, fold No. 2 is taken as testing set...
Running AQ at stage 2, fold No. 3 is taken as testing set...
Running AQ at stage 2, fold No. 4 is taken as testing set...

=========================== Run 3 ===========================
Running AQ at stage 1, fold No. 1 is taken as testing set...
Running AQ at stage 1, fold No. 2 is taken as testing set...
Running AQ at stage 1, fold No. 3 is taken as testing set...
Running AQ at stage 1, fold No. 4 is taken as testing set...
Running AQ at stage 2, fold No. 1 is taken as testing set...
Running AQ at stage 2, fold No. 2 is taken as testing set...
Running AQ at stage 2, fold No. 3 is taken as testing set...
Running AQ at stage 2, fold No. 4 is taken as testing set...

=============================================================
=========== Summary of 3 Runs of AQ ===========
=============================================================

                              Error %
Stage          ** overall  commission  omission
  1              15.667      6.722      21.097
  2              10.750      4.445      14.247
```

The following is stored in the default file **epe.out**:

```
========== An Easy Performance Evaluation for AQ ==========

=========================== Run 1 ===========================
        ------------------------------------
              2-stage 4-fold results
        ------------------------------------
                                  Error %
Stage  #training  #testing  ** overall  commission  omission
  1      126       41+       19.250      8.256      28.929
  2      252       83+       10.250      4.204      11.703

=========================== Run 2 ===========================
        ------------------------------------
              2-stage 4-fold results
        ------------------------------------
                                  Error %
```

Stage	#training	#testing	**	overall	commission	omission
1	126	41+		15.250	6.754	15.710
2	252	83+		10.500	4.394	14.569

Run 3

2-stage 4-fold results

				Error %		
Stage	#training	#testing	**	Overall	commission	omission
1	126	41+		12.500	5.155	18.651
2	252	83+		11.500	4.737	16.469

Summary of 3 Runs of AQ

		Error %		
Stage	**	overall	commission	omission
1		15.667	6.722	21.097
2		10.750	4.445	14.247

In each run, all the examples are randomly reorganized.

The same process happens to running this program in Leave-one-out mode.

REFERENCES

Michalski, R., Mozetic, I., Hong, J., and Lavrac, N. The multi-purpose increment al learning system AQ15 and its testing application to three medical domains. In AAAI-86, volume 2, pages 1041--1045, Ca. Kaufmann, 1986.

Clark, P. and Niblett, T. The CN2 induction algorithm. Machine Learning Journal, 3(4):261--283, 1989.

Quinlan, J.R., "C4.5: Programs for Machine Learning", Morgan Kaufmann, Los Altos, California, 1993.

Szczepanik, W., Arciszewski, T. and Wnek, J., "Empirical Performance Comparison of Two Symbolic Learning Systems Based On Selective And Constructive Induction," Proceedings of the IJCAI-95 Workshop on Machine Learning in Engineering, Montreal, Canada, August, 1995.

Weiss, S.M., & Kulikowski, C.A., *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning and expert systems*, Morgan Kaufmann, San Mateo, CA., 1992.

Wnek, J., & Kaufman, K., & Bloedorn, E., & Michalski, R.S., "Inductive learning system AQ15c: the method and user's guide", Reports of the Machine Learning and Inference Laboratory, MLI 95-4, George Mason University, Fairfax, VA., 1995.