Incremental Learning with Partial Instance Memory

Marcus A. Maloof[1] and Ryszard S. Michalski[2]

[1] Department of Computer Science, Georgetown University,
Washington, DC 20057, USA,
maloof@cs.georgetown.edu
[2] Machine Learning and Inference Laboratory, School of Computational Sciences,
George Mason University, Fairfax, VA 22030, USA,
michalski@mli.gmu.edu
Also, Institute of Computer Science, Polish Academy of Sciences, Warsaw

Abstract. Agents that learn on-line with partial instance memory reserve some of the previously encountered examples for use in future training episodes. We extend our previous work by combining our method for selecting extreme examples with two incremental learning algorithms, AQ11 and GEM. Using these new systems, AQ11-PM and GEM-PM, and the task computer intrusion detection, we conducted a lesion study to analyze trade-offs in performance. Results showed that, although our partial-memory model decreased predictive accuracy by 2%, it also decreased memory requirements by 75%, learning time by 75%, and in some cases, concept complexity by 10%, an outcome consistent with earlier results using our partial-memory method and batch learning.

1 Introduction

On-line learning systems with *partial instance memory* select and maintain a portion of the training examples from an input stream, using them for future training episodes. Depending on the task at hand and the goals of the learner, certain examples may be of higher utility than others, and in previous work [1], we selected *extreme examples* from the boundaries of induced concept descriptions under the assumption that such examples enforce, map, and strengthen these boundaries. To evaluate our method, we built an experimental system called AQ-PM that operates in a *temporal-batch* fashion: It processes training examples over time, combines them with the current set of extreme examples, induces a new set of concept descriptions, and selects a new set of extreme examples.

In this paper, we extend this work by combining our method for selecting extreme examples with two incremental learning algorithms: AQ11 [2] and GEM [3]. Although both are incremental, AQ11 discards its training examples, whereas GEM keeps all of its. Using these methods, we built two experimental systems, AQ11-PM and GEM-PM, and applied them to a variety of problems. Here, we present results for the problem of computer intrusion detection, one task from our earlier investigation of AQ-PM [1].

M.-S. Hacid et al. (Eds.): ISMIS 2002, LNAI 2366, pp. 16–27, 2002.

By conducting a lesion study, we examined the trade-offs between predictive accuracy, examples held in memory during learning, concept complexity, and learning time. Results mirror those of our previous inquiry [1]: By selecting and using examples from the boundaries of concept descriptions, learners can decrease memory requirements, concept complexity, and learning time at the expense of predictive accuracy.

In the following sections, we describe more fully the notion of partial instance memory and briefly examine past, related work. In the third section, we present the learning algorithms and the experimental systems that we developed. In Section 4, we discuss the data set used for evaluation, which we follow with descriptions of our experimental design and the experimental results. After analyzing these empirical findings, we conclude with a discussion of future directions.

2 Partial-Memory Learning

Agents that learn on-line will have potentially two types of memory—two types of concern to us—memory for storing concept descriptions and memory for storing instances. Not surprisingly, on-line learners vary widely in their use of these two types of memory.

Focusing on instance memory, IB1 [4], for example, is an instance-based method that stores all previously encountered training cases; thus, we call it a learner with *full instance memory* [3]. A related system, IB2 [4], stores only those instances that it misclassifies. It is an example of a system with partial instance memory. Finally, some on-line systems learn from and then discard new instances [29]. STAGGER [5] and WINNOW [6] are examples of such learners, ones with *no instance memory*.

Of concern here are systems with partial instance memory, which we will henceforth refer to as simply "partial-memory systems." An important issue is how such learners select examples from the input stream. One scheme is to select and store *prototypical examples* or *representative examples* [7]. Another approach is to remember a consecutive sequence of examples over a fixed [8] or changing window of time [9]. Yet another is to keep those examples that lie on or near the boundaries of concept descriptions [1].

In Figure 1, we present an general algorithm for incremental learning with partial instance memory. Input to the algorithm (line 1) is some number of data sets, one for each time step. While we are assuming that time is discrete, we place no restrictions of the number of examples present in each set: There could be none, there could be one, there could be several. This issue is important because it lets the learner track the passage of time. It also lets the learner operate at different time scales.

We assume that the learner begins with no concepts in concept memory and no instances in partial memory (lines 2 and 3). During the first time step (line 4, $t = 1$), assuming there are data (line 5), the learner operates in batch mode, and since there are no concepts, all of the examples in the data set are treated as if misclassified (line 6). Since partial memory is empty, the training set contains