Learn++.MT: A New Approach to Incremental Learning

Michael Muhlbaier, Apostolos Topalis, and Robi Polikar
Rowan University, Electrical and Computer Engineering Department
201 Mullica Hill Rd., Glassboro, NJ 08028, USA
{muhl1565,topa4536}@students.rowan.edu, polikar@rowan.edu

Abstract. An ensemble of classifiers based algorithm, Learn++, was recently introduced that is capable of incrementally learning new information from data-sets that consecutively become available, even if the new data introduce additional classes that were not formerly seen. The algorithm does not require access to previously used datasets, yet it is capable of largely retaining the previously acquired knowledge. However, Learn++ suffers from the inherent “out-voting” problem when asked to learn new classes, which causes it to generate an unnecessarily large number of classifiers. This paper proposes a modified version of this algorithm, called Learn++.MT that not only reduces the number of classifiers generated, but also provides performance improvements. The out-voting problem, the new algorithm and its promising results on two benchmark datasets as well as on one real world application are presented.

1 Introduction

It is well known that the amount of training data available and how well the data represent the underlying distribution are of paramount importance for an automated classifier’s satisfactory performance. For many applications of practical interest, obtaining such adequate and representative data is often expensive, tedious, and time consuming. Consequently, it is not uncommon for the entire data to be obtained in installments, over a period of time. Such scenarios require a classifier to be trained and incrementally updated – as new data become available – where the classifier needs to learn the novel information provided by the new data without forgetting the knowledge previously acquired from the data seen earlier. This raises the so-called stability-plasticity dilemma [1]: a completely stable classifier can retain knowledge, but cannot learn new information, whereas a completely plastic classifier can instantly learn new information, but cannot retain previous knowledge. Many popular classifiers, such as the ubiquitous multilayer perceptron (MLP) or the radial basis function networks, are not structurally suitable for incremental learning, since they are “completely stable” classifiers. The approach generally followed for learning from new data involves discarding the existing classifier, combining the old and the new data and training a new classifier from scratch using the aggregate data. This causes the previously learned information to be lost, a phenomenon known as catastrophic forgetting [2]. Furthermore, training with the combined data may not even be feasible, if the previously used data are lost, corrupted, prohibitively large, or otherwise unavailable.
We have recently introduced an algorithm, called Learn++, capable of learning incrementally, even under hostile learning conditions: not only does Learn++ assume the previous data to be no longer available, but it also allows additional classes to be introduced with new data, while retaining the previously acquired knowledge.

Learn++ is an ensemble approach, inspired primarily by the AdaBoost algorithm. Similar to AdaBoost, Learn++ also creates an ensemble of (weak) classifiers, each trained on a subset of the current training dataset, and later combined through weighted majority voting. Training instances for each classifier are drawn from an iteratively updated distribution. The main difference is that the distribution update rule in AdaBoost is based on the performance of the previous hypothesis [3], which focuses the algorithm on difficult instances, whereas that of Learn++ is based on the performance of the entire ensemble [4], which focuses this algorithm on instances that carry novel information. This distinction gives Learn++ the ability to learn new data, even when previously unseen classes are introduced. As new data arrive, Learn++ generates additional classifiers, until the ensemble learns the novel information. Since no classifier is discarded, previously acquired knowledge is retained. Other approaches suggested for incremental learning, a bibliography of ensemble systems and their applications can be found in and within the references of [4 ~9].

As reported in [4,5], Learn++ works rather well on a variety of real world problems, though there is much room for improvement. An issue of concern is the relatively large number of classifiers required for learning instances coming from a new class. This is because, when a new dataset introduces a previously unseen class, new classifiers are trained to learn the new class; however, the existing classifiers continue to misclassify instances from the new class. Therefore, the decisions of latter classifiers that recognize the new class are out-voted by the previous classifiers that do not recognize the new class, until a sufficient number of new classifiers are generated that recognize the new class. This leads to classifier proliferation.

In this contribution, we first describe the out-voting problem associated with the original Learn++, propose a modified version of the algorithm to address this issue, and present some preliminary simulation results on three benchmark datasets.

2 Learn++.MT

In ensemble approaches that use a voting mechanism for combining classifier outputs, each classifier votes on the class it predicts [10, 11]. The final classification is then determined as the class that receives the highest total vote from all classifiers. Learn++ uses weighted majority voting [12], where each classifier receives a voting weight based on its training performance. This works well in practice for most applications. However, for incremental learning problems that involve introduction of new classes, the voting scheme proves to be unfair towards the newly introduced class: since none of the previously generated classifiers can pick the new class, a relatively large number of new classifiers that recognize the new class are needed, so that their total weight can out-vote the first batch of classifiers on instances of the new class. This in return populates the ensemble with an unnecessarily large number of classifiers. Learn++.MT is specifically designed to address the classifier proliferation issue. The novelty in Learn++.MT is the way by which the voting weights are determined. Learn++.MT also uses a set of voting weights based on the classifiers’ performances,