IRACST – Engineering Science and Technology: An International Journal (ESTIJ), ISSN: 2250-3498
Vol.2, No.1, 2012

A Study of Meta-Learning in Ensemble Based Classifier

[1]Dr.M. Usha Rani, [2]G.T. Prasanna Kumari
[1]Head & Associate Professor, Dept. of Computer Science, Sri Padmavati Mahila Visva Vidyalayam, Tirupati -517502, AP, INDIA.
[2]Asso.Prof, Dept. of CSE&IT, Gokula Krishna College of Engg., Sullurpet-524121, AP, INDIA

***Abstract*--The idea of ensemble methodology is to build a predictive model by integrating multiple models. It is well-known that ensemble methods can be used for improving prediction performance. Researchers from various disciplines such as statistics and AI considered the use of ensemble methodology. Meta-learning is a technique that seeks to compute higher-level classifiers (or classification models), called meta-classifiers, that integrate in some principled fashion multiple classifiers computed separately over different databases. This study, describes meta-learning and presents Stacking, SCANN (Stacking, Correspondence Analysis and Nearest Neighbor), Arbiter Trees, Combiner Trees, Grading and the JAM system (Java Agents for Meta-learning).**

Keywords - Ensemble; Meta-learning; Stacking; SCANN; Arbiter Trees; Combiner Trees; Grading; JAM

I. INTRODUCTION

Many researchers have investigated the technique of combining the predictions of multiple classifiers to produce a single classifier (Breiman, 1996c; Clemen, 1989; Perrone, 1993; Wolpert, 1992). The resulting classifier (hereafter referred to as an ensemble) is generally more accurate than any of the individual classifiers making up the ensemble. Both theoretical (Hansen & Salamon, 1990; Krogh & Vedelsby, 1995) and empirical (Hashem, 1997; Opitz & Shavlik, 1996a, 1996b) research has demonstrated that a good ensemble is one where the individual classifiers in the ensemble are both accurate and make their errors on different parts of the input space.

The main idea behind the ensemble methodology is to weigh several individual classifiers, and combine them in order to obtain a classifier that outperforms every one of them. In fact, human being tends to seek several opinions before making any important decision. We weight the individual opinions, and combine them to reach our final decision (Polikar 2006).

James Michael Surowiecki, an American financial journalist, published in 2004 the book "The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations". Surowiecki argues, that under certain controlled conditions, the aggregation of information from several sources, results in decisions that are often superior to those that could have been made by any single individual—even experts.

There are two main methods for combining the base-classifiers' outputs: weighting methods and meta-learning methods. Weighting methods are useful if the base-classifiers perform the same task and have comparable success. Meta-learning methods are best suited for cases in which certain classifiers consistently correctly classify, or consistently misclassify, certain instances.

II. META-LEARNING

Meta-learning is itself a "learning" process. Meta-learning is about learning from learned knowledge. The idea is to execute a number of concept learning processes on a number of data subsets, and combine their collective results through an extra level of learning. A graphical representation of meta-learning three different classifiers is depicted in Figure 1. In this figure, two classifiers are derived from the same data set (either from different samples or from different learning algorithms, or both) while the third is induced from a separate set. The meta-learning algorithm combines the three classifiers into an ensemble meta-classifier by "learning" how they predict, i.e., by observing their input/output behavior.

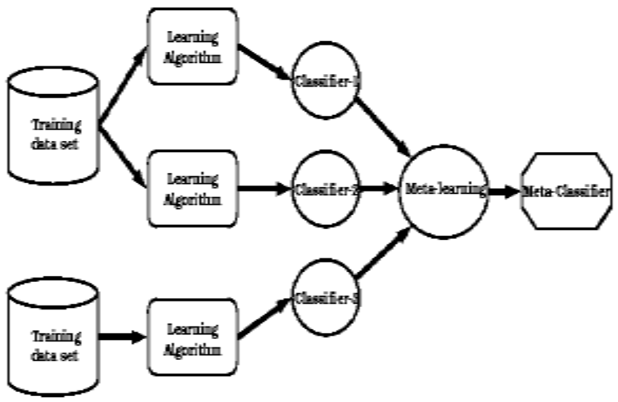

Figure 1: Meta-learning process

Meta-learning is a technique recently developed that deals with the problem of computing a "global" classifier from large and inherently distributed databases.

Meta-learning aims to compute a number of independent classifiers (concepts or models) by applying learning programs to a collection of independent and inherently distributed databases in parallel. The "base classifiers" computed are then collected and combined by another learning process. Here meta-learning seeks to compute a "meta-classifier" that integrates in some principled fashion the separately learned classifiers to boost overall predictive accuracy.

III. META-LEARNING METHODS

Meta-learning means learning from the classifiers produced by the inducers and from the classifications of these classifiers on training data. The five methods are stacking, SCANN, arbiter trees, combiner trees and grading aim to improve efficiency and scalability by executing a number of learning processes on a number of data subsets in parallel and by combining the collective results. The following sections describe the most well-known meta-combination methods.

A. Stacking

Stacked generalization (or stacking) (Wolpert 1992) is a different way of combining multiple models, that introduces the concept of a meta learner. Stacking may be (and normally is) used to combine models of different types. The procedure is as follows:

1) Split the training set into two disjoint sets.
2) Train several base learners on the first part.
3) Test the base learners on the second part.
4) Using the predictions from 3) as the inputs, and the correct responses as the outputs, train a higher level learner.

Note that steps 1) to 3) are the same as cross-validation, but instead of using a winner-takes-all approach, the base learners are combined, possibly non-linearly. Stacking is an ensemble technique that uses a meta-learner for determining which classifiers are reliable and which are not. Stacking is usually employed to combine models built by different inducers. The idea is to create a meta-dataset containing a tuple for each tuple in the original dataset. However, instead of using the original input attributes, Stacking uses the classifications predicted by the base-classifiers as the input attributes. The target-attribute remains as in the original training set. A test instance is first classified by each of the base classifiers. These classifications are fed into a meta-level training set to produce a meta-classifier. The meta-classifier that has been produced combines the different predictions into a final prediction.

In order to avoid over-fitting of the meta-classifier, the instances used for training the base-classifiers shouldn't be used to train the meta-classifier. Thus the original dataset should be partitioned into two subsets. The first subset is reserved to form the meta-dataset while the second subset is used to build the base-level classifiers. Consequently, the meta-classifier predictions reflect the true performance of baselevel learning algorithms.

Seewald (2003) presented strong empirical evidence that Stacking in the extension proposed by Ting and Witten (1999) performs worse on multi-class than on two-class datasets. This was true for all but one meta-learner that he investigated. The explanation given was that when the dataset has a higher number of classes, the dimensionality of the meta-level data is proportionally increased. This higher dimensionality makes it harder for meta-learners to induce good models, since there are more features to be considered.

The increased dimensionality has two more drawbacks. First, it increases the training time of the meta-classifier. For many inducers this problem is acute. Second, it also increases the amount of memory which is used in the training process. This could lead to insufficient resources, and limit the number of training cases (instances) from which an inducer can learn, thus damaging the ensemble's accuracy.

B. SCANN

The SCANN (Stacking, Correspondence Analysis and Nearest Neighbor) combining method (Merz 1999) uses the strategies of stacking and correspondence analysis. Correspondence analysis is a method for geometrically modeling the relationship between the rows and columns of a matrix whose entries are categorical. In this context Correspondence Analysis is used to explore the relationship between the training examples and their classification by a collection of classifiers. A nearest neighbor method is then applied to classify unseen examples. Here, each possible class is assigned coordinates in the space derived by Correspondence Analysis. Unclassified examples are mapped into the new space, and the class label corresponding to the closest class point is assigned to the example.

C. Arbiter Trees

According to Chan and Stolfo's approach, an arbiter tree is built in a bottom-up fashion. Initially, the training set is randomly partitioned into k disjoint subsets. The arbiter isinduced from a pair of classifiers and recursively a new arbiter is induced from the output of two arbiters. Consequently for k classifiers, there are log2 (k) levels in the generated arbiter tree.

The creation of the arbiter is performed as follows. For each pair of classifiers, the union of their training dataset is classified by the two classifiers. A selection rule compares the classifications of the two classifiers and selects instances from the union set to form the training set for the arbiter. The arbiter is induced from this set with the same learning algorithm used in the base level. The purpose of the arbiter is to provide an alternate classification when the base classifiers present diverse classifications. This arbiter, together with an arbitration rule, decides on a final classification outcome, based upon the base predictions. Figure 2 shows how the final classification is selected based on the classification of two base classifiers and a single arbiter.

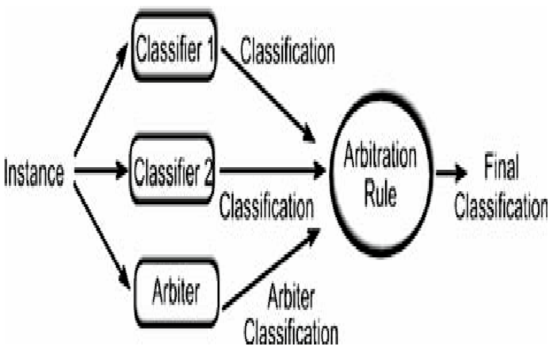

Figure 2: A prediction from two base classifiers and a single arbiter

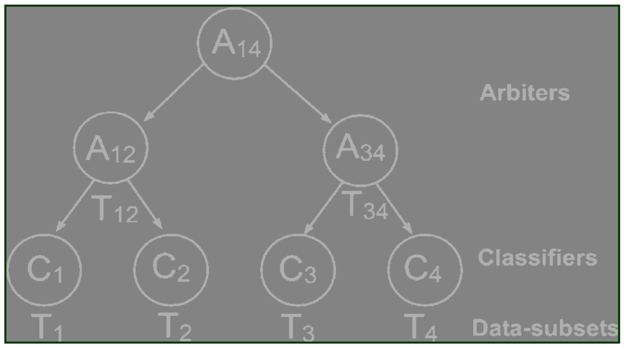

Figure 3: A sample Aribiter Tree

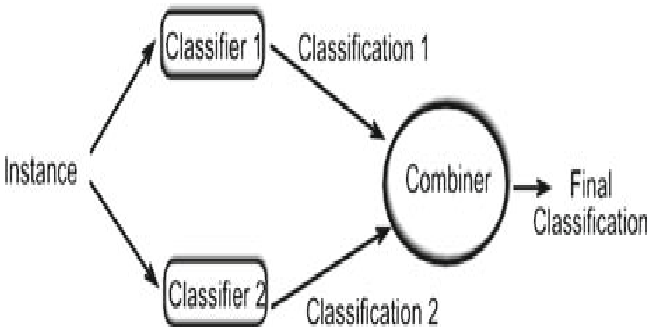

Figure 4: A prediction from two base classifiers and a single combiner

The process of forming the union of data subsets; classifying it using a pair of arbiter trees; comparing the classifications; forming a training set; training the arbiter; and picking one of the predictions, is recursively performed until the root arbiter is formed. Figure 3 illustrate an arbiter tree created for k = 4. T1 - T4 are the initial four training datasets from which four classifiers M1 - M4 are generated concurrently.

T12 and T34 are the training sets generated by the rule selection from which arbiters are produced. A12 and A34 are the two arbiters. Similarly, T14 and A14 (root arbiter) are generated and the arbiter tree is completed.

There are several schemes for arbiter trees; each is characterized by a different selection rule. Here are three versions of selection rules:
1) Only instances with classifications that disagree are chosen (group 1).
2) Like group 1 defined above, plus instances where their classifications agree but are incorrect (group 2).
3) Like groups 1 and 2 defined above, plus instances that have the same correct classifications (group 3).

Of the two versions of arbitration rules that have been implemented, each corresponds to the selection rule used for generating the training data at that level:
1) For selection rule 1 and 2, a final classification is made by a majority vote of the classifications of the two lower levels and the arbiter's own classification, with preference given to the latter.
2) For selection rule 3, if the classifications of the two lower levels are not equal, the classification made by the sub-arbiter based on the first group is chosen. In case this is not true and the classification of the sub-arbiter constructed on the third group equals those of the lower levels, then this is the chosen classification. In any other case, the classification of the sub-arbiter constructed on the second group is chosen. In fact it is possible to achieve the same accuracy level as in the single mode applied to the entire dataset but with less time and memory requirements (Chan and Stolfo). More specifically it has been shown that this meta-learning strategy required only around 30% of the memory used by the single model case. This last fact, combined with the independent nature of the various learning processes, make this method robust and effective for massive amounts of data. Nevertheless, the accuracy level depends on several factors such as the distribution of the data among the subsets and the pairing scheme of learned classifiers and arbiters in each level. The decision regarding any of these issues may influence performance, but the optimal decisions are not necessarily known in

advance, nor initially set by the algorithm.

D) Combiner trees

The way combiner trees are generated is very similar to arbiter trees. Both are trained bottom-up. However, a combiner, instead of an arbiter, is placed in each non-leaf node of a combiner tree. In the combiner strategy, the classifications of the learned base classifiers form the basis of the meta-learner's training set. A composition rule determines the content of training examples from which a combiner (meta-classifier) will be generated. Figure 4 illustrates the result obtained from two base classifiers and a single combiner.

Two schemes for composition rules were proposed. The first one is the stacking scheme. The second is like stacking with the addition of the instance input attributes. It has been shown that the stacking scheme per se does not perform as well as the second scheme . Although there is information loss due to data partitioning, combiner trees can sustain the accuracy level achieved by a single classifier. In a few cases, the single classifier's accuracy was consistently exceeded.

E) Grading

This technique uses "graded" classifications as meta-level classes. The term "graded" is used in the sense of classifications that have been marked as correct or incorrect. The method transforms the classification made by the k different classifiers into k training sets by using the instances k times and attaching them to a new binary class in each occurrence. This class indicates whether the k-th classifier yielded a correct or incorrect classification, compared to the real class of the instance.

For every classifier, one meta-classifier is trained whose aim is to indicate when each classifier tends to misclassify a given instance. At classification time, each classifier tries to classify an unlabeled instance. The final classification is obtained by combining the outputs of the classifiers that are recognized as correct by the meta-classification schemes. Grading is considered to be generalization of cross-validation selection, which partitions the training data into k sets, builds k - 1 classifiers by dropping one set at a time and then uses it to find a misclassification rate. Finally, the procedure simply chooses the classifier corresponding to the subset with the smallest misclassification. Grading tries to make this decision separately for each and every instance by using only those classifiers that are predicted to classify that instance correctly. The main difference between grading and combiners (or stacking) is that the former does not change the instance attributes by replacing them with class predictions or class probabilities (or adding them to it). Instead it modifies the class values. Furthermore, in grading several sets of meta-data are created, one for each base classifier. Several meta-level classifiers are learned from those sets.

The main difference between grading and arbiters is that arbiters use information about the disagreements of classifiers for selecting a training set; grading uses disagreement with the target function to produce a new training set.

IV. BENEFITS OF META-LEARNING

Meta-learning improves efficiency by executing in parallel the base-learning processes (each implemented as a distinct serial program) on (possibly disjoint) subsets of the training data set (a data reduction technique). This approach has the advantage, first, of using the same serial code without the time-consuming process of parallelizing it, and second, of learning from small subsets of data that fit in main memory.

Meta-learning improves predictive performance by combining different learning systems each having different inductive bias (e.g representation, search heuristics, search space). By combining separately learned concepts, meta-learning is expected to derive a higher level learned model that explains a large database more accurately than any of the individual learners. Furthermore, meta-learning constitutes a scalable machine learning method since it can be generalized to hierarchical multi-level meta-learning.

SCANN are sophisticated and effective combining algorithms, but too computationally expensive to combine models within domains with many models and large data sets. SCANN, in fact, is cubic in the number of available models.

Meta-learning is particularly suitable for distributed data mining applications, such as fraud detection in financial information systems. Financial institutions today typically develop custom fraud detection systems targeted to their own asset bases. Recently though, banks have come to search for unified and global approaches that would also involve the periodic sharing with each other of information about attacks.

The key difficulties in this approach are: financial companies avoid sharing their data for a number of (competitive and legal) reasons; the databases that companies maintain on transaction behavior are huge and growing rapidly; real-time analysis is highly desirable to update models when new events are detected and easy distribution of models in a networked environment is essential to maintain up to date detection capability. Meta-learning is a general strategy that provides the means of learning how to combine and integrate a number of classifiers or models learned separately at different financial institutions. JAM allows financial institutions to share their models of fraudulent transactions that each computes separately, while not disclosing their own proprietary data. Next, we describe how meta-learning is incorporated in JAM.

V. THE JAM SYSTEM

First we focus on the components of the system and the overall architecture. Assuming that the data mining system comprises of several data sites, each with its own resources, databases, machine learning agents and meta-learning capabilities, we designed a protocol that allows the data sites to collaborate efficiently without hindering their progress. JAM is architected as a distributed computing construct that supports the launching of learning and meta-learning agents to distributed database sites.

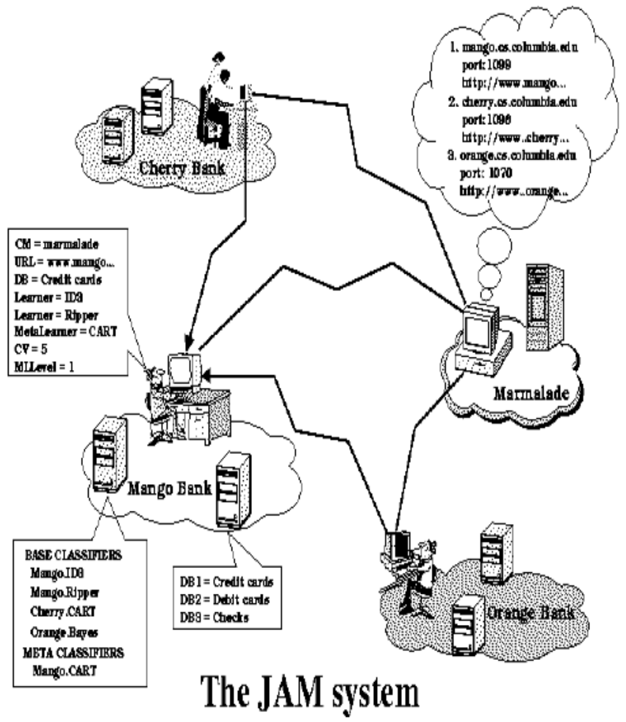

Figure 5: The architecture of the meta-learning system.

First, local or imported learning agents execute on the local database to compute the data site's local classifiers. Then, each data site may import (remote) classifiers from its peer data sites and combine these with its own local classifier using meta-learning agents. Again, the meta-learning agents can be either local or imported from other data sites. Finally, once the base and meta-classifiers are computed, the JAM system manages the execution of these modules to classify data sets of interest. These actions may take place at all data sites simultaneously and independently. The JAM system can thus be viewed as a coarse-grain parallel application where the constituent sites function autonomously and (occasionally) exchange classifiers with each other. The configuration of the distributed system is maintained by the Configuration Manager (CM), an independent server, much like a simple name server, that is responsible for keeping the state of the system up-to-date. The logical architecture of the JAM meta-learning system is presented in Figure 5. In this example, three JAM sites Orange, Mango and Strawberry exchange their base classifiers to share their local view of the learning task. The user of the data site controls the learning task by setting the parameters of the user configuration file, e.g. the algorithms to be used, the images to be used by the animation facility, the cross validation and folding parameters, etc.

VI. CONCLUSION

Ensemble methodology imitates our second nature to seek several opinions before making a crucial decision. The core principle is to weigh several individual pattern classifiers, and combine them in order to reach a classification that is better than the one obtained by each of them separately. Researchers from various disciplines such as pattern recognition, statistics, and machine learning have explored the use of ensemble methods since the late seventies. Given the growing interest in the field, it is not surprising that researchers and practitioners have a wide variety of methods at their disposal. An ensemble is largely characterized by the diversity generation mechanism and the choice of its combination procedure.

REFERENCES

[1] Arbel R, Rokach L (2006) Classifier evaluation under limited resources. Pattern Recognition Lett 27(14):1619–1631

[2] Pattern classification using ensemble methods by Lior Rokach

[3] Lior (2009) Ensemble Based Classifier

[4] Polikar R (2006) Ensemble based systems in decision making. IEEE Circuits Syst Mag :21-45

[5] Prodromidis AL, Stolfo SJ, Chan PK (1999) Effective and efficient pruning of metaclassifiers in a Distributed Data Mining System. Technical report CUCS-017-99, Columbia University

[6] Melville P, Mooney RJ (2003) Constructing diverse classifier ensembles using artificial training examples IJCAI 505-512

[7] Liu H, Mandvikar A, Mody J (2004) An empirical study of building compact ensembles. WAIM 622-627

[8] Kuncheva LI (2005) Diversity in multiple classifier systems (Editorial). Inf Fusion 6(1):3-4

[9] Kuncheva L (2005) Combining Pattern classifiers. Wiley Press, New York

Author's Profile

Dr.M.Usha Rani is an Associate Professor in the Department of Computer Science and HOD for MCA, Sri Padmavati Mahila Viswavidyalayam(SPMVV Womens' University), Tirupati. She did her Ph.D. in Computer Science in the area of Artificial Intelligence and Expert Systems. She is in teaching since 1992. She presented many papers at National and Internal Conferences and published articles in national & international journals. She also written 4 books like Data Mining - Applications: Opportunities and Challenges, Superficial Overview of Data Mining Tools, Data Warehousing & Data Mining and Intelligent Systems & Communications. She is guiding M.Phil. and Ph.D. in the areas like Artificial Intelligence, DataWarehousing and Data Mining, Computer Networks and Network Security etc.

Mrs G.T.Prasanna Kumari is an Associate Professor in the Department of Computer Science and Engineering, Gokula Krishna College of Engineering, Sullurpet. She is pursuing Ph.D., in Computer Science in the area of Distributed Data Mining Systems. She is in teaching since 1999. She presented papers at National and International Conferences and published articles.