Hybrid Learning Architectures for Advanced Data Mining:An Application to Binary Classification for Fraud Management

Steven H. Kim*, Sung Woo Shin**

Abstract

The task of classification permeates all walks of life, from business and economics to science and public policy. In this context, nonlinear techniques from artificial intelligence have often proven to be more effective than the methods of classical statistics.

The objective of knowledge discovery and data mining is to support decision making through the effective use of information. The automated approach to knowledge discovery is especially useful when dealing with large data sets or complex relationships. For many applications, automated software may find subtle patterns which escape the notice of manual analysis, or whose complexity exceeds the cognitive capabilities of humans.

This paper explores the utility of a collaborative learning approach involving integrated models in the preprocessing and postprocessing stages. For instance, a genetic algorithm effects feature-weight optimization in a preprocessing module. Moreover, an inductive tree, artificial neural network (ANN), and k-nearest neighbor (kNN) techniques serve as postprocessing modules. More specifically, the postprocessors act as second-order classifiers which determine the best first-order classifier on a case-by-case basis.

*) Graduate School of Management, Korea Advanced Institute of Science and Technology, Seoul, Korea

**) Samsung SDS, Seoul, Korea

In addition to the second-order models, a voting scheme is investigated as a simple, but efficient, postprocessing model. The first-order models consist of statistical and machine learning models such as logistic regression (logit), multivariate discriminant analysis (MDA), ANN, and kNN. The genetic algorithm, inductive decision tree, and voting scheme act as kernel modules for collaborative learning. These ideas are explored against the background of a practical application relating to financial fraud management which exemplifies a binary classification problem.

Key words: Collaborative learning, data mining, feature weighting, classification, fraud management

1. INTRODUCTION

1.1 Purpose

The objective of knowledge discovery and data mining (KDD) is to support decision making through the effective use of information. The practical aspect of KDD lies in the development of learning software to discover patterns, trends, or relations in databases [11].

The automated approach to knowledge discovery is especially useful when dealing with large data sets or complex relationships. For many applications, automated software may find subtle patterns which escape the notice of manual analysis, or whose complexity exceeds the cognitive capabilities of humans. Secondly, software for data mining may ensure the practicality of analyzing large data sets which would require an unjustifiable amount of manual effort. Lastly, automated learning tools may monitor complex environments such as production requirements on a continuous basis, updating their knowledge base immediately with the outbreak of new trends.

This paper explores the utility of an integrated approach relating to collaborative learning. The collaboration involves learning modules in the preprocessing and postprocessing stages in addition to the kernel model which serves as the foundation of the learning process. The approach is compared against individual techniques relating to statistical and machine learning models including logistic regression (logit), multivariate discriminant analysis (MDA), artificial neural network (ANN), k-nearest neighbor (kNN). The ideas are explored against the background of a practical application relating to financial fraud management in providing credit card services.

1.2 Background

Context. The past few decades have witnessed increasing interest in the development of software for automated learning. The tools have been applied widely to practical domains especially since the late 1980s.

To date, practical applications have tended to utilize single techniques in isolation. However, each tool has its advantages and drawbacks. For this reason, a collaborative learning approach to KDD offers the potential to take advantage of the strengths of various techniques while bypassing the limitations of each methodology.

Learning is a useful capability for dealing with unknown factors in current or future activities. One way to categorize learning systems takes the form of explicit versus implicit representation of knowledge. Explicit knowledge representation refers to the encoding of domain knowledge in terms and concepts readily comprehensible to the user. Examples of explicit representation lie in the Eurisko program [32] for discovering new concepts, or the learning classifier approach [14]. Explicit knowledge may be declarative, as in the statement of facts, or procedural, as in the encoding of instructions.

In contrast, implicit techniques refer to the generation of appropriate output based on input stimuli, without the direct representation of knowledge for processing the information. The implicit approach is reflected in neural network models, where information processing know-how is encoded in terms of multiplicative weights at nodes and on the arcs between pairs of nodes. A key advantage of the neural approach is that the system itself performs the bulk of the work for establishing the implicit knowledge. A second advantage lies in robustness, in that a neural network will degrade gracefully rather than catastrophically when its structure is modified in minor ways.

Another collaborative learning approach is to employ learning models in conjunction with statistical techniques. The incorporation of statistical tools including factor analysis, principal components analysis, and other multivariate methods as a preprocessor for the compaction of input dimensions enhances performances in certain situations while having little or no effect in others. Previous work on the impact of statistical methods has been reported in the literature [16, 22, 23, 31]. The current study investigated the effects of logistic regression and multivariate discriminant analysis in conjunction with machine learning techniques.

Explicit versus implicit models. A neural network offers many advantages such as robustness and graceful degradation [15]. However, most neural nets suffer from a number of limitations such as the need for long learning times. Another drawback lies in the implicit nature of the acquired knowledge, which cannot be explicitly communicated to a human decision maker [18, 19].

At lower levels of representation in the human brain, data is encoded in the form of voltage differences and structural changes, both mediated by chemical action. Yet, at some level of awareness, we are conscious of aggregate concepts, relationships, and abstractions. Hence a practical system should effectively couple the implicit and explicit forms of representing knowledge [44]. The framework and concepts for learning systems have been tailored to applications in various domains. In controlling dynamic systems, an effective approach is to implement a multistrategy scheme incorporating techniques such as nonlinear control, real-time sensing, and adaptive techniques [7, 8].

People take account of observations and utilize them for future decision making. Often the extrapolation to new situations is ad hoc, as in modifying a tactical plan from a card game to stock market investment. In other cases, the extrapolation is more formal and takes the form of inductive propositions such as formulas, principles, laws, or rules of thumb.

One pertinent methodology is found in case based reasoning (CBR). The key advantage of CBR lies in its affinity to human learning and the attendant ease of enhancing system performance. The knowledge in a particular domain can be stored in formats which are conventional for that domain. For instance, a knowledge base for failures in a communication network can store the information about previous malfunctions in the format used by human troubleshooters. This is in contrast to other knowledge level representations such as production rules, in which the system developer is required to extricate the pertinent decision rules used by a human.

The CBR methodology can be effective even if the knowledge base is imperfect. Certain techniques of automated learning, such as explanation-based learning, work well only if a strong domain theory exists. In contrast, CBR can use many examples to overcome the gaps in a weak domain theory while still taking advantage of the domain theory [37]. CBR can also be used when the descriptions of the cases, as well as the domain theory, are incomplete [47]. A further advantage of CBR is the relative ease of combining techniques with other approaches. An example of such compatibility is a system which uses case reasoning to solve problems whenever possible; otherwise it resorts to heuristics to decompose a problem into a simple one. The attempt to modify old solutions to the current context must take into account not only the basic features of the cases, but the relationships among their constituent parts. The mapping of relationships from one case to another represents the task of analogy.

To illustrate, consider a novel control problem which largely resembles a prior one in the case base. The use of the previous solution, however, would lead to sluggish response in the current application. This limitation could be addressed by introducing an extra component based on derivative control. But the revised design leads to a spasmodic control signal due to noise in the input signal. The latter difficulty can be resolved in various ways, such as inserting a moving-average smoothing unit immediately before the derivative controller. In this way, a process of iterative refinement can be employed to adapt an old solution to the new problem context.

A learning system, to be effective, must have the ability to draw on knowledge from diverse sources. The diversity can take the form of different formats for encoding knowledge as well as multiple modules within a single representation format [18]. The synergism of diverse formats is exemplified by the integration of production rules with the output of neural networks, or the interplay of case information with rules of thumb. The second major category of integration relates to the enhancement of performance through the fusion of new information with the old in an existing knowledge base. To illustrate the process of learning through knowledge integration, consider the diagnosis of failure in a sensor device. The causal relationships among the factors behind the problem may be represented in the form of a fault tree. The collective causal tree reflects an expansion of the reasoning capability of the system, including the ability to present the integrated knowledge to a human questioner [24]. In this way a systematic base of failure factors can be constructed from diverse cases without human intervention.

The comprehensive framework for learning systems can be adapted to the tasks of decision making. A global architecture for a predictive system is presented in [Figure-4]. Information from the environment is acquired through sensors and conveyed to a feature extractor as well as a human monitor. The input streams are processed by the predictive system in conjunction with dynamic system objectives and user decisions. The actions from the system, implemented through the activators, modify the external environment - such as the acquisition of a sensor reading - or the internal environment - such as the invocation of a contingency plan.

Induction refers to the generation of rules or the classification of objects through decision trees. The methodology generates new knowledge in the explicit form of rules or trees [39, 40]. However, the methodology can be cumbersome to use and has not been deployed as widely as it deserves.

Lazy learning algorithms. The class of lazy learning algorithms (LLAs) involves techniques which store precedent cases in memory with little or no preprocessing, then retrieve them on a selective basics as required by a problem situation [35, 50]. Perhaps a more accurate term than lazy learning would be delayed learning or real-time processing. LLAs have their roots in the nearest neighbor (NN) retrieval algorithms used in pattern recognition [4]. Other techniques in this category include the following approaches: k-nearest neighbor (kNN), case-based reasoning (CBR), memory-based reasoning, and instance-based learning.

Since the late 1980s, extensive research has been conducted on techniques for processing precedent cases including methods for indexing, retrieving, and adaptation to new cases [1, 27, 30, 43, 49]. This collective framework of techniques is known as case-based reasoning. Within the framework of CBR, the NN algorithm is the simplest version. In particular, the NN procedure retrieves the single precedent case in the database which most closely resembles the target problem at hand.

A generalization of NN is the k-nearest neighbor (kNN) algorithm, which calls the k closest precedents [4, 5, 34, 46, 50]. The NN and kNN algorithms are often reasonable and adequate methods for retrieving precedents when the features or attributes of the cases are orderly : no irrelevant or redundant features, no inter-feature correlation, and no noise [29, 38, 50].

For a lazy learning algorithm, the selection of appropriate cases relies on a similarity metric which takes into account the distance between pairs of cases in their state space of features. Consequently, the performance of the metric and the weighting of features are keys to the reasoning process [38]. In contrast to most other procedures, CBR is effective for applications involving weak domain knowledge; that is, complex fields where human expertise is unavailable or even nonexistent. In such situations, the need for automatic feature selection and weight optimization is particularly acute.

CBR may be regarded as a hallmark of the class of lazy learning algorithms. In particular, a kNN procedure represents a simplified CBR approach in which little or no processing is performed on the retrieved cases other than a weighted combination of the precedents.

Genetic Algorithms. A genetic algorithm (GA) is a procedure modeled after the processes of genetic evolution and population dynamics in natural selection. In essence, the procedure selects highly fit individuals and their chromosomes at random to yield offspring; within the new population, the unfit are eliminated and the fittest survive to contribute genetic material to the subsequent generation.

The seminal work on genetic algorithms dates from the mid-1970s [14]. Most applications of GAs involve the optimization of a performance function based on a domain of either continuous or discrete variables. Classical methods of optimization rely on the improvement of a single trajectory toward an optimum by computing the gradient at each step. In contrast, a GA promotes several solutions in parallel, and modifies them in a random fashion to obtain the subsequent iteration toward the optimum. The inherent parallelism and the advantage of directed random search permits the use of genetic algorithms for addressing computationally difficult problems even in the NP-hard category [13, 14]. In a GA, a chromosome is a sequence of symbols which represents an individual or candidate solution to the problem at hand. Often theses symbols represent numbers; the numbers in particular might be the binary digits 0 and 1 . The collection of individuals at each iteration is called the population. The optimization process at each iteration or generation involves selection followed by crossover and/or mutation.

More specifically, the individuals in a population are evaluated through a fitness function which measures their relative worth. The fittest individuals are then chosen for further processing. The concept of inheritance is implemented by selecting two fit individuals and crossing or mixing their chromosomes: this crossover operation involves slicing their chromosomes at random locations and recombining corresponding sections from previously distinct individuals. Another way to effect variation in the subsequent population takes the form of mutation: a random location is selected on a chromosome and its symbol is changed to another feasible value at random.

A particular sequence of symbols for a chromosome is called a genotype. At times two different genotypes may exhibit the same appearance and behavior; in that case they constitute a single phenotype. For instance, an offspring may inherit two genes for brown hair from its parents, thereby exhibiting brown hair. Its cousin may inherit a gene for black hair and another for blond, and thereby end up with brown hair as well. The two offspring have different genotypes but the same phenotype in the context of hair color.

On occasion, a population may reach a dead end in evolutionary terms. With insufficient variety in the common gene pool, the evolutionary process becomes congested and the population can attain only a locally optimal solution. To circumvent this possibility, mutation introduces a new aspect at random, thereby allowing the population to escape the local extremum and search other terrains for a global optimum. The rate of mutation must be high enough to avoid long periods of stagnation in the evolutionary process, but low enough to ensure a measure of stability; that is, providing the population with a chance to reach the local optimum before leaping into distant terrain.

The appropriate type and extent of mutation as well as crossover, will of course depend on the problem domain. Since the performance function is generally unknown in advance, the optimal configuration of the fitness function as well as the nature of crossover and mutation may be regarded as design issues which must be informed by detailed knowledge of the application area.

The basic structure of a GA is as follows, where P(t)denotes a population of candidate solutions to a given problem at generation t.

```
t = 0;
initialize P(t);
evaluate P(t);
while not (termination condition)
   begin
      t = t + 1;
      reproduce P(t) from P(t -1);
      recombine P(t);
      evaluate P(t);
   end;
```

Hybrid learning architectures. A hybrid or multistrategy approach to learning involves the interaction of two or more data mining techniques. One way to classify multistrategy learning lies in the following typology.

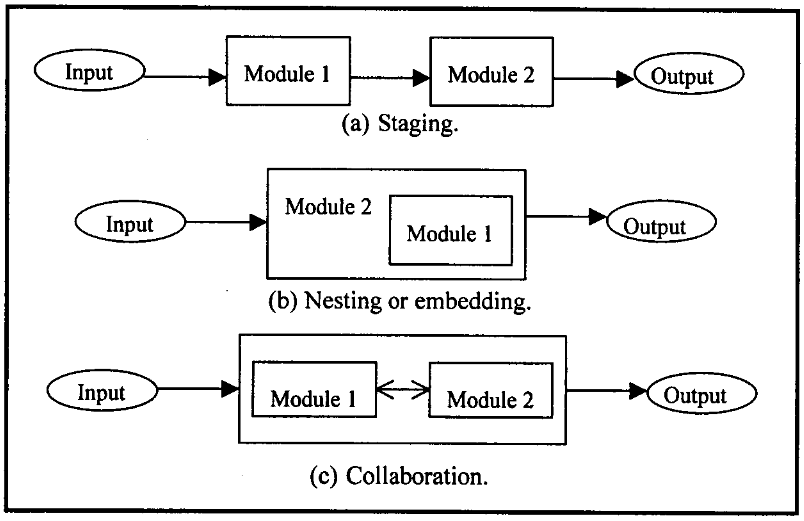
a. *Staging.* In this architecture, the output of one module enters a second module. The flow of information is unilateral: from the first module to the second.

b. *Nesting.* In this configuration, one technique is subsumed within another approach. For instance, a genetic algorithm serves to optimize the weights used in selecting precedents in case based reasoning.

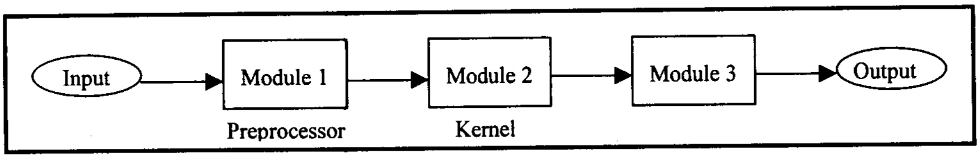
c. *Collaboration.* Two modules exchange information in a bilateral fashion. Where the collaborative architecture contains 3 or more modules, information flows may arise in mutilateral fashion.

The hybrid architectures are depicted in [Figure-1].

A staged architecture contains two or more modules. In that case, one of the modules may be viewed as the core or kernel component. The module prior to the kernel may then be regarded as the preprocessing unit. Similarly, the module lying posterior to the kernel can be regarded as a postprocessor. The arrangement is depicted in [Figure-2].



[Figure-1] A classification of hybrid or multistrategy architectures. Each module may be elementary or composite. For instance, Module 1 in a staged architecture may itself contain a nested module or even another staged architecture.



[Figure-2] Preprocessing and postprocessing modules. In a staged architecture with multiple modules, one of the components may be regarded as the main or kernel module. The component prior to the kernel, if one exists, may be regarded as a preprocessor. Similarly, the subsequent module may be viewed as a postprocessor.

Second-order learning. Second-order learning may be viewed as an example of a staged architecture. This approach refers to the selection of an elementary model based on its anticipated performance.1)

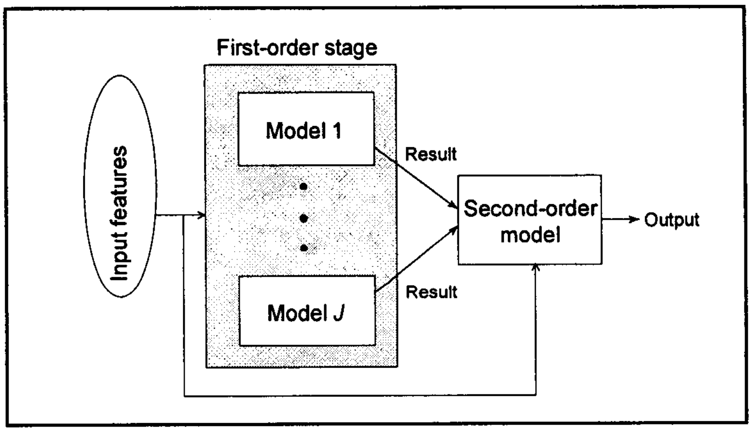
Second-order learning is a multistrategy approach to knowledge discovery. The first stage of the integrated architecture involves a basic set of learning models. The outputs from these first-order models enter a second stage, which determines which of the first-order models to use in each particular situation (typically specified by an input vector) [20]. The general procedure is depicted in [Figure-3].

In this paper, the elementary techniques representing first-order learning relate to statistical and machine learning techniques. Subsequently the second stage employs inductive reasoning, ANN and kNN to determine the conditions under which a particular model performs better than the other. Then the forecast from the superior method is selected on a case-by-case basis to determine the output of the overall system. In other words, the second stage serves as a metalevel predictive system to discern which of the elementary

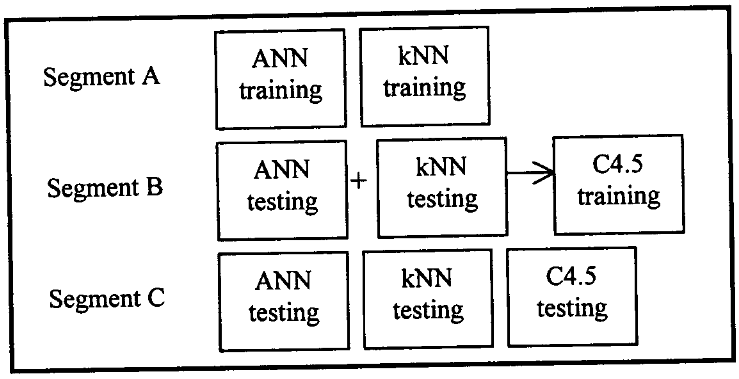
1) *A basic or elementary* model refers to one which utilizes only the input data. In other words, it is a "first-order" model. On the other hand, a second order model utilizes the output of a basic model in conjunction with the input data.

modules (ANN, kNN or Logit) will perform better.

In training and testing a second-order model, the dataset must be partitioned into three segments, [Figure-4] illustrates the approach when the first order modules consist of ANN and kNN, while the second-order model involves the C4.5 induction algorithm. The first segment (denoted "A" in the figure) is used to train the first-order models. The second segment ("B") serves to test the first-order models, and to use the results to train the second-order model. Finally, the third segment ("C") may be used to test each of the first- and second-order models.



[Figure-3] Schematic of the second-order learning architecture.



[Figure-4] Definition of partitioning of data sets to investigate the second-order learning.

Inductive reasoning. The C4.5 program is an inductive algorithm which produces a decision tree [39]. This procedure utilizes most of the concepts and tools employed by its predecessor, the ID3 algorithm [40]. The basic concept involves the use of the entropy metric from information theory to yield efficient decision trees. In addition, C4.5 employs a "split information" metric in order to avoid strong bias. A metric called a "gain ratio" is also used to construct decision trees of greater accuracy. Moreover C4.5 has other advantages compared to the ID3 algorithm, such as the ability handle continuous-valued attributes, and the generation of prior estimates of predictive accuracy.

Inverse of correlated errors. In practical systems involving competent models, it is common for two or more models to agree on the correct response. However, it is relatively unusual to find multiple models converging on the same incorrect result.

In nonlinear models, there is in general no guarantee of discovering the globally optimal model. On top of this theoretical limitation, there are the very practical constraints of limited datasets and processing time. For these reasons, it seems plausible to infer that some features - unknown to the human decision maker-of the dataset have led multiple models astray. Consequently, if cases from the untested dataset also lead the models down the same path in a consistent way, then this information is useful knowledge.

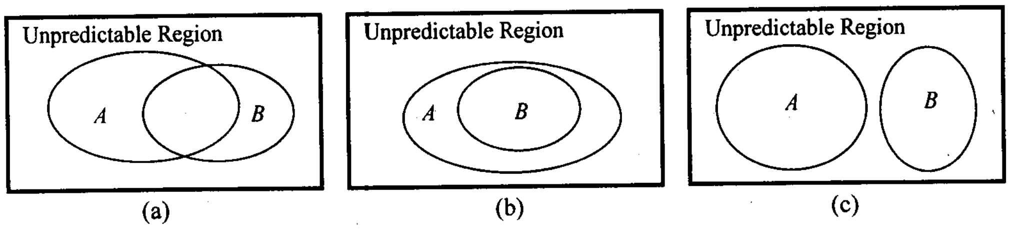
The collaborative model employed in this paper therefore utilizes the notion of correlated errors: if multiple models err in the same way on a particular case, then the correct response must lie in the complement among the potential set of responses.

· In the context of binary classification, the result is immediate. For instance, if multiple models yield a false positive in a particular type of situation, then the correct response should be a negative. In a sense, the reversal of a response on correlated errors represents a type of "negative voting".

Collaborative learning. Collaborative learning might be viewed as a multilateral version of second-order learning. In this paper, the basic techniques representing first-order learning related to logit, MDA, kNN and ANN. In addition, inductive reasoning, ANN, and kNN were used to select among the predictions from the elementary learning techniques

The collaborative learning model employed the method of error reversal on correlated mistakes. More specifically, if two out of the former elementary models produce correlated errors, the converse of their response is used for further analysis. The second-order learning stage served as a metalevel prediction system to determine the conditions under which one or more primary modules outperformed the others. Then the prediction from the superior method was selected on a case-by-case basis to determine the output of the overall system.

A genetic algorithm serves as a preprocessor by optimizing the weights of features for selecting cases to be used by the kNN classifier. A kNN model which employs the GA as a preprocessor is labeled GA-kNN. Similarly, an ANN model with a GA preprocessor is denoted GA-ANN. Since a GA module requires significant processing time, its fitness function ought to be a lazy-style learner similar to kNN. In this study, the GA-ANN model adapted kNN as the fitness function for the ANN classifier: much like previous work where kNN was used to approximate the counter propagation neural network [2].



[Figure-5] Potential regions of competence corresponding to two distinct classifiers. The symbol A denotes the region of competence for model A; and B for model B. Intuitively, the performance of a second-order learner should be the union of A and B. In diagram (b), model A is a superset of B; consequently the predictive performance of a second-order learner should not exceed that of model A. For a second-order learner, the situation in diagrams (a) and (c) are the most interesting. In practical situations, fully disjoint regions as depicted diagram (c) are not likely to occur. This study focused on second-order learning for the situation in diagram (a).

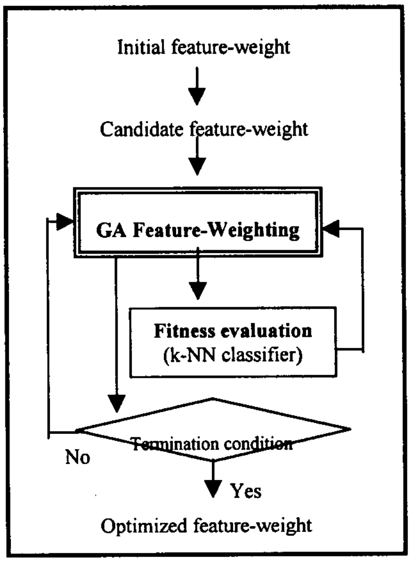
Among the four first-order models, the correlation between the logit and MDA models tended to be high (greater than 0.7). Since the predictive accuracy of logit is higher than that of MDA, the techniques of ANN, kNN and logit were used as voting modules. This voting strategy offered the advantages of simplicity and speed.

2. COLLABORATIVE LEARNING IN THE CLASSIFICATION TASK

2.1 GA as the Preprocessor

The goal of a collaborative learning approach is to combine two or more learning techniques in a synergistic fashion. As in other hybrid architectures, one of the components might involve a nested module. In particular, a genetic algorithm can be employed to optimize the weights in a lazy learning algorithm [25].

The overall strategy for the optimization is depicted in [Figure-6]. In addition, the procedure for encoding and decoding chromosomes for efficient weight discovery is outlined in [Figure-7].



[Figure-6] schematic of a simple feature-weighting strategy.

A. Representation

1. Encoding step
Gray coded representation for flexible change of decoded representation.
E. g., 011 (0.3) -> 100 (0.4) all changes are needed

B. Interpretation

1. Decoding step 1
GrayToBinary transformation.
E. g., 100 -> 111

2. Decoding step 2
BinaryToDecimal interpretation.
New_value = decimal * constant.

In this study, the range of feature-weights was [0,1.5] with 16 steps. E. g., 0.0, 0.1, 0.2,..,1.5

[Figure-7] Efficient feature-weight representation and interpretation in the form of a chromosome.

Motivation for a new performance metric. At first glance, an appropriate metric for tasks in forecasting or classification lies in the hit rate: the proportion of correct predictions. The hit rate has, in fact, been employed as the standard measure of performance in the literature. Unfortunately, the hit rate by itself suffers from the following limitations:

1. *False validation of a local extremum as the global optimum.* Even when a procedures lies far from the global optimum, the metric can yield a high hit rate even 100% for the particular test set at hand. Such misleading results are especially likely to occur for small test sets. An example of false validation is depicted in [Figure-8].

2. *Promotion of stagnation.* As indicated in the previous section, the course of optimization can lead to congestion or stagnation. In such situations only a mutation is likely to deliver the system into virgin territory. However, in a software simulation - as in nature - most mutations are likely to be detrimental to the performance of an organism. This is true even if the mutation eventually leads to other adaptations which yield better performance in subsequent generations. Consequently, the single metric of hit rate will tend to block mutations and thereby reinforce stagnation in the evolutionary process.

3. *A standard trade-off in predictive tasks lies in accuracy versus cost.* The cost takes the form of processing time as well as data collection requirements [6, 45, 51]. To minimize both varieties of cost, it is imperative to weight features or attributes in an efficacious way [38].

The preceding discussion highlights the importance of designing a performance metric which promote high accuracy as well as low cost.

Specification of metrics. As a preliminary task toward developing an appropriate fitness function, it is necessary to define an effective clustering procedure for associating similar cases. To this end, we identified a metric of distance or skew which would yield high differentiation among the clusters: in other words, large gaps between clusters but small gaps among the cases within a single cluster.

More specifically, the inter-target distance (*TD*) is defined as the distance between two target or output vectors. In a similar way, the inter-source distance (*SD*) is defined as the distance between two source or input vectors. The skew (*SK*) is defined in terms of the aggregate difference in distance between *TD* and *SD* for the entire training set (one training example is compared against all the others).

In the case of a large (or small) value of *TD*, a large (or small) *SD* could yield a small value for *SK*. A dataset with a low value of *SK* indicates good classification performance. The motivation behind this policy is the assumption that "Distinct phenomena have dissimilar causes while similar phenomena have similar causes". One beneficial consequence of this policy lies in feature reduction, which lowers the cost of data preprocessing.

The metrics of *SD* and *TD* between the i^th^ and j^th^ cases are defined as:

$$SD_{i,j} = \sqrt{\sum_{k=1}^{n} W_k (S_{i,k} - S_{j,k})^2} \qquad TD_{i,j} = \sqrt{\sum_{k=1}^{m} (T_{i,k} - T_{j,k})^2}$$

Further, the SK metric is defined as

$$SK = \sum_{i=1}^{p} \sqrt{\sum_{j=1}^{p} (TD_{i,j} - SD_{i,j})^2}$$

for a continuous-valued target, or

$$SK = \sum_{i=1}^{p}\sum_{j=1}^{p} sSD_{i,j} - \sum_{i=1}^{p}\sum_{j=1}^{p} dSD_{i,j}$$

for a discrete-valued target. Here the following notation is used:

n: number of input features
W_k: weight of k^{th} feature
$S_{i,k}$: value of k^{th} input feature in the i^{th} case
$S_{j,k}$: value of k^{th} input feature in the j^{th} case
m: number of output features
$T_{i,k}$: value of k^{th} output feature in the i^{th} case
$T_{j,k}$: value of k^{th} output feature in the j^{th} case
p: total number of training cases
$sSD_{i,j}$: the value of $SD_{i,j}$ when $T_{i,k}$ is the same as $T_{j,k}$ for all k
$dSD_{i,j}$: the value of $SD_{i,j}$ when $T_{i,k}$ differs from $T_{j,k}$ for all k

Moreover, we adopted the concept of majority and minority developed by Punch (Punch, et al. 1993).

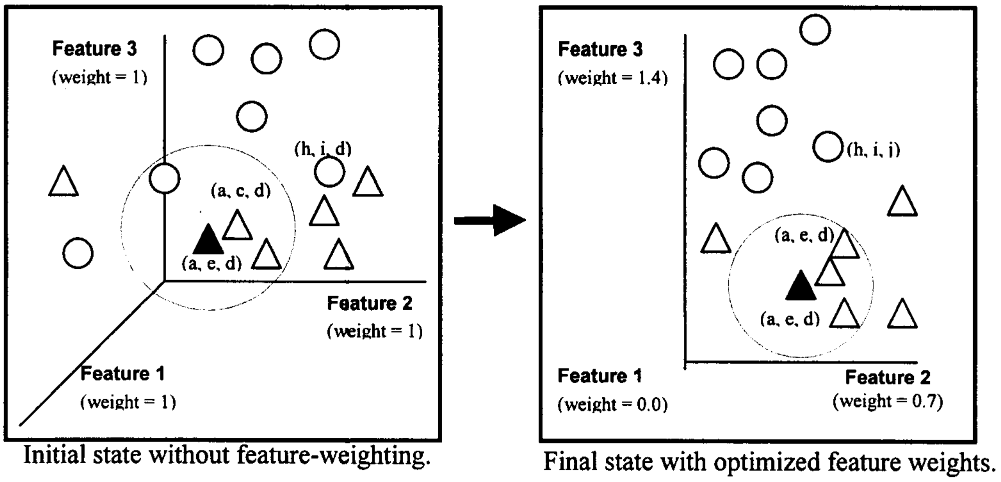
Finally, the fitness function may be defined in the following way:

$$\text{Fitness} = \alpha SK + \lambda \text{HP} + \eta (\text{cMa} / (\text{cMi} + 1))$$

where the following notation is used:

HP: hit percent = (correctClassifiedCases / totalTrainingCases) * 100
cMa: the cardinality of majority cases in k neighbors
cMi: the cardinality of minority classes in k neighbors
α: tunable parameter ranging from -0.1 to -0.001
λ: tunable parameter ranging from 1.0 to 5.0
η: tunable parameter ranging from 0.1 to 1.0

A comprehensive overview of the fitness function is depicted in [Figure-8].



[Figure-8] Example of an adaptive transformation of *feature weights for kNN* through the reduction of a dimension and re-scaling of axes. Here the number of neighbors is k=3. From the initial state of 3 features, the weight of Feature 1 has been changed to 0.0 (feature-selection effect) and the others to greater or less than 1.0. The value of *cMa* changed from 2 to 3 and *cMi* from 1 to 0. Moreover, *SK* was decreased sufficiently to yield source-level homogeneity within the same class (logical input vector (a, c, d) has been changed to (a, e, d) and (h, i, d) to (h, i, j) in a different class). In both situations, it is possible to obtain a hit rate of 1.0; however, the situation on the right is closer to the optimal solution.

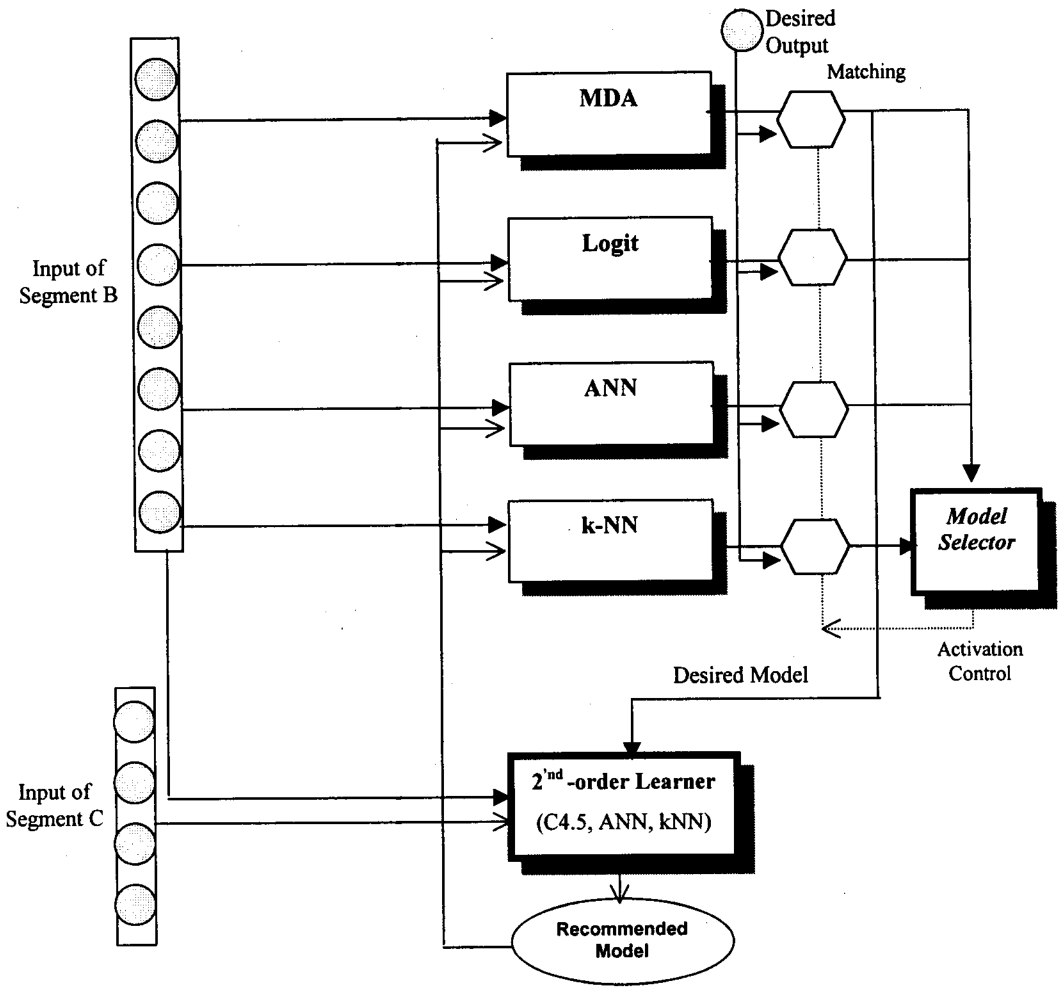
▲: Input pattern
△: Stored pattern with desired class
○: Stored pattern triggering misclassification

2.2 Postprocessing approaches

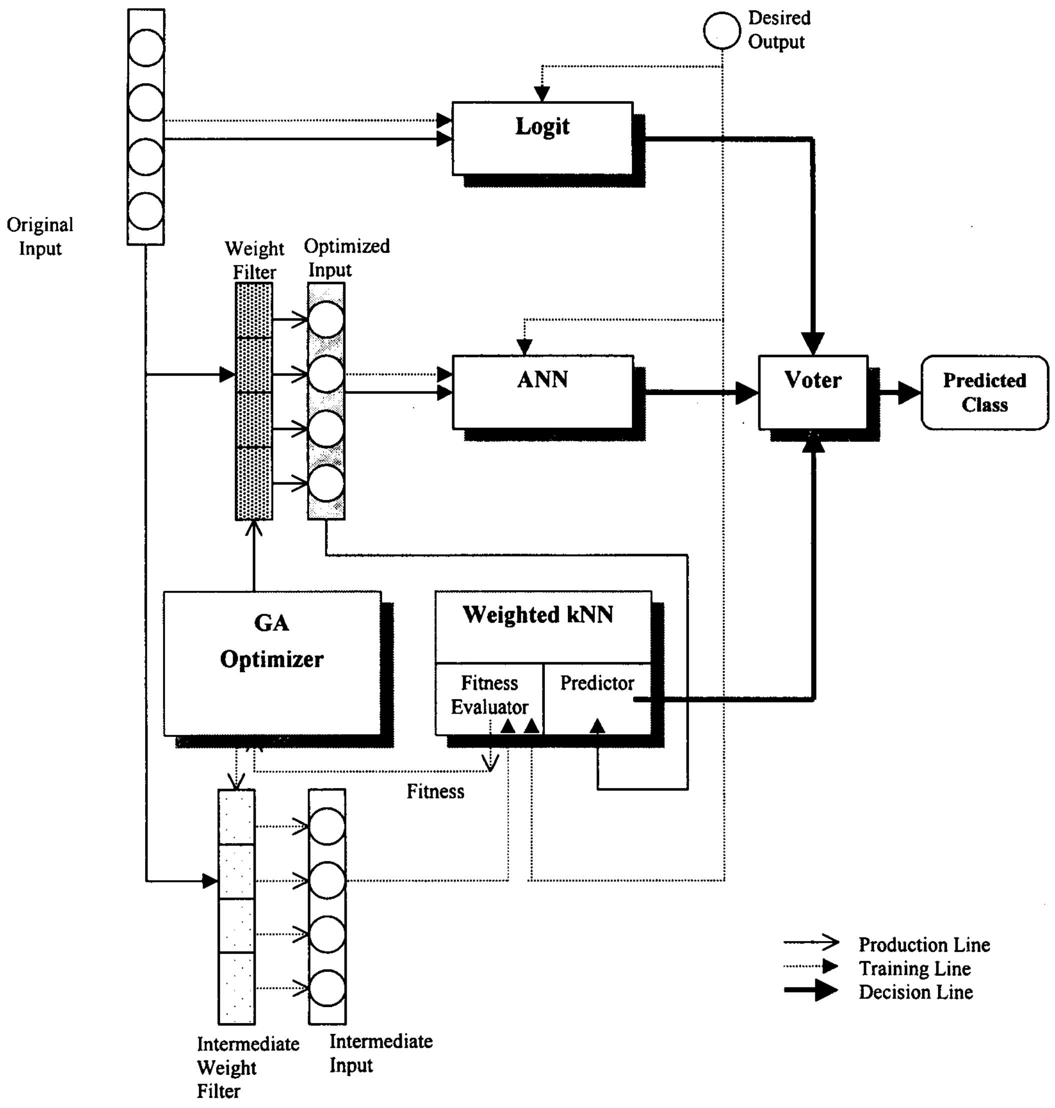
According to the terminology depicted in [Figure-2], the second-order learner and the voter act as hybrid postprocessing models. The detailed scheme for second-order learning is presented in [Figure-9]. The *model selector* identifies two appropriate models based on the correlation matrix for the four models. The desired model is chosen on a case-by-case basis for the second-order learner using Segment B of the dataset.

In contrast to the second-order learner, the voter is a simple postprocessing model which bypasses the expense of a protracted training phase. The *voter* utilizes the output of the ANN, Logit, and kNN modules to determine the appropriate output according to a simple majority rule.

The basic configuration for the voting scheme is depicted in [Figure-10]. In this architecture, a GA can serves as a preprocessor to optimize the feature weights for kNN.



[Figure-9] A second-order architecture involving second-order learning.



[Figure-10] Collaborative architecture involving an optimizing prerocessor and a voting postprocessor.

3. EXPERIMENTAL STUDY

3.1 Data sets

For the experiments, we employed three data sets: two drawn from a popular repository of data for machine learning, and one from a Korean firm. The three data sets are described in [Table-1].

[Table-1] Description of data sets used in the experiments.

Data set	Size			# of Features	# of Classes	Description
	Seg-ment A	Seg-ment B	Seg-ment C			
Aus-credit	270	210	210	14	2	Australian credit card data for classifying fraudulent transactions (confidential)
Ger-credit	400	300	300	24	2	German data for approving credit card renewals (public).
Kor-credit	400	300	300	10	2	Korean credit card data for classifying fraudulent transactions (confidential).

UCI Machine Learning Repository. The first two datasets were selected from the UCI machine learning repository. This resource contains datasets which serve as standard yardsticks for evaluating algorithms developed by the machine learning community. The applications we selected involved the realm of finance, especially typical binary classification problem- financial fraud. The information was compiled in concert with a multinational project sponsored by the ESPRIT programme [9]. The applications involved credit card approval based on customer demographics and usage histories.

3.2 Preprocessing, Implementation, and Experiments

To ensure a measure of consistency in the weights for employing LLAs, each feature of each dataset was normalized into the unit interval [0, 1]. On the other hand, the raw data were first standardized (into Z scores) for the ANN. Moreover, missing data values were filled in robust fashion: to circumvent the biased impact of outliers, the median value of each feature was used to replace missing values.

The genetic algorithm used in this study was tailored after the Goldberg model [13]. The fitness function employed relative weighting. In particular, a weighted kNN classifier [35] was used to measure fitness in the training sample space. More specifically, the weight vectors (individuals in a population) produced by the GA were fed into the kNN module in calculating the weighted distance. The ANN model was the standard backpropagation algorithm [41], but with an enhancement called Quickprop to accelerate the learning process [10].

All code except for the statistical models was written in C++ on a Sun UltraSparc Unix machine. The parameter settings for the GA were selected largely as reported in the literature [2, 3, 13, 38, 51]. More specifically, the parameter settings for each dataset were as follows.

- Population size: 50~70
- Number of generations: 50~200
- Probability of crossover: 0.5~0.7
- Probability of mutation: 0.001~0.01
- Probability of selection by ranking: 0.6
- Probability of selection by roulette: 0.4
- Crossover point: single point crossover

For the efficient and flexible representation of information, each chromosome was a gray coded string for which the Hamming distance for each pair of adjacent genes was equal to one. We adopted the single-point crossover strategy while taking account of the interference from the two-point crossover with gray code [26].

The number of nearest neighbors (k) used in kNN was selected through a five-fold cross-validation procedure. For this procedure, the value of k was varied from 1 to the square root of the total number of training patterns [26].

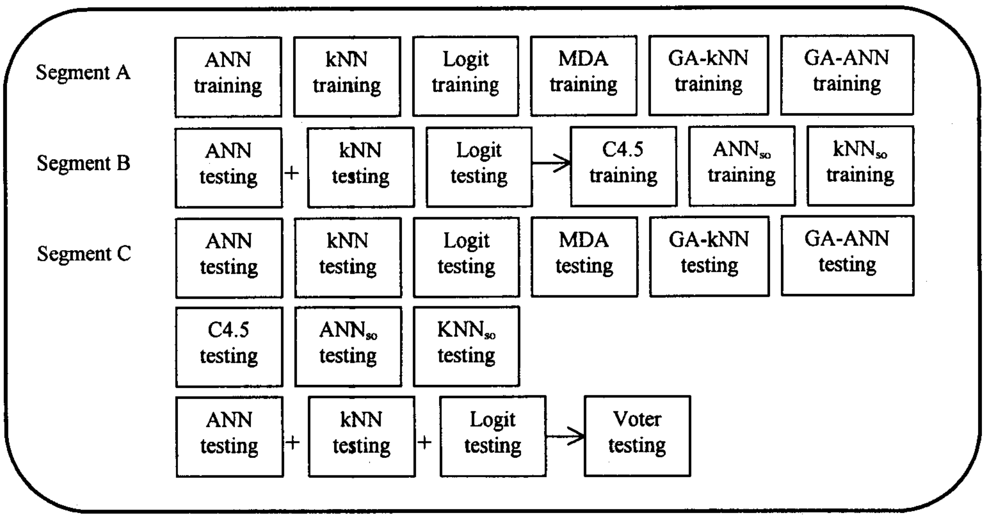
As indicated earlier, the second-order model lay in the form of the C4.5 procedure. A sample output during a user interaction with the program is displayed in [Figure-11]. For instance, the first couple of lines in the decision tree is equivalent to the following rule: *If ((x3 <= 0.205) and (x13 > 416)) then kNN is the best first-order model.* Here x3 and x13 denote particular features or attributes.

```
Simplified Decision Tree:
x3 <= 0.205 :
|   x13 > 416 : KNN (2.0/1.8)
|   x13 <= 416 :
|   |   x2 <= 19.42 : ANNKNN (3.0/1.1)
|   |   x2 > 19.42 : INVERSE (8.0/3.5)
x3 > 0.205 :
|   x8 <= 0 : ANNKNN (90.0/3.8)
|   x8 > 0 :
|   |   x9 > 0 : ANNKNN (66.0/8.4)
|   |   x9 <= 0 :
|   |   |   x7 > 7.415 : KNN (4.0/2.2)
|   |   |   x7 <= 7.415 :
|   |   |   |   x2 <= 39.58 :
|   |   |   |   |   x5 <= 5 : KNN (2.0/1.8)
|   |   |   |   |   x5 > 5 :
|   |   |   |   |   |   x13 <= 40 : INVERSE (4.0/2.2)
|   |   |   |   |   |   x13 > 40 : ANNKNN (24.0/11.2)
|   |   |   |   x2 > 39.58 :
|   |   |   |   |   x14 <= 14 : INVERSE (4.0/2.2)
|   |   |   |   |   x14 > 14 : ANN (3.0/1.1)
Tree saved
Evaluation on training data (210 items):
      Before Pruning           After Pruning
     -------------           ------------

     Size      Errors    Size      Errors    Estimate
     49      14( 6.7%)   21      24(11.4%)  (18.7%)    <<
```

```
Evaluation on test data (210 items):
       Before Pruning              After Pruning
      -------------              ------------
     Size       Errors   Size      Errors   Estimate
      49     62(29.5%)   21     57(27.1%)  (18.7%)    <<

        (a)  (b)  (c)   (d)      <-classified as
        ----------------
              2    6    3      (a): class ANN
              4   13           (b): class KNN
              3  145    8      (c): class ANNKNN
         1    2   19    4      (d): class INVERSE
```

[Figure-11] Example of training and testing the second-order model using C4.5.

During the experimental stage, five separate trials were conducted using different partitions of the datasets into training and test segments. For instance, the casebase of 690 records for the Australian application was partitioned randomly into 270 cases for the training set and the remainder for the test segment. The segmentation and use of the datasets is presented in [Figure-12].



[Figure-12] Segmentation of data sets to investigate the effectiveness of hybrid learners including GA preprocessing, second-order learning (so), and voting.

3.3 Results

This study investigated the efficacy of nine hybrid models against the four elementary models. The models are presented in [Table-2].

[Table-2] Three types of models used in this study

Elementary models	Hybrid models with four classes(FCM)	Hybrid models with two classes (TCM)	
ANN	ANN(#^)	TCM-A	GA-ANN(*)
kNN	KNN(#^)		GA-kNN(*)
Logit	C4,5(#^)		Voter(#)
MDA		TCM-B	ANN(#^)
			kNN(#^)
			C4.5(#^)

* Preprocessing model
# postprocessing model
^ second-order model

Among the nine hybrid models investigated in this study, the first three involved second-order learners with four classes (FCM). These second-order learners operated on the collective output of two elementary models A and B. Hence there were four possible combinations or classes of predictions for the second-order module: only A is correct; only B is right; both are correct; or both are wrong.

Among the remaining six models, the first three models (TCM-A) were all binary classifiers. Two of these latter models employed GA as a preprocessor, whereas the third was a simple postprocessing voter.

The expected accuracy is 25% for the FCM and 50% for the TCM models. Consequently the accuracy of a second-order learning model (FCM) cannot exceed that of the first-order models.

If there were enough conflicting samples (only one of the two first-order models is correct), it would be possible to construct a two-class second-order classifier. On the other hand, it is very difficult to distinguish conflicting patterns from noise. In this study, two-class classification for second-order learning was not feasible because of the sparsity of conflicting patterns: if the samples in this category were to be partitioned into three segments, there would not be enough cases for a meaningful analysis.

An alternative approach to second-order learning with four classes lies in a stepwise binary classification procedure such as ordinal pairwise partitioning [28].

In the first stage, a second-order model determines whether the elementary models are *able* or *unable* to correctly classify a case. The able class indicates that at least one of the two first-order models can correctly classify, while the class *unable* implies that neither can. The second step is invoked only for the *able* cases from the first stage; the second step involves the identification of the correct elementary model.

A data partition of three segments was used to validate the effectiveness of hybrid learning models, the first partition (A) was designed for the training of individual models, the second (B) for the training of collaborative learning models including FCM, TCM-A and TCM-B, and finally the third (C) for the evaluation of overall performance for each model.

The predictive accuracy of individual models on Segments B and C are summarized respectively in [Table-3] and [Table-4]. For the three application domains, the overall performance of the elementary models was an average hit rate of about 70 percent.

[Table-3] Performance of atomic models on Segment B with 2 classes (fraud, nonfraud).

Data set (# of patterns in Segment B; # of total patterns)	**Accuracy (%)**			
	ANN	**kNN (k)**	**Logit**	**DA**
Aus_credit (210; 690)	83.3	84.4 (5)	85.2	86.7
German_credit (300; 1000)	76.0	69.0 (15)	75.0	70.0
Korean_credit (300; 1000)	57.0	49.0 (7)	49.5	52.5
Average	**72.1**	**67.5**	**69.9**	**69.7**

[Table-4] Performance of atomic models on Segment C with 2 classes (fraud, nonfraud)

Data set (# of patterns in Segment C; # of total patterns)	**Accuracy (%)**			
	ANN	**kNN**	**Logit**	**DA**
Aus_credit (210; 690)	84.3	83.8	87.6	86.7
German_credit (300; 1000)	77.0	72.0	76.0	70.0
Korean_credit (300; 1000)	55.0	51.3	50.0	53.0
Average	**72.1**	**69.0**	**71.2**	**69.9**

From this experiment it was difficult to determine whether the overall predictive accuracy of the hybrid learning models is superior to that of the individual models. The performance of all four-class second-order models (FCM) were inferior to that of the solitary models. However, the two-class collaborative learning models (TCM-A) outperformed the individuals.

In [Table-5], the performance of the FCM models are shown in white cells and the TCM-A models in shade. According to the last row, all the FCM architectures yielded average hit rates of under 70%. The FCM models were all bested by each TCM-A model.

Statistical tests for the differential performance among various pairs of models is presented in [Table-6]. According to the table, the performance of the FCM models (C4.5so, kNNso and ANNso) tended to be inferior to the individuals at a statistically significant level. However, the performance of the TCM-A models (VOTER, GA-kNN and GA-ANN) was statistically superior on most of the data sets.

[Table-5] Performance of the FCM models against TCM-A models on Segment C. The results of the FCM models are in white cells; TCM-A models are presented in shade. The subscript "so" denotes a second-order learner.

Data Set	Accuracy(%)					
	Preprocessing model		Postprocessing Model			
	GA-kNN	GA-ANN	C4.5 so	kNN so	ANN so	VOTER
Aus_credit	88.1	86.2	72.9	81.4	78.1	86.2
German_credit	78.0	76.3	51.7	65.3	58.3	78.7
Korean_credit	62.0	60.3	44.0	49.0	48.3	58.0
Average	**76.0**	**74.3**	**56.2**	**65.2**	**61.6**	**74.3**

From the experiments, one interesting result emerged. The performance of the simplest collaborative model, VOTER (with low computational cost), was reasonably good compared to the expensive models employing GA preprocessors. The best performance was produced by the two-class GA-preprocessed models. In consideration of the efficiency and simplicity of the model, the two-class postprocessing VOTER offered much merit.

[Table 6] Pairwise tests for proportions to compare atomic and hybrid models. The symbol *A* indicates that the difference was significant at $p < 0.05$ in the Australian dataset; *G* for the German; and *K* for the Korean.

	GA-kNN	GA-ANN	$C4.5_{so}$	kNN_{so}	ANN_{so}	VOTER
ANN	*A K*	*A K*	*A G K*	*A G K*	*A G K*	*A G K*
kNN	*A G K*	*A G K*	*A G K*	*A G K*	*A G K*	*A G K*
Logit	*G K*	*K*	*A G K*	*A G K*	*A G*	*G K*
DA	*G K*	*G K*	*A G K*	*A G K*	*A G K*	*G K*

Since the TCM-B models involved a two-step process of identification, their overall accuracy can be obtained by multiplying the accuracy at each step. Thus the TCM-B would be inferior to the individual models unless the performance at each step is relatively high.

[Table-7] presents the performance from both the first and second steps. Although the overall accuracy at the first step was slightly superior to that of the individuals, that in the second step was almost the same (in this case, the prior expectation is that the TCM-B is effective only when the performance level of second step is near 100%). Hence, a multiplication of results from the two steps yields an overall performance level of about 50%. Consequently, the two-step procedure appears to have no practical value.

[Table-7] Performance of second-order models (TCM-B) on Segment C with two classes and two steps. In the first step, the two classes were: at least one out of two is able (can predict), vs. both are unable. At the second step: one model is able; the other model is able; or both are.

Data set (# of *able*; # of *unable*)	Accuracy (%)								
	C4.5$_{so}$			kNN$_{so}$			ANN$_{so}$		
	1st	2nd	Total	1st	2nd	Total	1st	2nd	Total
Aus_credit (184; 26)	85.7	84.8	72.7	87.6	85.9	75.2	85.5	82.1	70.2
German_credit (244; 56)	78.3	70.5	55.2	78.5	74.2	58.2	75.2	71.2	53.5
Korean_credit (180; 120)	57.8	55.0	31.8	54.4	53.8	29.3	56.1	53.9	30.2
Average	73.9	70.1	**51.8**	73.5	71.3	**52.4**	72.3	69.1	**49.9**

4. CONCLUSIONS

This paper has presented a comparative study of knowledge-based methods for classification. The application involved the binary classification of financial fraud patterns among credit card vendors.

Binary classification through an atomic model is a simpler task than distinguishing among four classes using a second-order model. For this reason, the predictive accuracy of second-order models for classifying four classes was inferior to all other atomic (first-order) models. This study highlighted the fact that a hybrid architecture for metalevel learning should carefully consider the nature of the training set such as the number of cases, the nature of exemplars, the type of features, the linear separability of clusters, and so on.

In contrast to the second-order models, collaborative models yielded superior results. These latter models involved a GA preprocessor, or a voting scheme as a postprocessor.

A limitation of this study lay in the lack of samples which induced conflicting results. However, a data warehouse environment with extensive databases could provide second-order models with numerous examples of conflicting cases involving negative correlation.

In summary, the results indicate that the complexity inherent in developing a learning system for practical applications can be addressed by the judicious use of a spectrum of methodologies from data mining. In particular, an integrated method involving preprocessing and voting can significantly outperform more elementary approaches. The computational experiments indicate that learning techniques can be combined synergistically to yield performance beyond that of the component modules.

The hybrid approach appears to be applicable to many other domains where neural networks and other data mining tasks are currently employed: forecasting, pattern recognition, and so on. Consequently, a promising direction for the future is to apply the concepts in this paper to novel domains to determine the generality of the results.

Reference

[1] [Aha, D. W., Kibler, D. and Albert, M. K.] (1991). Instance-based learning algorithms , Machine Learning, vol. 6, pp.37-66.

[2] [Brill, F. Z., Brown, E. and Martin, W. N.] (1992). Fast genetic selection of features for neural network classifiers , IEEE Trans. Neural Networks, vol. 3, no. 2, pp.324-328.

[3] [Chambers, L.] (1995). Practical Handbook of Genetic Algorithms, CRC Press.

[4] [Cover, T. M. and Hart, P. E.] (1967). Nearest neighbor pattern classification , IEEE Trans. Information Theory, vol. 13, no. 1, pp.21-27.

[5] [Dasarathy, B. V.] (1991). Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, Los Alamitos, CA: IEEE Computer Society Press.

[6] [Dash, M. and Liu, H.] (1997). Feature selection for classification , Intelligent Data Analysis, http://www-east.elsevier.com/ida/browse/0103/ida00013/article.htm.

[7] [Egilmez, K. and S. H. Kim] (1990). "A Logical Approach to Knowledge Based Control". J. of Intelligent Manufacturing, v. 1, pp.59-76.

[8] [Egilmez, K. and S. H. Kim] (1992). "Control under Uncertainty through Zone Logic". Trans. ASME/ J. of Dynamic Systems, Measurement and Control, v. 114(3), pp.375-389.

[9] [Esprit Statlog] (1993). Comparative testing and evaluation of statistical and logical learning algorithms for large-scale applications in classification, prediction and control.

[10] [Fahlman, S. E.] (1988). Faster-learning variations on back-propagation: An empirical study . In T. J. Sejnowski G. E. Hinton and D. S. Touretzky, editors, 1988 Connectionist Models Summer School, San Mateo, CA, 1988. Morgan Kaufmann.

[11] [Fayyad, U. M., et al.] (1996). Advances in Knowledge Discovery

and Data Mining, AAAI Press / The MIT Press.

[12] [Forrest, S.] (1993). Genetic algorithms: principles of natural selection applied to computation , Science, vol. 261, pp. 872-878.

[13] [Goldberg, D. E.] (1989). Genetic Algorithms in Search, Optimization, and Machine Learning, Reading, MA: Addison Wesley.

[14] [Holland, J. H.] (1975). Adaptation in Natural and Artificial Systems, Ann Arbor: University of Michigan Press.

[15] [Hopfield, J. J.] (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities , Proc. Nat. Acad. Sciences USA, v. 79(8), pp. 2554-2558.

[16] [Jo, H., Han, I. and Lee. H.] (1997). Bankruptcy prediction using case-based reasoning, neural networks, and discriminant analysis , Expert Systems with Applications, vol. 13, pp. 97-108.

[17] [Kelly, J. D., Jr. and Davis, L.] (1991]. A hybrid genetic algorithm for classification , in Proc. Int. Joint Conf. on Artificial Intelligence, pp. 645-650.

[18] [Kim, S. H.] (1994a). Learning Systems for Process Automation through Knowledge Integration. Proc. Second World Congress on Expert Systems, Lisbon, Portugal, CD, item 63.

[19] [Kim, S. H.] (1994b). Learning and Coordination. Dordrecht, Netherlands: Kluwer.

[20] [Kim, S. H. and J. Joo] (1997). Enhanced Prediction of Network Delays through Second-Order Data Mining: Application to Video on Demand , 1997 Spring Conf. Proc. of Korean Management Science Society and the Korean Institute of Industrial Engineers, Pohang, Korea, Apr. pp. 195-198.

[21] [Kim, S. H. and D. S. Kang] (1997). Implicit versus Explicit Learning for Forecasting: Case Study in Daily Stock Index Prediction , PACES/SPICIS 97 Conf. Proc., Singapore, Feb. pp. 93-99.

[22] [Kim, S. H. and K. M. Kim] (1996) Integration Multivariate Statistics and Neural Networks for Financial Prediction: Case Study in Interest Rate Forecasting , Proc., First Asia-Pacific

Decision Sciencess Inst. Conf., Hong Kong, June pp. 995-1003.

[23] [Kim, S. H. and H. Noh] (1996). A Comparative Study of ARIMA and Neural Network Models: Case Study in Korea Corporate Bond Yields Korean Management Science and Operations Research '96 Fall Conf. [Kor. Title], Seoul, Sept, pp. 19-22.

[24] [Kim, S. H. and M. B. Novick] (1993). Using Clustering Techniques to Support Case Reasoning . International J. of Computer Applications in Technology, v. 6(2/3), pp. 57-73.

[25] [Kim, S. H., Shin, S. W.] (1998). "Optimizing the retrieval of precedents in case-based reasoning through a genetic algorithm", Korean Expert Systems Society '98 Fall Conf. [Kor. title], Seoul, Nov., pp.123-129.

[26] [Koehn, P.] (1994). Combining genetic algorithms and neural networks: the encoding problem , Master's thesis, The University of Tennessee, Knoxville.

[27] [Kolodner, J.] (1993). Case-based reasoning, Morgan Kaufmann Pulishers, Inc.

[28] [Kwon, Y. S., Han, I., and Lee, K. C.] (1997). Ordinal pairwise partitioning (OPP) approach to neural networks training in bond rating , Intelligent Systems in Accounting, Finance and Management, vol. 6, pp.23-40.

[29] [Langley, P. and Iba, W.] (1993). Average case of a nearest neighbor algorithm , in Proc. Int. Joint Conf. on Artificial Intelligence, pp.889-894.

[30] [Leake, D., Kinley, A., and Wilson, D.] (1995). Learning to improve case adaptation by introspective reasoning and CBR , in Proc. Int. Conf. on Case-Based Reasoning, Sesimbra, Portugal.

[31] [Lee, K. C., Han, I. and Kwon, Y.] (1996). Hybrid neural network models for bankruptcy predictions , Decision Support Systems, vol. 18, pp.63-72.

[32] [Lenat, D.] (1983). EURISKO: A Program that Learns New Heuristics and Design Concepts: The Nature of Heuristics, III: Program Design and Results , Artificial Intelligence, v. 21(2),

pp.61-98.

[33] [Liu, H. and Motoda, H.] (1998). Feature transformation and subset selection , IEEE Intelligent Systems, pp.26-28.

[34] [Looney, C. G.] (1997). Pattern Recognition Using Neural Networks, NY: Oxford University Press.

[35] [Mitchell, T. M.] (1997). Machine Learning, NY: McGraw-Hill.

[36] [Murphy, P.] (1995). UCI Repository for machine learning databases, Irvine, CA: Univ. of California, Dept. of Information and Computer Science.

[37] [Porter, B. W., E. R. Bareiss and R.C. Holte] (1990). Concept Learning and Heuristic Classification in Weak-Theory Domains. Artificial Intelligence, Vol. 45, No. 1-2, pp.229-63.

[38] [Punch, W. F., Goodman, E. D., et al.] (1993). Further research on feature selection and classification using genetic algorithms , in Int. Conf. on Genetic Algorithms., pp.557-564.

[39] [Quinlan, J. R.] (1986). Induction of Decision Trees. Machine Learning, v. 1(1), pp.81-106.

[40] [Quinlan, J. R.] (1993). C4.5 Programs for Machine Learning, Morgan Kaufmann Publishers.

[41] [Rumelhart, D. E., Hinton, G. E. and Williams, R. J.] (1986). Learning Internal Representations by Error Propagation , in Parallel Distributed Processing: Explorations in the Microstructures of Cognition, vol. 1, pp.318-362, Cambridge, MA: MIT Press.

[42] [Schank, R. C. and Abelson, R.] (1977). Scripts, plans, goals and understanding , Hillsdale, NJ: Lawrence Erlbaum.

[43] [Schank, R. C. and Riesbeck, C.] (1990). Inside Case-Based Reasoning, Hillsdale, NJ: Lawrence Erlbaum.

[44] [Shibazaki, H., and S. H. Kim] (1992). Learning Systems for Manufacturing Automation: Integrating Explicit and Implicit Knowledge , Robotics and Computer-Integrated Manufacturing, v. 9.

[45] [Siedlecki, W. and Sklansky, J.] (1989). A note on genetic algorithms for large-scale feature selection , Pattern Recognition

Letters 10, pp.335-347.

[46] [Standfill, C. and Waltz, D.] (1986). Toward memory-based reasoning , Communications of the ACM, pp.1213-1228.

[47] [Sycara, K. and D. Navinchandra] (1991). Index Transformation Techniques for Facilitating Creative Use of Multiple Cases. In Proc. 12th Int. Joint Conf. on Artificial Intelligence, Morgan Kaufman, Los Altos, CA, pp.347-352.

[48] [Vafaie, H. and De Jong, K.] (1998). Feature space transformation using a genetic algorithms , IEEE Intelligent Systems, pp.57-65.

[49] [Watson, I.] (1997). Applying case-based reasoning: Techniques for enterprise systems, Morgan Kaufmann Pulishers, Inc.

[50] [Wettschereck, D., Aha, D. W., and Mohri, T.] (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms , Artificial Intelligence Review, vol. 11, pp.273-314.

[51] [Yang, J. H. and Honavar, V.] (1998). Feature subset selection using a genetic algorithm , IEEE Intelligent Systems, pp.44-49.

개선된 데이터마이닝을 위한 혼합 학습구조의 제시

김형관 · 신성우

요 약

본 논문에서는 통계적 분류모형과 지능적 분류모형을 전처리 및 후처리 형태로의 결합을 통해 각 단위모형의 장점을 상호보완 함으로써 예측력을 높일 수 있는 학습구조를 제시한다. 기존 데이터에서의 최적의 학습 차원 (input feature-dimension)의 결정을 위한 전처리기의 제시와 통계적, 지능적 단위 모형의 장점만을 학습함으로써 새로운 의사결정사항에서 최적의 단위 모형 제시를 위한 이차 학습기 (second-order learner) 및 복수의 단위 모형들의 다수 합의 원칙에 따른 의사결정 모형을 후처리 결합 형태로써 제시한다.
본 연구에서는 혼합 학습구조의 성능 평가를 위해 데이터마이닝의 가장 대표적인 이진 분류의 문제에 적용하였다. 실험 결과, 전처리 구조 및 다수 합의 원칙에 따른 후처리 구조의 경우 단위 모형대비 통계적으로 유의한 우월성을 나타냈으며, 단위모형의 장점만을 학습하는 이차 학습기의 경우 기존연구에서의 시계열 연속 데이터에서와 같은 우월성을 보이지 못하였다. 연속 시계열과 같은 경우 주어진 문제에 대한 추천모형이 유일한 반면, 이진분류 문제와 같이 복수의 단위모형 추천이 가능한 상황에서는 학습이 용이하지 않기 때문이라 사료된다.
향후 이차 학습기의 성능개선을 위해서는 퍼지 확신도 개념의 도입이 필요하다 사료되며, 그밖에 Sugeno Integral, Dempter-Shaffer, Borda count등의 알고리즘을 최적모형 제시를 위한 후처리 모형으로 적용함으로써 보다 개선된 혼합 학습구조의 개발이 가능할 것으로 기대된다.