Comparison of Symbolic and Connectionist Approaches to Local Experts Integration

Pedro R. Romero and Zoran Obradović

School of Electrical Engineering and Computer Science
Washington State University, Pullman WA 99164-2752

***Abstract*— Monostrategy classification systems are very limited in the type of knowledge they can use for decision making. On the other hand, potentially better results are achievable using *multistrategy systems* that integrate two or more types of knowledge representation and/or multiple inference underlying a decision process. In this paper a decision tree and a neural network technique for competitive integration of heterogeneous local experts are proposed. The local experts are either symbolic rule-based classifiers or neural network based monostrategy learning systems. The integration is simple, as it involves no modification of existing symbolic components. The proposed competitive integration systems are tested versus a previously used cooperative neural network based approach. The experimental results on a small financial advising problem indicate significant performance improvements when using the neural network based competitive integration approach as compared to the results obtained from either the individual classifiers, a decision tree based symbolic integration or a cooperative neural network integration method. The best competitive neural network results are achieved by incorporating prior knowledge and a dynamic neural network local expert into the integrated system.**

I. Introduction

Despite their good performance on well understood domains, expert systems tend to be costly in terms of development and maintenance. In addition, they are incapable of synthesizing new knowledge and adapting to changing environments. On the other hand, neural networks technology is being increasingly used on various prediction and pattern classification problems in complex domains. While a neural network is a powerful nonlinear system able to approximate well almost any continuous function, its main weakness for learning in noisy domains is that its ability is restricted to lower-level knowledge extraction from limited sample data [1].

This paper proposes two novel approaches to the development of *multistrategy* classification systems consisting of diverse *monostrategy* classification modules. In our experiments those modules are pre-existing rule-based expert systems and backpropagation based neural networks. The proposed methods performances are compared to that of an existing integration approaches that has been reported to provide good results. The proposed and previously studied integration approaches are described in Section 2. The experimental results on a simple financial advising problem are presented in Section 3.

This work was supported in part by the National Science Foundation under grant IRI-9308523 to Z. Obradovic.

II. Multistrategy Classification Systems

The three multistrategy integration techniques compared in this study are based on the combination of existing expert systems (called *local experts*) for a given classification problem. A machine learning system is used to either select the correct output from those given by the local experts or to combine them in the most efficient way in order to produce the correct response. The three approaches, one symbolic and two neural network-based, differ both in the way they combine the monostrategy system's contribution and in the input data they use. In the rest of this paper, the symbolic learner approach is referred to as "decision tree", while the neural network-based techniques are respectively called "cooperative network" and "competitive network".

A. The Decision Tree Based Integration

This technique uses a decision tree as the integrating learning system. The objective of the decision tree is to classify each example according to which one of the expert systems, if any, should be used to predict its class. For example, given a system with two local experts, the objective of a decision tree is to distinguish among four possibilities:

Class 0	None of the local experts classifies this example correctly.
Class 1	Expert 1 alone classifies the example correctly.
Class 2	Expert 2 alone classifies the example correctly.
Class 3	Both local experts classify the example correctly.

In general, given two local experts this decision tree combiner transforms an n-class classification task into a 4-class problem. When $n < 4$, this could be interpreted as *increasing* the complexity of the problem. However, even for binary classification, the use of both local experts and a decision tree combiner might actually *simplify* the input space decision regions, as shown in Figure 1. The figure on the left

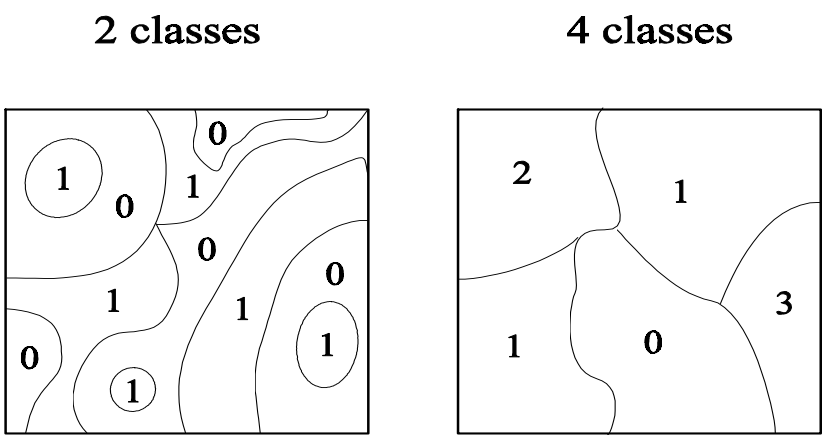

Figure 1: Decision regions for the monostrategy and multistrategy classification problems.

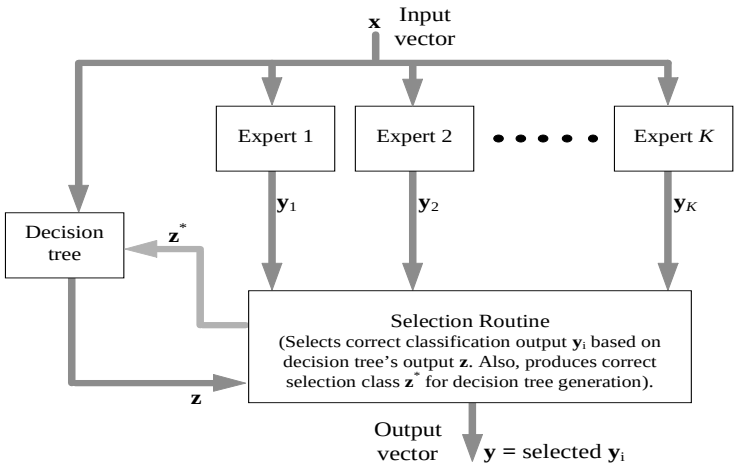

Figure 2: The Decision Tree Based Multistrategy System.

shows the decision regions for a 2-class problem. The figure on the right shows a possible class distribution resulting from using two expert systems as explained above. As it can be seen, there are now four classes, but the distribution of examples among the different classes is simpler. The assumption that this will be the case depends on the characteristics of the classification problem to be solved.

A problem with this method is that the number of classes that the decision tree combiner has to recognize grows exponentially with the number of local experts used (8 classes for three experts). This could be simplified by assigning areas where several experts are correct to just one of them, according to some criterion.

Figure 2 shows an implementation of the proposed system. Each training pattern consists of an input vector $\mathbf{x}$ and a desired response y^*, which represent the current pattern's correct class. Each local expert outputs a response y_i which is fed to a selection routine. This class information is used for the decision tree generation. The selection routine compares the local experts' outputs to the desired response y^*. Then, it assigns the input $\mathbf{x}$ to a class z^*. After the three is generated, the system works as follows: The input vector $\mathbf{x}$ is fed to the decision tree and to all local experts. The decision tree produces a response z and each expert i outputs a response y_i. All these responses are given to the selection routine, which, based on the value of z, selects one of the y_i's. The selected value is then output as the system's response.

When all experts are wrong, the system's output can be generated randomly or inferred from the data. The problem studied in this paper consists of a binary classification task with two local experts, so, if both local experts are wrong, the system only has to output the opposite class from that selected by both experts.

The construction of the decision tree can be carried out using various decision tree generation methods. The method used in this experiment is GID3* [4], which groups together irrelevant attribute values before generating the three (using ID3). The classification problem addressed in this paper

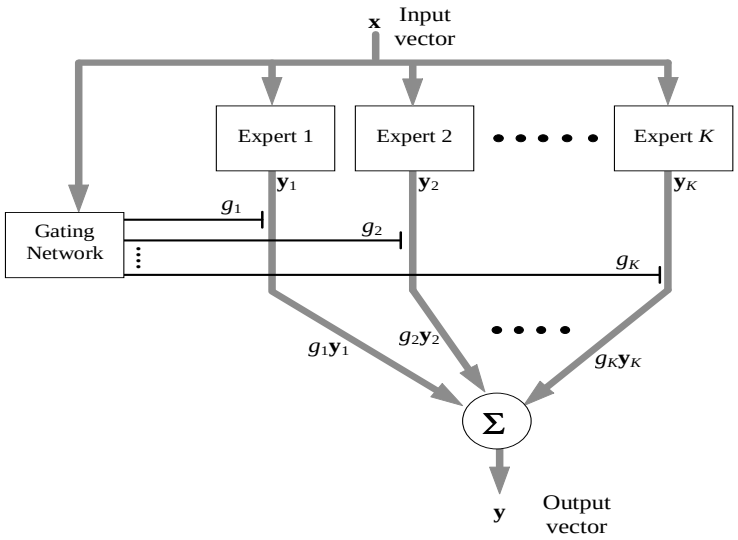

Figure 3: The Competitive Network Based Multistrategy System.

has real attributes which have to be discretized before using GID3*. This is achieved by using a multiple-interval discretization technique proposed in [3]. The common approach is to discretize the attributes at each node of the decision tree. For the problem studied here, we found that a single discretization step performed before tree generation always achieved better generalization, and so the results reported here were obtained using the latter technique.

B. The Competitive Network Based Integration

The competitive selection approach is an extension of the mixture of local experts architecture proposed in [5] which uses a *gating network* to integrate the responses of all local experts, selecting the most competent local expert for any given example (see Figure 3).

In the original architecture from [5] all K local experts are backpropagation neural networks. A supervised learning process is carried out using a set of training examples, each consisting of an input vector $\mathbf{x}= [x_1, x_2, \ldots, x_n]$ and a desired response vector $\mathbf{y}$*. As in the decision tree system, the input vector $\mathbf{x}$ is applied to both the local expert networks and the gating network.

The gating network is a one-layer neural network whose K output units use a *softmax* activation function to produce their outputs, $g_i = e^{s_i} / \sum_{j=1}^{K} e^{s_j}$, for $1 \leq i \leq K$. Here $s_i = \sum_{l=1}^{n} w_{il} x_l$ is the weighted input sum of the ith output unit, with w_{il} being the weight from the lth input to the ith output unit. The system's output $\mathbf{y} = \sum_{i=1}^{K} g_i \mathbf{y}_i$ corresponds to a weighted average of the individual expert's outputs $\mathbf{y}_1, \ldots, \mathbf{y}_K$. Notice that the softmax function is a continuous version of the winner take all selection criteria. This means that the gating network will usually select only one expert (i) by assigning a value close to 1 to its corresponding parameter (g_i) and values close to zero to the others (g_j, $j \neq i$).

The system is trained to get a maximum likelihood estimate response vector $\mathbf{y}^*$ by optimizing the error function

$$E = -\ln \sum_{i=1}^{K} g_i e^{-\frac{1}{2\sigma_i^2} \|\mathbf{y}^* - \mathbf{y}_i\|^2}$$

Here, σ_i^2 is a scaling term and E represents the log likelihood of generating a response vector $\mathbf{y}^*$. This allows a competitive learning process by training only the most competent local expert(s) on a given example, because the backpropagated error at each expert i is a function of its corresponding g_i.

In the competitive integration system used in this paper we assume that the local expert components are not only neural networks, but also classification systems of other types. We will also assume that only the gating network and the neural network components (if any) do the learning as explained above, while the other expert components are fixed and can only be used to respond to the input patterns. This approach is more demanding for the gating network, which can not rely on the fixed local experts to learn the correct classification within their assigned regions. To the best of our knowledge such models of heterogeneous mixtures of experts have not been studied before.

C. The Cooperative Network Based Integration

The cooperative integration approach is based on a jury decision making process. It has been successfully applied to several real-life classification problems including protein structure prediction [8]. This simple approach uses neural network learning to integrate different local expert classification modules by combining the outputs from K *local experts* as shown in Figure 4.

In this architecture, each example for the combiner training process is constructed from an input vector $\mathbf{x}$ and a desired response vector $\mathbf{y}$*. First, a vector $\mathbf{z} = [\mathbf{y}_1 \mathbf{y}_2 \ldots \mathbf{y}_K]$ is assembled using the classification response vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K$ obtained from K local expert classification modules when presented with the input vector $\mathbf{x}$. Second, vector $\mathbf{z}$ is fed to the combining neural network, which finally produces the output vector $\mathbf{y}$. The combining module is then trained using the desired response vectors ($\mathbf{y}$*).

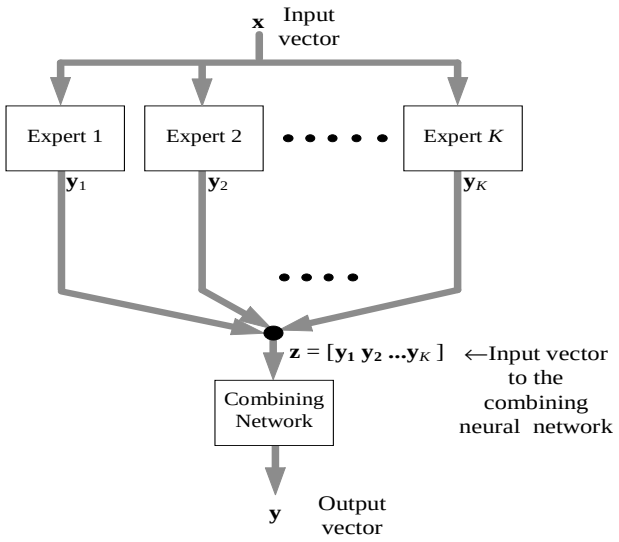

Figure 4: The Cooperative Network Based Multistrategy System.

(1)	if (savings_adequate and income_adequate)	then	**invest_stocks**
(2)	if dependent_savings_adequate	then	savings_adequate
(3)	if assets_high	then	savings_adequate
(4)	if (dependent_income_adequate and *earnings_steady*)	then	income_adequate
(5)	if debt_low	then	income_adequate
(6)	if ($savings \geq dependents \times 5000$)	then	dependent_savings_adequate
(7)	if ($income \geq 25000 + 4000 \times dependents$)	then	dependent_income_adequate
(8)	if ($assets \geq income \times 10$)	then	assets_high
(9)	if ($annual\_debt < income \times 0.30$)	then	debt_low

Table 1: Financial advisor rule base.

III. Experimental Results

A. A Financial Advising Model

The multistrategy classification systems are tested on a small financial advising problem in which the objective is to advise users whether to invest in the stock market or not, based on several financial parameters. The problem is represented by a rule base adapted from [7] and shown in Table 1. Although this model is extremely simplified, it illustrates issues involved in realistic financial advising.

The system's input consists of six real variables, shown in italics in Table 1. The system output **invest_stocks** is a binary variable, representing the recommendation on whether to invest in stocks or not.

In this work, the rule base from Table 1 was used to generate training and testing data. The idea is to produce "imperfect" expert systems for this problem and try to combine them as efficiently as possible by using the techniques explained before. In our experiments with the neural network combiners the training and testing sets were generated independently, with 1,000 examples in the training set and 10,000 in the test set. For the decision tree approach different training sets were used, ranging from 120 to 500 examples, and the testing set was the same one used for the other approaches. Pruned versions (i.e. with one or more rules missing) of the rule-base were used to create four imperfect local expert systems with diverse performances, which were used as models of real-life, rule-based financial advising systems developed using incomplete knowledge. Also, a one-layer, backpropagation neural network with four hidden units (called NN) was trained on the generated data and used as a local expert.

The expert systems shown in Table 2 can be separated into three classes: pessimistic, optimistic and mixed. A pessimistic expert's errors are always false negative predictions, that is, errors in which the output is 0 (recommending to stay out of stocks) when it should be 1 (recommending to invest). On the other hand, an optimistic expert's errors are all false positive predictions (it outputs 1 when it should say 0). A mixed expert makes both kinds of errors.

B. Results and Analysis

The multistrategy approaches presented in Section 2 were applied to the financial advising problem explained in Section 3.A. The local expert systems shown in Table 2 were used as monostrategy classification modules.

Experiments were performed by applying the three integration approaches to different combinations of expert systems and/or neural networks. For comparison purposes, an upper bound on the success rate of each combination was computed for the given test data set. This upper bound represents the maximum possible success rate achievable by

System	Success Rate	False prediction		Pruned rules	Expert type
		negative	positive		
Expert 1	65%	100%	0%	(2),(6)	pessimistic
Expert 2	69%	100%	0%	(5),(9)	pessimistic
Expert 3	82%	100%	0%	(3),(4),(7),(8)	pessimistic
Expert 4	73%	0%	100%	(7),(4)	optimistic
NN	87%	57%	43%	not applicable	mixed

Table 2: Local Expert Systems Performance.

Expert systems	Success rate on test data			
	Upper bound	Decision tree	Cooperative network	Competitive network
Experts 1 + 2	86.05%	67.75%	86.05%	85.78%
Experts 1 + 3	90.37%	68.60%	90.37%	90.02%
Experts 2 + 3	95.56%	70.00%	95.56%	94.53%
Experts 1 + 4	100.00%	68.60%	58.62%	90.60%

Table 3: Integration of two experts.

always following the correct advise, i.e., selecting a correct expert when at least one of the local experts is correct. When the bound is below 100%, it means that there are cases where all local experts are wrong, and so it is impossible to output a correct answer either by selecting one of them, or by combining their outputs. Notice that this bound is by no means tight. For example, a pair of "dumb" experts, in which one of the modules always outputs 0 and the other always outputs 1, has an upper bound of 100% (they are never both wrong), but the information they provide is nil. Thus, the cooperative combiner network can only achieve a very limited performance, while the gating network used in the competitive approach is left with the task of learning to classify the patterns by itself. Actually, the gating network has the advantage of being fed the original input in addition to the outputs of the "dumb" experts, and so it can learn to some extent how to classify the patterns. For the financial problem, a gating network combining two "dumb" experts achieved a test set success rate close to 72%.

The results of integrating several pairs of expert systems are shown in Table 3. The table presents the computed upper bound on the accuracy of each combination and the generalization (testing data) success rates obtained by implementing the decision tree, cooperative and competitive network, respectively. The results shown for the decision tree approach are averaged over twelve training sets of different sizes (120,200,300 and 500 examples). It is interesting to note here that the monostrategy decision tree approach (i.e., using a decision tree to solve the original classification problem) gave a better result (74.75%) than all the decision tree-based multistrategy systems tested here. Notice that, when combining pairs of pessimistic experts, both neural network-based methods produce excellent results. This is so because it is very easy to combine either two or more pessimistic or two or more optimistic experts by feeding their outputs to an AND gate (if the experts are optimistic) or an OR gate (if the experts are pessimistic). Obviously, both problems can be learned by a single neuron.

In contrast, the combination of pairs of experts of different types (1+4) proves to be much more difficult for the cooperative combiner approach. In this case, the competitive network achieves around 90% accuracy, and the decision tree does a better job than the cooperative combiner.

The results of combining two of the expert systems with a neural network are summarized in Table 4. The upper bound for the success rate is measured using the symbolic expert and the fixed neural network. The table shows two different implementations of the competitive network. In the fixed neural network case, the neural network shown in Table 2 was used as an expert system, i.e. it only responded to the inputs, with no further training. In the dynamic learning method, the neural network local expert was trained at the same time as the gating network.

Notice that the introduction of a "mixed" local expert (the neural network) degrades the performance of the cooperative combiner. The gating network, on the other hand does a very good job on integrating these systems, especially when the local expert neural network is allowed to learn simultaneously with the gating network. The decision tree approach managed to outperform the cooperative combiner in one of the cases, but its results continued to be very poor.

In the final experiment, each pair of experts shown in Table 3 is combined with a neural network in order to test the performance of more complex heterogeneous systems. The four systems shown in Table 5 are heterogeneous mixtures of two symbolic experts and a backpropagation neural network, using the cooperative and both competitive network approaches as in previous experiments from Table 4.

It is apparent that the competitive network performs much better than the cooperative network. We can also see that, again, allowing the expert neural network to be trained simultaneously with the gating network produces a better result compared to using a fixed neural network expert. However, some combinations (1+3+NN and 1+4+NN) are not improving as compared to the results obtained by their respective experts when they are combined alone with a neural network (see Table 4). By examining the results shown in Table 5, it seems apparent that the performance of the combined systems improves when the local experts have more similar success rates.

Expert systems	Success rate on test data				
	Upper bound	Decision tree	Cooperative network	Competitive network	
				Fixed NN	Dynamic NN
Expert 3 + NN	94.11%	74.05%	72.10%	89.00%	93.24%
Expert 4 + NN	96.62%	66.10%	78.21%	91.45%	95.33%

Table 4: Integration of one expert and a neural network.

Expert systems	Success rate on test data			
	Upper bound	Cooperative network	Competitive network	
			Fixed NN	Dynamic NN
Expert 1 + 2 + NN	99.60%	76.20%	93.79%	96.47%
Expert 1 + 3 + NN	97.71%	83.11%	88.40%	94.70%
Expert 2 + 3 + NN	98.73%	76.13%	88.51%	96.98%
Expert 1 + 4 + NN	100.00%	74.52%	91.63%	94.52%

Table 5: Integration of two experts and a neural network.

From all the results shown, it is clear that the competitive network approach is a far better tool for integrating monostrategy classification systems than both the cooperative network and the decision tree methods.

IV. Conclusions

The advantages of the multistrategy classification systems discussed in this paper are numerous:

- They integrate pre-existing prediction systems without the need for internal modification, which allows the continued use of the existing systems while improving their performance in a very inexpensive way.
- They allow the adaptive combination of older prediction systems into new, more powerful systems. The ability to explain the reasoning used in reaching a solution is maintained in the new system. When neural networks are incorporated in the multistrategy system this capability is partially lost, as there will be no explanation for the examples solved by a neural network component.
- They lower the cost of developing new classification systems by allowing the combination of specialized experts developed for smaller sub- domains. This property is especially important for large real-life problems, which are usually easier to approach locally than globally.

Several approaches to the development of multistrategy classification systems starting from existing monostrategy modules are compared on a simple financial advising problem. The proposed competitive neural network approach was found to be superior to the other integration methods and to each monostrategy classification system tested. Despite its poor performance on the studied problem, the decision tree approach could be useful on classification problems with many classes, providing local expert systems are available.

It should be noticed that the gating network used in the competitive integration approach was originally designed to combine adaptive backpropagation local experts. Thus, the gating network is a powerful, yet simple, integrator that partitions the input space among the local experts by means of simple hyperplanes. Each local expert can then learn a complex decision surface within its assigned subregion. In contrast, when combining fixed experts as in this paper, the gating network is forced to find a more complex partitioning in order to better use prior knowledge from fixed experts. This is a difficult task for the single layer gating network used here. Our current work involves examining alternate approaches that can overcome this limitation. Those include constructing more powerful gating networks and using a hierarchical competitive integration scheme as proposed in [6].

Observe that once the different local experts are efficiently integrated, the obtained system can be used as prior knowledge when developing other types of multistrategy classification systems. In particular, in our current study a multistrategy integrated system consisting of two symbolic local experts is embedded in a constructive neural network architecture described in [2] for further generalization improvement.

References

[1] Y. Abu-Mostafa, "Financial Market Applications of Learning from Hints," in *Neural Networks in the Capital Markets,* First Int. Workshop, A.N. Refenes (ed.), Willey, England, 1994.

[2] J. Fletcher and Z. Obradovic, "Combining Prior Symbolic Knowledge and Constructive Neural Networks," *Connection Science Journal,* vol. 5, Nos 3 & 4, 1993, pp. 365-375.

[3] U.M. Fayyad and K.B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning". *Proc. of IJCAI-93*, 1993, pp. 1022-1027.

[4] U.M. Fayyad, "Branching on attribute values for decision tree generation". *Proc. of the 12th National Conference on Artificial Intelligence* , 1994, pp. 601-606.

[5] R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, 1991, pp. 79-87.

[6] M. I. Jordan and R. A. Jacobs, "Hierarchical Mixtures of Experts and the EM Algorithm," *Neural Computation*, vol 6, no. 2, 1994, pp. 181-214.

[7] G.F. Luger and W.A. Stubblefield, *Artificial Intelligence and the Design of Expert Systems*, Benjamin/Cummings, 1989.

[8] X. Zhang, J. Mesirov, and D. Waltz, "Hybrid System for Protein Secondary Structure Prediction," *J. Molecular Biology,* vol 225, 1992, pp. 1049-1063.