Mathematical and Computational Applications, Vol. 14, No. 3, pp. 177-186, 2009.
© Association for Scientific Research

CSLS: CONNECTIONIST SYMBOLIC LEARNING SYSTEM

Mehmet Sabih Aksoy[1] and Hassan Mathkour[2]
[1]Department of Information Systems
[2]Department of Computer Science
King Saud University
Riyadh 11543, Saudi Arabia
email: msaksoy@ksu.edu.sa

Abstract- This paper presents CSLS, a symbiotic combination of inductive and neural learning. CSLS has two components, an induction algorithm to carry out inductive learning and a multi-layer perceptron (MLP) to implement neural learning. The paper outlines the operation of the components of CSLS and describes how the combined system is designed to utilise the individual strengths of inductive and neural learning to the best advantage. The paper gives the results of evaluating CSLS on the IRIS data and Breast-Cancer-Wisconsin-data classification problems. These clearly demonstrate the main benefit of the symbiotic combination: the combined system performs better than either of its components.

Keywords- Inductive learning; neural networks; symbiotic systems.

1. INTRODUCTION

Two important branches of machine learning are inductive learning and neural learning [1,2,3]. Both kinds of learning can be supervised or unsupervised and involve generalizing from a set of specific examples. In supervised learning, each example consists of an object, represented by its attributes or features, and the class to which the object belongs. An induction algorithm or a neural network extracts explicit or implicit rules for correctly classifying the objects in the example set. In unsupervised learning, each example only comprises object attributes; the correct object class is unknown. The induction algorithm or neural network learns to cluster the example objects into groups according to some measure of the similarity between their attributes.

A number of induction algorithms and neural network models have been proposed. Well-known induction algorithms include ID3 [4] C4.5 [5] and AQ [6, 7]. Some of the popular neural network types are the multi-layer perceptron [8], the learning vector quantisation network [9] and ART2 [10].

Existing induction algorithms and neural networks have both strengths and weaknesses. The paper explores the idea of combining an induction algorithm and a neural network to exploit the advantages of these two types of learning tools.

The paper comprises two main sections. Section 2 examines the contrasting features of inductive and neural learning and the general benefits of a combined inductive and neural learning system. Section 3 outlines the two components of the proposed combined system, describes how it operates and presents the results of evaluating it on standard classification tasks.

2. BENEFITS OF A COMBINED INDUCTIVE AND NEURAL LEARNING

The followings are the contrasting features of inductive and neural learning:

- Inductive learning does not suffer from the non-convergence problem which sometimes occurs with neural learning. An induction algorithm can always extract rules from a training set and is much faster than a neural network to learn from the same data set(s).
- Inductive learning is normally easier to implement than neural learning. Induction algorithms do not have internal parameters that require setting, unlike neural networks for which parameters to be chosen include the number of neurons, the number of neuron layers, the neighborhood size, the learning rate, and the vigilance threshold. The problem is that, often, there is not a systematic procedure for selecting those parameters.
- Inductive learning basically involves learning symbolic information. On the other hand, neural learning requires numerical data.
- Inductive learning results in transparent and interpretable rules. Neural learning tends to produce a black box with implicit input-output characteristics.
- Neural learning is generally more robust than inductive learning when the training data is noisy [11]. Also, neural networks are not as sensitive to the exact order of presentation of the training data as are most inductive algorithms.
- Neural learning is often better at generalizing than inductive learning. Invariably, inputs can be found that are not recognizable or classifiable by a rule set obtained by induction even though those inputs may only be slightly different from inputs that are classifiable. However, a neural network can always produce outputs to any given inputs [12].
- From the above discussion, combining inductive learning with neural learning can result in a system having the strengths of both neural networks and induction algorithms and therefore a better performance than either of them individually.

3. CSLS: CONNECTIONIST SYMBOLIC LEARNING SYSTEM

The proposed combined system for inductive and neural learning (CSLS) comprises two parts. The first part is RULES-EXT [13]. The second part is a multi-layer perceptron, which is perhaps one of the most simple, reliable and commonly used types of neural network. For completeness and ease of reference, a summary of the main features of RULES-EXT and the multi-layer perceptron is given below, prior to the description of the combined system.

3.1. RULES-EXT

RULES-EXT [13] is a simple algorithm for extracting a set of classification rules from a collection of examples for objects belonging to one of a number of predefined classes. RULES-EXT is an improved version of RULES3 algorithm [14]. The extra features that RULES-EXT has over RULES3 are: (1) The number of required files to extract a knowledge base (a set of rules) is reduced to 2 from 3. RULES3 a file for the set of examples, a file for attributes names and a file for classes names. RULES-EXT establish the classes file from the set of examples automatically. (2) The repeated examples are eliminated, (3) The users are able to change the order of attributes (4) The system is able to fire rule(s) partially if any of the extracted rules cannot fully be satisfied by an unseen example.

RULES-EXT uses the same rule forming procedure of RULES3 algorithm. An object must be described in terms of a fixed set of attributes, each with its own range of possible values which could be nominal or numerical.

An attribute-value pair constitutes a condition in a rule. If the number of attributes is N_a , a rule may contain between one and N_a conditions. Only conjunction of conditions is allowed in a rule. The attributes must all be different if the rule comprises more than one condition.

RULES-EXT extracts rules by considering one example at a time. It forms an array consisting of all attribute-value pairs associated with the object in that example, the total number of elements in the array being equal to the number of attributes of the object. The rule forming procedure may require at most N_a iterations per example. In the first iteration, rules may be extracted with one element from the array. Each element is examined in turn to see if, for the complete set of examples, it appears only in objects belonging to a unique class. If so, a candidate rule is obtained with that element as the condition. In either case, the next element is taken and the examination repeated until all elements in the array have been considered. If no rules have been formed, the second iteration begins with two elements of the array to be examined at a time. Rules formed in the second iteration therefore have two conditions. The procedure continues until an iteration when one or more candidate rules can be extracted or the maximum number of iterations for the example is reached. In the latter case, the example itself is adopted as the rule. If more than one candidate rule is formed for an example, the rule that classifies the highest number of examples, is selected and used to classify objects in the collection of examples. Examples of which objects are classified by the selected rule are removed from the collection. The next example remaining in the collection is then taken and rule extraction is carried out for that example. This procedure continues until there are no examples left in the collection and all objects have been classified. Figure 1 summarizes the steps involved in RULES-EXT.

```
Step 1.  Eliminate (if there are any) repeated examples
Step 2.  Find out all classes contained in the set of examples
Step 3.  Define ranges for the attributes which have numerical values and assign labels to them
Step 4.  Set the minimum number of conditions ($N_{cmin}$) for each rule
Step 5.  Take an unclassified example
Step 6.  $N_c = N_{cmin} - 1$
Step 7.  If $N_c < N_a$ then $N_c = N_c + 1$
Step 8.  Take all values or labels contained in the example
Step 9.  Form objects which are combinations of $N_c$ values or labels obtained in Step 8
Step 10. If at least one of the objects belongs to a unique class then form rules with them;
         ELSE go to Step 7
Step 11. Select the rule which classifies the highest number of examples
Step 12. Remove examples classified by the selected rule
Step 13. If there are no more unclassified examples then STOP;   ELSE go to Step 5
```

Figure 1. Induction procedure in RULES-EXT
(N_c= number of conditions, N_a= number of attributes)

3.2. Multi-Layer Perceptron

The multi-layer perceptron (MLP) is based on the perceptron, the oldest type of artificial neural network. An MLP normally consists of an input layer, an output layer and one or more hidden layers of neurons. Signals propagate in one direction from the input layer through the hidden layers to the output layer. Consequently the network is known as a feedforward network.

The neurons in an MLP usually have non-linear output activation (i.e. a non-linear transfer function). This enables an MLP to perform complex mappings which could not be achieved by the original single-layer perceptron.

MLPs are generally trained to carry out a particular mapping by applying the backpropagation supervised learning algorithm. Errors, or differences between the actual output of the network and the desired output corresponding to some training input, are propagated backwards from the output layer towards the input layer to determine the necessary adjustments to the weights of the connections between neurons in the network. The adjustments are made by following the error gradient. The aim of the training is to find the set of weights yielding the smallest error.

Training is controlled by a learning rate (η) and momentum constant (α), both in the range 0 to 1. The learning rate affects the amount of weight modification in response to a training input. Large values of η cause learning instability and conversely too small a value of η slows the learning process unacceptably. In some cases it might be useful to start with a large η and then reduce it to achieve a gradual convergence to the global minimum. The momentum constant α acts to smooth the weight modifications. In general, a high value of α can speed up the training [15, 16].

3.3. The Combined System: CSLS

CSLS (Connectionist Symbolic Learning System) is an improved version of COSINE (A Combined System for Inductive and Neural Learning) [17]. It employes RULES-EXT instead of RULES3 for induction part.

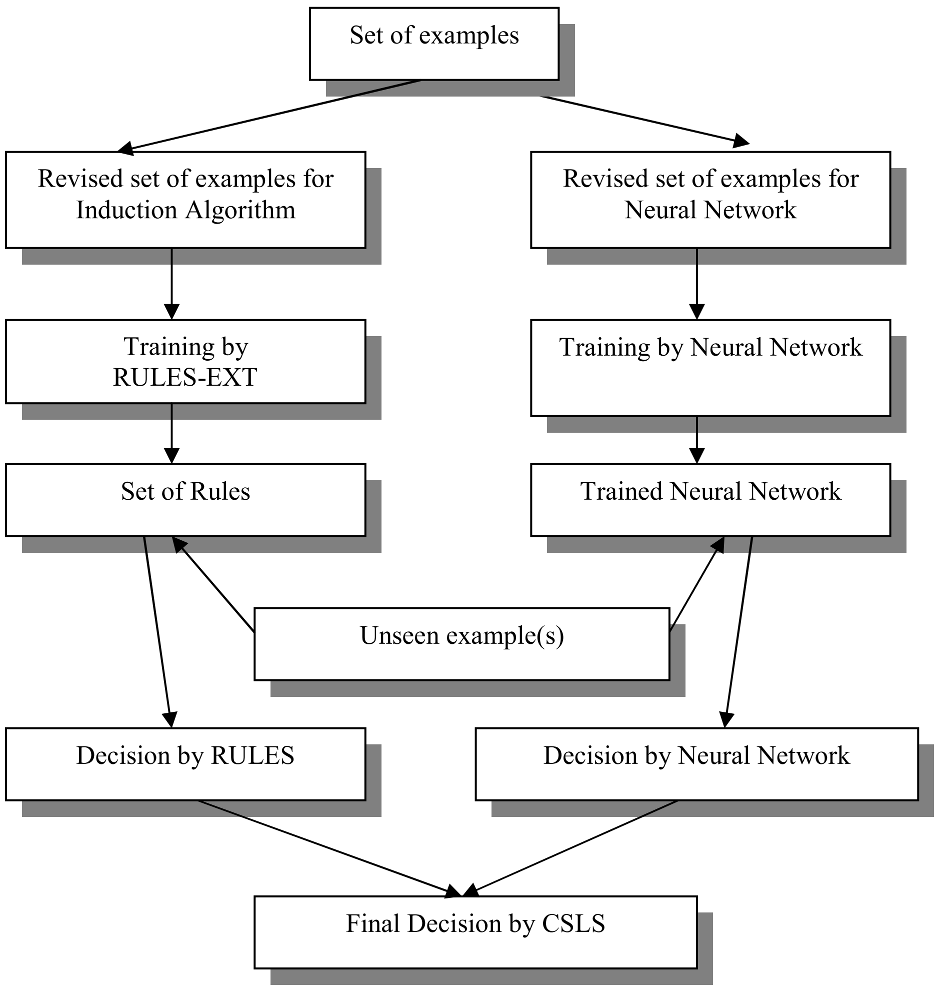
Set of examples
Revised set of examples for Induction Algorithm
Revised set of examples for Neural Network
Training by RULES-EXT
Training by Neural Network
Set of Rules
Trained Neural Network
Unseen example(s)
Decision by RULES
Decision by Neural Network
Final Decision by CSLS

Figure 2. The Operation of CSLS

Before CSLS can be employed, the multi-layer perceptron requires configuring. That is, the number of input and output neurons, the number of hidden layers, and the number of neurons in the hidden layers have to be decided. For a given problem, the number of input neurons is taken as the number of attributes and the number of output neurons, the number of classes. Each attribute is assigned to an input neuron and a class, to an output neuron. For simplicity, the number of hidden layers is fixed at 1, since it has been demonstrated that one hidden layer is sufficient for an MLP to carry out arbitrarily complex mappings [18]. The number of hidden neurons is made approximately equal to twice the number of input neurons, as it has been found empirically that this would yield a good performance. The hidden neurons have sigmoidal activation functions. The input and output neurons have linear activation functions. The training parameters (learning rate, momentum constant and maximum number of iterations) of the MLP also need setting. The training data has first to be converted into suitable formats. Numerical attributes are automatically divided into equal intervals which are given symbolic labels for RULES-EXT to handle. Nominal attributes are assigned numerical values before being input to the MLP. Classes corresponding to particular input attributes are converted into binary numbers which are set as the desired outputs of the MLP. In addition to the training data, a selection of test data is retained to evaluate the generalization ability of RULES-EXT and the MLP after they have been trained on a given problem. The generalization ability is defined as the percentage of correct classifications made over the total number of test data.

When an unseen input is presented to CSLS, five situations can occur:

I. The MLP and RULES-EXT have the same output. In this case, that output is taken as the output of CSLS.
II. The MLP has a valid output but RULES-EXT does not have a valid output. In this situation, the output of CSLS is equated to that of the MLP.
III. The MLP does not have a valid output and the input matches one of the rules induced by RULES-EXT. The output of RULES-EXT is then adopted as that of CSLS.
IV. The MLP and RULES-EXT give different outputs. The output of CSLS in this situation is that of the component with the highest generalization ability as determined using the test set.
V. The MLP does not have a valid output and the input is not classifiable by RULES-EXT. In this case, the input is added to the training set to update the weights of the MLP and to extract new rules for RULES-EXT.

CSLS has the ability to improve one of its components using the other component. For example, if RULES-EXT cannot have a valid output but the MLP can produce a valid output corresponding to the same input, then this input-output pair is employed as a new example to train RULES-EXT. On the other hand, if the MLP does not have a valid output when presented with a new input which, in contrast, RULES-EXT is able to classify, then the outcome of RULES-EXT is set aside to train the MLP. The Operation of CSLS is shown in Figure 2.

4. EVALUATION RESULTS

The system was evaluated on the IRIS data and Breast-Cancer-Wisconsin-data classification problems that are explained in the following three sections.

4.1. The IRIS data Problem

The data was employed by Fisher [19] to derive the linear discriminant function and is still the standard discriminant analysis example used for testing most current statistical pattern classification algorithms [20,21]. The data set contains 150 examples belonging to one of three classes named Iris-Setosa, Iris-Versicolor and Iris-Virginica. Each item in the data set has four attributes, Sepal-Length, Sepal-Width, Petal-Length and Petal-Width. The data contains 150 examples. 59 examples were randomly chosen for training and 91 for testing CSLS. The number of quantisation levels were set to 11 for all examples for RULES-EXT. The minimum number of conditions for each rule was set to 2 and the number of rules to be extracted for each example was set to maximum. The main features of the training case for RULES-EXT are summarized in Table 1.

Table 1. The Main Features of Training of IRIS data and Breast-Cancer-Wisconsin-data for RULES-EXT

	attributes	classes	quantization Levels	training examples	extracted rules	selected Rules
IRIS	4	3	11	59	115	20
Breast Cancer	10	2	6	200	30	30

The same data set has been used to train MLP as well. The main features of training IRIS data for MLP is given in Table 2.

Table 2. The Main Features of Training of IRIS data and Breast-Cancer-Wisconsin-data for MLP

	Number of							
	inputs	outputs	hidden layers	hidden nodes	learning Rate	momentum constant	iterations	training examples
IRIS	4	3	1	9	0.4	0.8	300000	59
Breast Cancer	10	2	1	11	0.4	0.8	100000	200

4.2. The Breast-Cancer-Wisconsin-data Problem

Samples arrive periodically as Dr. Wolberg reports his clinical cases [21]. The database therefore reflects this chronological grouping of the data. The data contains 10 attributes namely: 1. Sample code number, 2. Clump Thickness, 3. Uniformity of Cell Size, 4. Uniformity of Cell Shape, 5. Marginal Adhesion, 6. Single Epithelial Cell Size, 7. Bare Nuclei, 8. Bland Chromatin, 9. Normal Nucleoli, and 10. Mitoses. The values for all attributes except Sample code number is an integer number from 1 to 10. The class is 2 for benign and 4 for malignant.

The data contains 698 examples. 200 were randomly chosen for training and 498 for test.

The main features of the training case for RULES-EXT are summarized in Table 1, and for MLP in Table 2 respectively.

4.3. Performance of CSLS for IRIS Data and Breast-Cancer-Wisconsin Data

The CSLS was tested for the two example problems and the results are given in Table 3 and Table 4 respectively. For IRIS data problem 91 and for Breast-Cancer-Wisconsin data problem 498 randomly chosen test examples have been used to evaluate the performance of the system. For IRIS data, the MLP correctly classified 84 examples out of 91, the RULES-EXT correctly classified 80 examples out of 91. The CSLS correctly classified 86 examples out of 91. As it can be seen from Table 3, the performance of CSLS is better than the performance of its two components. The reason is that, the unclassified examples by MLP and RULES-EXT are different examples. The CSLS could classify some of these examples and that is why its performance is better.

Table 3. Performances of the MLP, RULES-EXT and CSLS for IRIS data.

	MLP	**RULES-EXT**	**CSLS**
Number of test examples	91	91	91
Number of correctly classified examples	84	80	86
Number of misclassified examples	7	11	5
Performance	92%	88%	95%

The performance of CSLS for Breast-Cancer-Wisconsin data is also much better than its components. As it can be seen in Table 4, the CSLS was unable to correctly classify 17 examples out of 498. The number of unclassified examples by MLP is 39 and by RULES-EXT is 46. The performance of the system was 97% while the performance of MLP was 92% and the performance of RULES-EXT was 91%. The reason for the performance of RULES-EXT being less than the performance of MLP is that, the data sets for both cases contain totally numerical attributes.

Table 4. Performances of the MLP, RULES-EXT and CSLS for Breast-Cancer-Wisconsin data.

	MLP	**RULES-EXT**	**CSLS**
Number of test examples	498	498	498
Number of correctly classified examples	459	452	481
Number of missclassified examples	39	46	17
Performance	92%	91%	97%

4. CONCLUSION

CSLS is a symbiotic combination of inductive and neural learning. It possesses the strengths of these individual forms of learning: a guaranteed ability to learn, an equal aptitude for handling symbols and numerical data, a high degree of robustness and a good capacity for generalization. The symbiotic nature of CSLS is evident both in its adoption of the best output of its components as the overall system output and in its use of one component to improve the other when appropriate. This has resulted in CSLS having a better performance than either of its components as demonstrated in the IRIS data and Breast-Cancer-Wisconsin data classification problems.

ACKNOWLEDGEMENT

The authors would like to thank Research Center in the College of Computer and Information Sciences in King Saud University for their support and fund provided to complete this work. We also would like to thank to UCI Machine Learning Repository for providing the data sets used to train and test the system [21].

5. REFERENCES

1. D.W. Aha, D. Kibler, and M.K. Albert, Instance-based learning algorithms, in *Machine Learning, 6*, Ed: J.R. Quinlan, Kluwer Academic Publishers, Boston, pp.37-66,1991.
2. T. Samad, Towards connectionist rule-based systems, in *Proc. IEEE Int. Joint Conf. on Neural Networks*, Vol 2, June 24-27, pp.525-532, 1988.
3. D. Fisher, K. McKusick, R. Mooney, J.W. Shavlik, and G. Towell, Processing issues in comparisons of symbolic and connectionist learning systems, in *Proc. Sixth Int. Workshop on Machine Learning*, Cornell Univ., Ithaca, New York, pp.169-173,1989.
4. J.R. Quinlan, Learning efficient classification procedures and their applications to chess end games, in *Machine Learning, An Artificial Intelligence Approach*, Eds: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, Tiago, Palo Alto, CA:, pp.463-482, 1983.
5. J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, California ISBN 1-55860-238-0, 1993.
6. R.S. Michalski, and J.B. Larson, Selection of most representative training examples and incremental generation of VL1 hypothesis: The underlying methodology and the descriptions of programs ESEL and AQ11, Report No. 867, Department of Computer Science, University of Illinois, Urbana, Illinois, 1978.
7. G. Cervone, L.A. Panait, and R.S. Michalski, The Development of the AQ20 Learning System and Initial Experiments, *Tenth International Symposium on Intelligent Information Systems*, Zakopane, Poland, June, 2001.
8. D.E. Rumelhart, and J.L. McClelland, Parallel distributed processing, Vol 1, MIT Press, Cambridge, MA, 1986.
9. T. Kohonen, G. Barna and R. Chrisley, Statistical pattern recognition with neural networks: benchmarking studies, in proc. *Int. Joint Conf. on Neural Networks, Vol1,* San Diego, California, pp.61-68, 1988.
10. G.A. Carpenter and S. Grossberg, ART-2: Self-organization of stable category recognition codes for analog input patterns, *Applied Optics*, pp.4919-4930, 1987.
11. P.K. Mogili and A.K. Sunol, Machine learning approach to design of complex distillation columns, in *Applications of artificial intelligence in engineering VIII*, Eds: G.Rzevski, J.Pastor and R.A. Adey, Vol 2, Elsevier, pp.755-770, 1993.
12. M. Caudill, Expert networks, in *Neural Network PC Tools, A Practical Guide*, Eds: R.C.Eberhart and R.W.Dobbins, Academic Press, New York, pp.189-214, 1990.
13. M.S. Aksoy and H. Mathkour, Developing a General Purpose Shell for Automatic Knowledge Acquisition, a Research Project funded by King Saud University, CCIS Research Center, KSA, 2005.
14. D.T. Pham and M.S. Aksoy, A new algorithm for inductive learning, *Journal of Systems Eng.,* No.5 pp. 115-122, UK, 1993.
15. P.D. Wasserman, Neural computing theory and practice, Van Nostrand Reinhold, New York, 1989.
16. M. Zeidenberg, Neural network model in artificial intelligence, Ellis Horwood, West Sussex, UK, 1990.

17. M.S. Aksoy, New Algorithms for Machine Learning, PhD Thesis, University of Wales, College of Cardiff, 1994.
18. K. Funahashi, On the approximate realisation of continuous mappings by neural networks, *Neural Networks*, 2, pp.183-192, 1989.
19. R.A. Fisher, The use of multiple measurements in taxonomic problems, *Ann. of Eugenics*, 7, 179-188, pp. 466-475, 1936.
20. S. M. Weiss and T. Kapouleas, An empirical comparison of pattern recognition, neural nets and machine learning classification methods, in *Readings in Machine Learning*, Eds: J.W.Shavlik and T.G.Dietterich, Morgan Kaufmann, San Mateo, CA, pp.177-183, 1990.
21. A. Asuncion and D.J. Newman, UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science, 2007.