Finding Relevant Knowledge: KBCC Applied to DNA Splice-Junction Determination

Jean-Philippe Thivierge
Department of Psychology
McGill University
Montreal, QC Canada H3A 1B1
management@neurostate.com

Thomas R. Shultz
Department of Psychology and
School of Computer Science
McGill University
Montreal, QC Canada H3A 1B1
shultz@psych.mcgill.ca

Abstract - KBCC was applied to splice-junction determination using biological rules as prior knowledge. Because not all rules were recruited, KBCC was useful in assessing relevant CC sources. Fewer recruits were required than CC. However, KBCC did not outperform CC on learning speed and accuracy, due to limited relevancy of some sources.

I. INTRODUCTION

Increasing interest is being expressed to the possibility of knowledge transfer in machine learning. Much of the research in this area is conducted with the premise that networks able to profit from existing knowledge will learn faster, with more accuracy, less computation, and a better match to human cognition, as humans usually try to use prior knowledge in new learning.

Faster training times are a definite issue of concern with neural networks. In fact, these learning schemes reportedly require 100 to 1000 times as much training time as some symbolic learning algorithms [8]-[4]-[14]. Knowledge-transferring networks are a promising solution to this issue. Their advantage over standard training methods were in part demonstrated in a study of direct transfer of knowledge via connection weights [6]. Results of this algorithm demonstrate that learning is faster in networks with pre-set weights than when weights are randomly initialized. Similar results were found in another study [12]. Training times were faster and more efficient when training patterns were divided into subsets representing different parts of an overall task. Complex problems can be learned more quickly if a network's training set is divided into a series of increasingly difficult subtasks that are learned sequentially. Also, incorporating *a priori* rules in a network reduces the need for the empirical learning system to address learning of small disjuncts [7].

Learning accuracy is another important consideration when evaluating the performance of neural networks. In this regard also, knowledge-transferring networks seem to present certain advantages. The initial parameters of a network (i.e., weight values and topology) can greatly affect how well concepts are learned [1]-[5]. If these parameters could be adjusted according to the targeted task, the network could presumably avoid local minima and attain better accuracy.

Experiments on knowledge-transfer vary according to how and where past knowledge is injected into a network. Source knowledge can help configure network topology, input unit values (through pre-processing), node activation functions, objective functions, target unit values (through post-processing), training data order, initial weights (i.e., network transfer [6]), and learning parameters such as learning rate and momentum.

The present experiment examines an extension of the cascade-correlation algorithm (CC) able to recruit past knowledge in the form of single units or full networks encoding previously learned functions [3]. This particular learning scheme, referred to as Knowledge-Based Cascade-Correlation [9], constructively recruits existing knowledge, in the form of previously learned networks, in the service of new learning.

The task selected for the current research is one of splice-junction determination. DNA (deoxyribonucleic acid) typically forms a long sequence of chained amino acids, conventionally coded by the letters A, C, T, and G. In analyzing the content of these chains, a classic problem consists in the sorting of valid sequences (exons) from superfluous amino acids that do not contribute to the process of protein creation (introns). In this regard, a splice site is the point in the sequence where there is a shift from an exon strand to an intron strand or vice-versa. To identify splice-junctions, a preliminary set of five rules exist that help retrieve splice-junctions from a sequence. These rules are presented in Table 1. The notation numbers locations starting from a potential splice site, and then cites the symbol to be found. For instance, "@-5 'G'" indicates that, 5 symbols previous to the starting point, a "G" symbol should occur. By convention, no positive signs are indicated in the notation. The rules themselves only classify about 60% of the examples correctly as splice sites. Rules 2 and 5, for example, help sort out entries that are near the end of a sequence, because a splice-site typically would not occur at this location.

0-7803-7278-6/02/$10.00 ©2002 IEEE

TABLE 1
INITIAL RULES FOR SPLICE-JUNCTION DETERMINATION

Exon-Intron site:

1) @-3 'MAGGTRAGT'

2) must not contain the following stop codons (sequences of three bases that typically signal the end of an encoding region):

@-3 'TAA'	@-4 'TAA'
@-5 'TAA'	@-3 'TAG'
@-4 'TAG'	@-5 'TAG'
@-3 'TGA'	@-4 'TGA'
@-5 'TGA'	

Intron-Exon site:

3) @-3 'YAGG'

4) must be pyramidine-rich, meaning that it should have 6 'Y' between -15 and -6.

5) must not contain the following stop codons:

@1 'TAA' @2 'TAA'
@3 'TAA' @1 'TAG'
@2 'TAG' @3 'TAG'
@1 'TGA' @2 'TGA'
@3 'TGA'

The goals of this study were two-fold. The first goal was to examine the learning behaviour of KBCC on a real task. More precisely, we wanted to know if KBCC would benefit from prior knowledge, and how. A second goal was to establish a comparison between the learning performances of KBCC and those of CC on the same task. This was to give us an indication of how beneficial prior knowledge is in KBCC. Based on past results, it is hypothesized that KBCC will select the rule-based sources over control sigmoid units. In [11], KBCC preferred, in decreasing order, exact knowledge, relevant knowledge, distant and overly complex knowledge, irrelevant knowledge, and finally no knowledge. It is hypothesized that, with the help of recruited sources, KBCC will learn the task of splice-site recognition to an above-chance level.

II. DESCRPTION OF KBCC

A. General description

The KBCC algorithm is based on the cascade-correlation (CC) algorithm of Fahlman and Lebiere [3]. Research has shown that CC is capable of approximating functions and exhibiting good interpolation characteristics [13]. KBCC is essentially an extension of a CC network able to recruit sub-networks as well as single units into its architecture. It initially constructs a feed-forward network with no hidden units, and then trains by adjusting its weights according to the set of examples presented. This learning process iteratively alternates between two phases, namely input and output, with the constraint to always start and end training in output phase. A solution is reached by growing the network in depth, thus creating new fully connected hidden layers.

We now offer a basic description of the network before elaborating on each of its phases. Net input into a unit i is the weighted sum of its inputs from other units:

$$x_i = \sum_j w_{ij} a_j \qquad (1)$$

where i is the index of the receiving unit, j is the index of the sending unit, a is the activation of each sending unit, and w is the connection between units j and i. A sigmoid squashing function of range [-0.5, 0.5] was employed on the net input:

$$f(x) = \frac{1}{1+e^{-x}} - 0.5 \qquad (2)$$

The output connections, on the other hand, employed an asigmoid function in the range [0,1].

The weight update algorithm employed throughout is the Quickprop algorithm [2]. Quickprop is significantly faster than standard back-propagation because it supplements the use of slopes with second-order information on curvatures, which it estimates with the aid of slopes on the previous step. The default values for the output phase are as follows: $0.175/n$ for the learning rate (ϵ), 2.0 for the maximum growth factor (μ), and 0.0002 for the weight decay (γ). In input phase, these values are $\epsilon = 1.0/nh$, where n is the number of patterns and h is the number of units feeding the candidates, $\mu = 2.0$, and $\gamma = 0.0$.

B. Output phase

In this phase, the network adjusts only the weights connecting the input layer to the output layer.
The function to minimize is the sum-squared error (SSE) over all outputs and all training patterns:

$$F = \sum_o \sum_p \left(V_{o,p} - T_{o,p}\right)^2 \qquad (3)$$

where $V_{o,p}$ is the activation of output unit o for pattern p and $T_{o,p}$ is the target value of output o for pattern p. The output phase ends when one of these criteria is met: either (a) a pre-determined number of epochs has passed (the default is 100 epochs); (b) error reduction stagnates for a number of consecutive epochs (the default for minimum change is set to 0.01, and the default minimum for epochs

set to 8); or (c) when all output activations are within some range of their target value (the default score threshold is set to 0.4).

C. Input phase

In input phase, KBCC trains a number of candidates in parallel in order to maximize the correlation between the activation of the candidate and error at the output level. These candidates constitute the network's candidate pool, and include both single units (e.g., sigmoid units), and previously trained CC networks. The candidates from the pool receive input from all existing units except the outputs. All other weights in the network are frozen from their last epoch in output phase, and the weights entering the candidates are initialized randomly using a [0,1] weight range, with one exception. That is, the first copy of each type of candidate has its weights initialized with values of 1.0 for connections between inputs of the target network to corresponding inputs of the source network, and 0.0 elsewhere. This is to enable the network to make quick use of exact relevant knowledge, without much retraining. For candidate *c*, the correlation to maximize is given by:

$$G_c = \frac{\sum_{o_c}\sum_{o}\left|\sum_{p}\left(V_{o_c,p}-\overline{V}_{o_c}\right)\left(E_{o,p}-\overline{E}_{o}\right)\right|}{\#O_c \cdot \#O \cdot \sum_{o}\sum_{p} E_{o,p}^2} \qquad (4)$$

where $\bar{E}_o$ is the mean error at output unit *o*, and $\overline{V}_{o_c}$ is the mean activation output o_c of candidate *c*. The output error at pattern *p* is $E_{o,p} = V_{o,p} - T_{o,p}$. The input phase ends when one of the following criteria is met: either (a) after a pre-determined number of epochs or (b) when at least one correlation reaches a given value (default set at 0.2) and correlation maximization stagnates for a given number of epochs. The minimum change for this criteria is set to a proportion of 0.03, and the number of consecutive epochs is set to 8. At the end of the input phase, only the candidate with the highest correlation to the output error is conserved in the network. This new recruit is set in place by connecting its output weights to the outputs of the network with weights of small random values of range [0,1]. These connections are frozen in place in order to conserve the feature representation acquired in training.

III. METHODS

A. Datasets

The splice site recognition dataset was extracted from the UCI Machine Learning Repository. The specific data set used is the Primate splice-junction gene sequences (DNA) with associated imperfect domain theory. All examples are taken from GenBank 64.1 (ftp site: genbank.bio.net). Simulations were run using only a portion of this dataset, thus limiting the training to a total of 1000 patterns decomposed as 300 exon-intron site examples, 300 intron-exon site examples, and 400 of neither. This set deployed a 10-fold validation set used for testing [14]. In 10-fold cross validation, the set of examples is partitioned into 10 sets of equal size. Networks are trained using 9 of the sets and tested on the 10^{th}. This procedure is repeated 10 times so that each set is used as the testing set once. Unary encoding was used to pre-process the data from the repository. This encoding implemented a local representation of the input, such that 'A' becomes "0001", 'C' becomes "0010", 'T' becomes "0100", and 'G' becomes "1000". The target outputs to all networks corresponded to two values of 1 or 0. The first value was of 1 if the input contained an intron site, and the second value was 1 if the input contained an exon site. If no splice site was contained in the input, both values returned 0.

B. Source networks

Source networks added to the pool of candidates included networks of the same start topology as KBCC. These were trained to represent biological rules of splice-junction determination (see Table 1). To achieve this, the source networks were trained to distinguish between sequences that fulfilled a particular rule, and sequences that did not. Five training sets of 1000 patterns each (500 splice-junction, 500 null patterns) were defined according to the five pre-defined rules. Encoding of the input for the rules was restricted to the particular section of the sequence where the rule was to be found. For instance, if a given rule required an "ATG" sequence at position @-3, only that portion of the sequence was encoded.

C. CC and KBCC simulations

For CC simulations, the data pool for candidate units included 4 sigmoid units of output range [-0.5, 0.5]. For KBCC simulations with prior knowledge, four versions of a given CC source (single rule conditions), or four versions of each CC source (all rules condition) were added to the pool of sigmoids. For all simulations, a maximum of 20 hidden recruits was permitted. Ten networks were trained on each of the experimental conditions. Default values were used for the various parameters wherever possible.

IV. RESULTS

One-way ANOVAs were used, with an alpha accuracy of 0.01. Post-hoc comparisons were made with the LSD technique.

0-7803-7278-6/02/$10.00 ©2002 IEEE

A. CC network on the full problem

Cascade-correlation networks trained on the full splice-junction classification task found a solution after an average of 517 epochs by recruiting all possible 20 hidden units. The pattern classification error (sum-squared error; SSE) reached 0.4 on average.

B. CC networks on single rules

Between networks each trained on one of the five rules, no significant differences were found in number of training epochs or final SSE. All 20 possible hidden units were recruited in each condition. Results are summarized in Table 2.

TABLE 2
AVERAGE RESULTS OF CASCADE-CORRELATION TRAINED ON SINGLE RULES

	Recruited units	Training epochs	Sum-squared error
Rule 1	20	451	1.4
Rule 2	20	454	2.5
Rule 3	20	460	1.2
Rule 4	20	454	2
Rule 5	20	456	2

1)*Number of epochs:* Significant group differences were found when comparing the number of epochs between all experimental conditions, pooling single rules and all rules together. Post-hoc analysis revealed a consistent difference between the single rule 4 condition and all other conditions (see Figure 1).

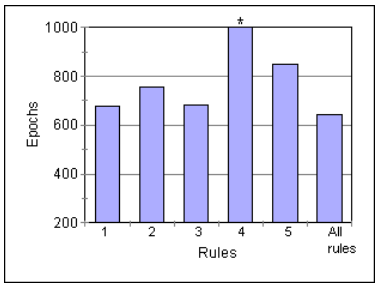

Figure 1. Training epochs for KBCC in the single-rule and all rules conditions. *p<0.01.

2)*Sum-squared error:* No significant differences were found in the SSE of the various conditions. A similar pattern of gradient descent emerged. For the all rules condition, the average SSE was of 0.45.

3)*Generalization:* No significant group differences were found in the single-rule conditions or in the all rules condition.

4)*Total number of recruits:* The total number of recruits includes both sigmoids and rule-based sources. This count was found to be higher in the single rule 4 (see Figure 2). In the all-rules condition, the average number of total recruits was of 4.8.

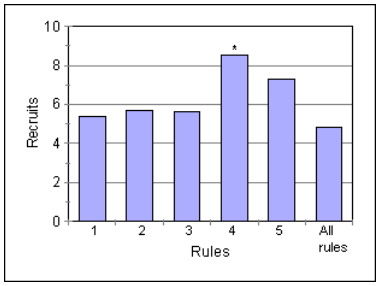

Figure 2. Average number of recruits in all conditions. *p<0.01.

5)*Number of rules recruited:* When comparing the number of rules recruited in all conditions, it was found that single rule conditions 4 and 5 required significantly less recruits, while 3 required substantially more.

6)*Preference for certain rules in the all rules condition:* In the all rules condition, certain rules seem to have been preferred over others in the recruitment process. Rule 1 (average: 1.4) and 2 (average: 1.8) seem to have been favored over the intron/exon rules, which did not get recruited at all by any of the networks.

Figure 3 compares the results of CC and KBCC on the full problem, using all rules.

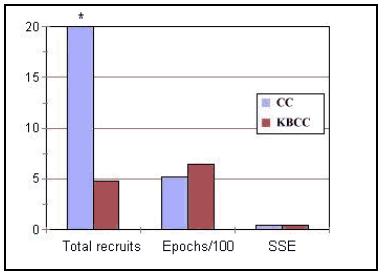

Figure 3. CC and KBCC on full problem of splice-junction determination using all rules. *p<0.01.

The only significant difference found between the two algorithms was in the number of total recruits the network required during training.

V. DISCUSSION

Throughout this study, KBCC was particularly useful in determining which rules were most relevant for solving the problem. Shavlik et al. employed their knowledge-based artificial neural network (KBANN) in solving the task of splice-junction determination using data from the same database as the current study [8]. Their network was able to find a solution by incorporating all the rules of splice-junction determination. Our results suggest that not all the rules are necessary in finding a best-solution classifier to the splice-junction determination problem. This

0-7803-7278-6/02/$10.00 ©2002 IEEE

advantage of KBCC, based on it being a constructive algorithm, might also help advance research in biology in finding rules that best capture the problem. It seems that KBCC best solves the problem by using a combination of 1 and 2 rules, and sigmoid units. For the condition including all prior knowledge, intron/exon rules were completely dismissed in solving the task. This doesn't seem to be due to a deficient learning of the source knowledge, as classification error is relatively low in the sources. Rather, it could be explained by the varying degree of relevancy of some of the sources compared to others.

Finally, the current study highlights some of the advantages and limits of KBCC as compared to its more primitive CC form. The advantage of using KBCC for solving the splice-junction determination problem does not seem to reside in training time (epochs), final classification error (sum-squared error), generalization error, or size of the final network. Rather, the important difference resides in the total number of recruits necessary (see Figure 3). In other words, KBCC achieves a solution in the same amount of time as CC and with the same precision, but requires much less incorporation of units (single sigmoids or subnetworks). One explanation for these results is that the all rules condition does not perform better than the best of the rules. An optimal solution for this problem does not necessarily require a combination of various rules, but rather at least one highly relevant one (e.g., rule 2). In conclusion, the improvements in learning time and accuracy of knowledge-based neural networks are highly dependent on the relevancy of the prior knowledge to which it has access. Poor prior knowledge may not yield significant improvements, as the network will have to construct its own correct representation.

A few paths are being explored for further research. One reason why some sources might not have been found relevant to the task is because of the input encoding, which was restricted to a delimited portion of the full sequence. We could try training the sources to full sequences, and see if the same source knowledge is preferred. It could also be possible to encode the sources as in KBANN, where the position of each rule is controlled. Another issue that needs to be addressed is the over-training of the source knowledge used in KBCC. Perhaps the source networks used for the KBCC required less than 20 units to succeed. If this is the case, then KBCC could have learned with a smaller solution.

VI. ACKNOWLEDGMENTS

This research was supported by a scholarship to J.P.T. from the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) as well as a Tomlinson scholarship, a grant to T.R.S. from FCAR, and a grant from the Natural Sciences and Engineering Research Council of Canada. This work has benefited from comments from Francois Rivest as well as one anonymous reviewer.

VII. REFERENCES

[1] S., Ahmed, "A Study of Scaling and Generalization in Neural Networks," Technical Report CCSR-88-13, University of Illinois, Center for Complex Systems Research, 1988.

[2] S.E., Fahlman, "Faster-Learning Variations on Back-Propagation: An Empirical Study," Proceedings of the 1988 Connexionist Models Summer School, Las Altos, CA, Morgan Kaufmann, 1988.

[3] S.E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture," Advances in Neural Information Processing Systems 2, Morgan Kaufmann, pp. 525-532, 1989.

[4] D.H. Fisher and K.B. McKusick, "An Empirical Comparison of ID3 and Back-Propagation," Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 788–793, Detroit, MI, 1989.

[5] J.F. Kolen and J.B. Pollack, "Back-Propagation is Sensitive to Initial Conditions," Advances in Neural Information Processing Systems, Denver, CO, Morgan Kaufmann, Vol. 3, 1990.

[6] L. Y. Pratt, J. Mostow, and A.K. Candace, "Transfer of Learned Information Among Neural Networks," Proceedings of AAAI, 1991.

[7] J.W. Shavlik, "A Framework for Combining Symbolic and Neural Learning," Machine Learning, Vol. 14, pp. 321-331, 1994.

[8] J.W. Shavlik, R.J. Mooney, and G.G. Towell, "Symbolic and Neural Network Learning Algorithms: An empirical Comparison, " Machine Learning, Vol. 6, pp. 111-143, 1991.

[9] T.R. Shultz, and F. Rivest, "Knowledge-Based Cascade-Correlation," Proceedings of the International Joint Conference on Neural Networks, Los Alamitos, CA, IEEE Computer Society Press, Vol. 5, pp. 641-646, 2000.

[11] T.R. Shultz, & F. Rivest, "Knowledge-Based Cascade-Correlation: Using Knowledge to Speed Learning," Connection Science, Vol. 13, pp. 1-30, 2001.

[12] S.J. Tetewsky, T.R. Shultz, and Y. Takane, "Training Regimens and Function Compatibility: Implications for Understanding the Effects of Knowledge on Concept Learning," Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society, Mahwah, NJ, Erlbaum, pp. 304-309, 1995.

[13] S. Waugh, and A. Adams, "Function Evaluation and the Cascade-Correlation Architecture," ICNN, Perth, pp. 942-946, 1995.

[14] S.M. Weiss, and I. Kapouleas, "An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods," In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI, pp. 688-693, 1989.