Transferring Domain Rules in a Constructive Network: Introducing RBCC

Jean-Philippe Thivierge†, Frederic Dandurand, and Thomas R. Shultz
Department of Psychology
McGill University
Montreal, QC Canada H3A 1B1
†jpthiv@ego.psych.mcgill.ca

Abstract - **A new type of neural network is introduced where symbolic rules are combined using a constructive algorithm. Initially, symbolic rules are converted into networks. Rule-based Cascade-correlation (RBCC) then grows its architecture by a competitive process where these rule-based networks strive at capturing as much of the error as possible. A pruning technique for RBCC is also introduced, and the performance of the algorithm is assessed both on a simple artificial problem and on a real-world task of DNA splice-junction determination. Results of the real-world problem demonstrate the advantages of RBCC over other related algorithms in terms of processing time and accuracy.**

I. Introduction

There is a long-standing debate in machine learning opposing symbolic rule-based systems to neural networks. Some empirical research has found a pronounced speed advantage of symbolic learning algorithms over neural networks [1][2][3]. Another advantage of symbolic systems over neural networks is that it is easier to translate an expert's knowledge about a given problem into an algorithm. This knowledge, in turn, can facilitate the acquisition of a problem to which it applies. The operation of translating domain knowledge into an artificial solver can be particularly advantageous in problems where available data is very noisy or only partial. One common problem, however, is that often the domain knowledge available is incomplete, or only of limited relevance to the problem at hand. A solution to this may be to combine the rule-based process of symbolic systems with the inductive process of neural networks. Such a hybrid system could potentially deal with incomplete rules as well as degraded input [4].

In attempting to combine a rule-based system with a neural network, a possible problem is that many possible rules may be available for the learner to choose from. The learner must select one or a subset of the best rules available, and this may result in a large number of combinations. An emergent paradox is that it can be hard to know which rules to select without a good problem representation, and hard to obtain a good problem representation without selecting an appropriate set of rules. This vicious circle can be escaped by combining both rule-based inference and inductive learning, thus making it possible to take small steps towards an adequate problem representation before a full set of rules get selected. This problem of rule selection is known to the field of learning theory, where finding a particular rule (or model) that satisfies a given logical theory is said to be NP-complete [5].

Another obstacle to rule selection is the possibility of a combinatorial explosion in the combination of the available rules. For instance, rule *A* may be crucial in solving a problem, but its selection and proper use may depend on the prior application of rule *B*. These possible complications in applying a rule to the solution of a problem make the study of rule-based learning a challenging area. In fact, few neural network algorithms are able to capture a combination of rule-based and inductive learning. One such model is KBANN [6], a method for converting a set of symbolic domain rules into a feed-forward network with the final rule conclusions at the output, and intermediate rule conclusions at the hidden unit level. This technique involves designing subnetworks representing rules. These subnetworks are combined to form the main network used to learn the task. The knowledge contained in the subnetworks does not have to be complete or perfectly correct, because the main network can adjust its connecting weights to adaptively make up for missing knowledge.

Other models address some of the issues involved in knowledge-based problem solving (e.g., KRES, [7]; discriminability-based transfer, [8]; MTL, [9]; mixture-of-experts, [10]; explanation-based neural networks, [11]; DRAMA, [12]; LISA, [13]; ATRIUM, [14]; see [15] for a partial review). Finally, a modular network has been proposed to capture a combination of rule-based and case-based performance on past tense acquisition [16]. However, none of the models provide a satisfying answer to the problems discussed above.

The current paper introduces a new algorithm for selecting and transferring symbolic domain rules injected into a neural network. We first describe this new technique along with a pruning method, and then test its performance on an artificial as well as a real-world task.

II. Description of RBCC

The model proposed in the current paper is a special case

of the Knowledge-based Cascade-correlation algorithm (KBCC)[17]. As will be argued, this new model, termed Rule-based Cascade-correlation (RBCC), has the ability to perform selection and use of rule-based prior knowledge. RBCC does so by combining rule-based inference and inductive learning.

A. Training RBCC networks

RBCC is part of a family of neural networks that includes Cascade correlation (CC) [18] and KBCC [17]. These algorithms share many common features with regards to the way they are trained and the way they are structured. These networks are initially composed of a number of input and output nodes connected together by some weights. The goal of learning is to adjust the weights in order to reduce the sum of squared errors (SSE) obtained by comparing the output of the network V to a target response T:

$$F = \sum_{o}\sum_{p}\left(V_{o,p} - T_{o,p}\right)^2 \tag{1}$$

summed over every output O and every pattern P. Only the weights that link directly to the output nodes get adjusted for this purpose. In addition to adjusting their weights, networks of the CC family can also expand their topology while learning. This is perfomed by training a number of candidate components in parallel, and eventually selecting the one with the highest covariance with the network's error, as calculated by G:

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

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 $E_{o,p} = V_{o,p} - T_{o,p}$. CC, RBCC, and KBCC differ from one another by the type of components made available to them in order to grow. CC networks are at the most basic level, and are limited to adding simple nodes, most commonly logistic units. KBCC networks, however, can also recruit fully pre-trained neural networks. Finally, in RBCC, the network is able to recruit formal rules encoded in neural networks. The newly recruited components get placed between the input and the output layers of the network. The goal of recruiting new units is to increase computational power according to the demands of a given task. Weights feeding the hidden nodes are trained to maximize the covariance between the output of the hidden nodes and the error of the network. A number of new units can be added, each in a new layer connected to all layers below it and to the output layer.

RBCC is able to perform both inductive learning and rule-based inference, respectively by lowering its error rate and by recruiting pre-trained networks encoding rules. RBCC is also able to perform knowledge selection by incorporating in its architecture those elements that best help solve the target task. In addition, RBCC has the advantage of being able to incorporate prior knowledge that only represents a partial solution to the target problem [19]. Finally, because new components are incorporated by stacking them on top of previously incorporated components, RBCC can potentially perform knowledge combination.

B. Encoding rules in RBCC networks

The domain rules used by RBCC are encoded in a similar fashion to KBANN [21]. KBANN proposes a way to generate networks that represent *if...then* rules. However, in their original paper, the authors of KBANN only propose a framework for creating disjunctive ("OR") and conjunctive ("AND") rules. In our current paper, we propose a generalization that allows for any *n-of-m* rule, where n is the number of features to select among m. This generalization has the advantage of simplifying the notation by replacing long chains of conjunctive and disjunctive rules by a simpler *n-of-m* chain. This strategy also enables us to produce a generalized rule for creating any *n-of-m* network. In a rule network, all weights are equal to W=4, except the bias:

$$x = (2N + 2M - 1)W \tag{3}$$

where x is the value of the bias weight [21]. The network itself is composed of two layers, one with m number of units for the input, and one with a single unit for the output. This way of encoding rules enables the creation of a network that fires only if at least n of the m features are present at the input. The rule networks created are presented to RBCC for selection in the same way that candidates are presented to CC and KBCC networks. Thus, the proposed model is a combination of rules generated by KBANN, and learning performed by RBCC.

A given domain rule available to RBCC can be defined according to a Horn clause. Horn clauses are a way to represent conjunctive and disjunctive rules. This representation involves a string of features each identified by a different number. If all conditions are met, the rule returns true; otherwise it returns false. A comma separates each feature in the string, and indicates a conjunction ("AND"). If a feature is necessary for the rule to return true, its number is included in the string. Conversely, if the absence of a feature is necessary for the rule to return true, its number is represented using a 'not' statement following the format "not(i)", where i represents the feature in question. For instance, if rule A requires the presence of feature i and absence of feature j to fire, this would be represented as follows:

$$A := i, not(j) \quad (4)$$

C. Pruning RBCC networks

It is possible to analyse the contribution of individual connections in RBCC networks by using a measure of saliency [22], and remove connections that have a low overall contribution. Saliencies have been used elsewhere to remove unnecessary or redundant connections from CC networks [23] (see [16], for a rule-based network that incorporates pruning). Saliencies are based on the second order partial derivatives of the objective function driving the learning process, with respect to weights. These values are contained in the Hessian matrix. To facilitate computations, only the diagonal entries of the matrix are considered, all other weights being assumed to be of negligibly small values. Saliencies are computed from these diagonal entries as:

$$s_k = h_{kk} u_k^2 / 2 \quad (5)$$

for every connection k, where h_{kk} are diagonal entries of the Hessian matrix, and u are the weight values. It is argued that saliencies provide a more reliable account of a connection's contribution to a given solution than simple weight magnitudes, because weights of small value can still have an important contribution [22]. Because networks of the CC family are trained according to two distinct objective functions, namely error reduction and network growing, two separate computations are required for the saliencies, namely one for the output-side weights (feeding the output layer), and one for the input-side weights (feeding the hidden layers). The weights feeding the output layer are trained to minimize the categorization error of the network (Eq. 1). The calculation of saliencies for these weights remains unchanged from other simulations with CC [23]. However, in RBCC, the input-side weights are trained to maximize a score of covariance G between rule output and network error (Eq. 2). The saliencies of input-side weights are determined by computing the second order partial derivatives of the covariation with respect to the weights feeding each rule. The first derivative of input-side weights w_{ij} feeding a rule f is obtained as:

$$\frac{\partial G}{\partial w_{ij}} = f' f' z x_{ij} \quad (6)$$

where z is the weight linking the output of the rule to the output of the network. The second derivative is:

$$\frac{\partial^2 G}{\partial w^2_{ij}} = z w_{ij} f'' f'' z x_{ij} \quad (7)$$
$$= z^2 w_{ij}^2 f'' f''$$

III. An artificial problem

A small hand-designed problem was generated to assess the performance of RBCC. We aimed at creating a task where some features of the input patterns were clearly more informative of category membership than others. We also aimed at generating some domain knowledge for this task that would concern only those informative features. In this way, we wanted to determine if the network could use domain knowledge efficiently, and if it could assign this knowledge to the correct features of the task by pruning out irrelevant connections.

The task consisted in determining whether each input pattern belonged to the target category or not. Patterns not belonging to the target category were exclusively composed of random features. Patterns belonging to the target category contained an equal number of random features and features termed "diagnostic". These diagnostic features conveyed critical information about category membership because they occurred with a high probability (p=0.9) within the category, and with low probability outside of it (p=0.1). A small-scale representation of the problem is shown in Figure 1. This illustration shows 3 patterns of six features each, three of those features being generated at random, and the three others being diagnostic. The diagnostic features are based on the prototype shown above the figure. In each pattern, a different feature of this prototype is flipped (highlighted in Figure 1). The actual problem used was comprised of twenty patterns belonging to the target category. These patterns were comprised of twenty features each, namely ten random and ten diagnostic. Twenty patterns not belonging to the target category were also present in the set.

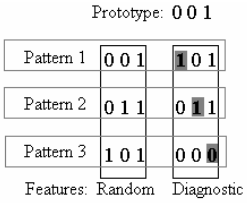

Figure 1. Design of the artificial problem.

The domain knowledge K associated with this task consisted in a rule that fired if and only if all of the diagnostic features were present. Following a Horn clause notation, and according to the prototype generated for the target category, this can be written as:

$$K := not(1), 2, 3, 4, not(5), 6, not(7), 8, not(9), not(10) \quad (8)$$

where each number represents a feature of the prototype.

Training was carried out until all patterns were classified correctly. Once training was completed, the network was tested on its ability to categorize partially occluded patterns where either the random or diagnostic features

were missing (replaced by zeros). In addition, the location of the features was varied to be either expected or unexpected. Originally, the random features occupied the left portion of the patterns, while the diagnostic features occupied the right portion. If the network is sensitive to this characteristic of the problem, categorization error should be high when it is presented with the random patterns on the right, or the diagnostic patterns on the left.

Two types of networks were used for this task, namely CC (no available rule, only simple logistic units) and RBCC (with available domain rule as well as simple logistic units). Twenty networks of each were ran.

Results were analyzed using analysis of variance (ANOVA) with a minimum level of significance of $p<0.01$. This technique has been used in a related experiment to determine if a reliable difference existed between two experimental conditions involving neural networks [19].

A. Training epochs

By means of a one-way ANOVA, we found a significant group difference in epochs between CC and RBCC ($F(3, 79) = 20.98$, $p<0.001$) conditions. Fewer training epochs were required for RBCC.

B. Testing accuracy

Testing accuracy was analyzed using a 3-way ANOVA with domain knowledge as a between subject factor, and pattern type and location as within subject factors. Significant main effects of domain knowledge ($F(1,38) = 1807.96$, $p<0.01$) were found, showing that the use of domain knowledge led to lower error rates.

The interaction of domain knowledge x pattern was also significant ($F(1,38) = 63.89$, $p<0.01$), as was the interaction of pattern x location ($F(1,38) = 49.37$, $p<0.01$). These results suggest that RBCC was sensitive to the available domain knowledge being concerned specifically with the diagnostic features, as well as mapping specifically to a certain location on the training patterns.

C. Saliencies

The saliencies of input-side weights linking to the diagnostic and random features of the target patterns are presented in Figure 2. A good strategy for RBCC is to consider more strongly the diagnostic features when establishing a correspondence between the task and the available domain knowledge. This strategy is reflected by the network assigning higher saliencies to the right of the target patterns, where the diagnostic features are found.

The saliencies linking to output-side weights were also computed (see Figure 3). The expected result is that the network should focus its attention exclusively on the diagnostic features. These expectations were confirmed; the saliencies of weights linking to the diagnostic features were larger than those linking to the random features.

Taken together, the saliencies of input and output weights suggest that the network was able to detect which features of the problem were more informative of category membership. The network was also able to adequately link the features of the rule to their corresponding features in the task to solve.

Pruning did not significantly improve categorization accuracy when compared to a baseline condition. This could be due to limited network sizes, permitting only limited pruning. In fact, the effects of pruning on increasing accuracy have been found in CC networks of large sizes [23]. Given the limited conclusions that can be derived from our small artificial problem, we now turn to a larger and more realistic task.

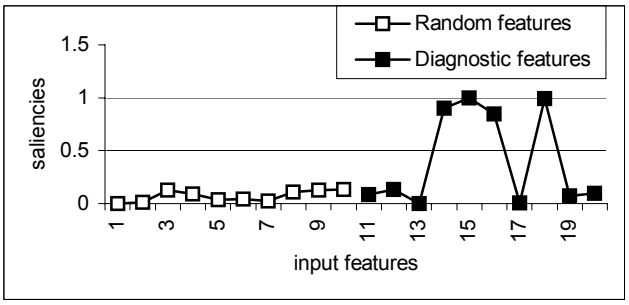

Figure 2. Saliencies of input-side weights for a typical network.

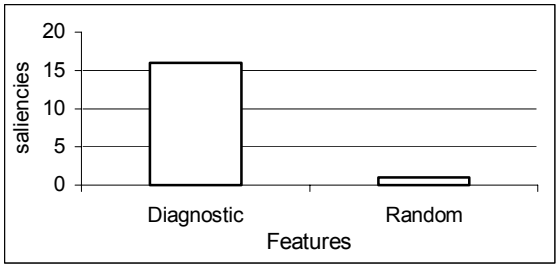

Figure 3. Saliencies of output-side weights for a typical network.

IV. DNA problem

The goal of this experiment is to compare the performance of CC, KBCC, and RBCC networks on a realistic task with a large data set. The difference between RBCC and KBCC is that, in the latter, domain knowledge has to be acquired through learning instead of being injected into the network. Some real-world problems have been successfully tackled by KBCC networks, including speech recognition [20], and DNA splice-junction determination [19]. In both cases, KBCC was able to use the available prior knowledge to improve its performance. This effect is hypothesized to be even more markedly observed with RBCC, because the available domain knowledge is more precisely represented than in standard KBCC.

The dataset for this problem was taken from the UCI Machine Learning Repository (Primate splice-junction

gene sequence (DNA) with associated imperfect domain theory). In biology, the splice-junction problem consists in finding DNA sequences that serve in the encoding of proteins (see [6] for a full description) . Five domain rules are available to help perform this classification. These rules encode regularities found in certain sequences that delimit the boundaries (exon and intron) of protein-encoding regions (see Table 1). The notation employed in Table 1 indicates the location of interest, followed by the sequence to be expected. For instance, "@-5 'G'" indicates that five symbols left of the starting point, a "G" symbol should occur in the sequence. Together, the five rules only classify about 60% of the instances correctly, so this problem is a good candidate for a network that combines both rules and inductive learning.

TABLE 1
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'

A subset of the database consisting in 1000 randomly sampled examples was used. This set was divided into a ten-fold cross validation [3], thus splitting the dataset in ten equal subsets. A network was trained on nine of these subsets and tested on the tenth. This procedure was repeated ten times with a different network so that each set is used as the test set once. Other data pre-processing considerations can be found in [19].

The data for CC and KBCC networks is taken from [19]. For the RBCC networks, a total of five rule networks were encoded, each representing a different rule. For the KBCC networks, five subnetworks were trained to acquire each of the rules (see [19] for a complete description). In all networks (CC, KBCC, and RBCC), logistic units were available for constructing the architecture. Two separate conditions were set up for the RBCC networks. In one, termed the "all rules" condition, all five rules were available for the network to select from. In the other, termed the "best rules" condition, only rules 1 and 2 were available. These rules were selected with a high frequency by KBCC networks in [19].

Table 2 compares the performance of CC, KBCC, KBANN, and RBCC networks. Significant differences were found for training ($F(3, 39) = 19.23$, $p<0.01$) as well as testing ($F(3, 39) = 103.81$, $p<0.01$) accuracy, where RBCC attained lower error than CC, KBCC, and KBANN. Significant differences were also found in the number of training epochs ($F(3, 39) = 138.51$, $p<0.01$), where RBCC using only the best rules took the shortest time, followed by CC, KBCC, RBCC using all rules, and KBANN, which took the longest. Finally, the number of installed units was different between all networks ($F(3, 39) = 2228.50$, $p<0.01$), with CC taking the most, followed by RBCC (all rules), KBANN (all rules) KBCC, KBANN (best rules), and RBCC (best rules). The KBCC, RBCC, and KBANN networks, however, generated architectures of much larger sizes than CC, because the latter only recruited single units, and not full networks.

Thus, RBCC outperformed CC, KBCC, and KBANN in terms of accuracy, but took more epochs to complete unless the number of rules to select from was restricted to a small subset representing the best rules. This suggests that the abilities of RBCC to efficiently select among prior knowledge of varying relevance is somewhat less than that of KBCC.

One advantage of RBCC over KBCC is the compactness of representation of the rules. Rules one to five took respectively 609, 441, 651, 399, and 441 connections to encode in KBCC, whereas each rule was encoded in 240 connections using RBCC. Finally, RBCC's rules are encoded more precisely, because the networks representing them are designed not to make errors.

TABLE 2
CC, KBCC, KBANN, AND RBCC NETWORKS ON THE DNA TASK

	Train SSE	Test SSE	Epochs	Recruits
CC	40.0*	79.27*	517*	20.0
KBANN all rules	839.60	91.38*	1000*	3.0*
KBANN best rules	862.51	94.61*	1000*	7.0***
KBCC	45.09*	74.17*	644.5**	4.8**
RBCC all rules	7.01*	32.66	600.25**	7.0***
RBCC best rules	6.24*	26.54	300.3	3*

* = homogeneous subsets within row ($p < 0.01$).

The improved accuracy of RBCC when compared to KBCC reinforces a documented idea according to which the advantages of using knowledge transferring networks are directly linked to the quality and relevance of the prior knowledge available [17]. The improvements in the quality of the prior knowledge available when going from KBCC to RBCC could be mainly responsible for the improved accuracy of the latter.

V. Conclusions

In the current paper, we introduced RBCC, a

constructive neural network that can incorporate domain rules into its architecture as it learns a given task. By combining RBCC with a pruning technique, and applying the algorithm to a small artificial problem, we demonstrated an ability to correctly link the features of a rule to the corresponding features of a task. In addition, the algorithm was proven efficient in solving a real-world problem of DNA analysis using available domain knowledge.

RBCC offers a strong alternative to symbolic models of analogical transfer. These models, despite providing flexible representations of structure, are often criticized for being semantically brittle [24][25]. This "brittleness" is due to the fact that if only some of the conditions of a given rule are satisfied, the rule will not be activated. This has severe consequences with regard to the abilities of rule systems to generalize to novel data. The formalism of a rule leaves it unable to reach beyond the necessary and sufficient features that define it [16]. By comparison, in connectionist networks, a phenomenon of graceful degradation occurs, whereby the probability that a rule will fire is slowly reduced as fewer and fewer of the conditions are met. Rule-based models are also neurologically unrealistic [12]. Finally, from an engineering perspective, the ability to perform rule-based transfer in neural networks opens the possibility of integrating new findings about a classification as it becomes available through scientific research, as could be the case for DNA analysis.

Future work could focus on devising a more efficient pruning technique to remove useless connections in RBCC networks. At the current state, it would be computationally demanding to prune connections from a large RBCC, as the one generated for the DNA problem. We are also interested in possible algorithms to pre-select the knowledge to be injected in RBCC, because the speed of learning of such networks can be severely compromised if they must select among a large number of rules.

VI. Acknowledgements

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), scholarships to F.D. from FCAR and Natural Sciences and Engineering Research Council of Canada (NSERC), as well a grant to T.R.S. from NSERC. J.P.T. would like to thank Vanessa Taler as well as one anonymous reviewer for comments on the manuscript.

VII. References

[1] Fisher, D.H., & McKusick, K.B. (1989). An empirical comparison of ID3 and Back-propagation. *Proceedings of the eleventh international joint conference on artificial intelligence*, 788-793.

[2] Shavlik, J.W., Mooney, R.J., & Towell, G.G. (1991). Symbolic and neural network learning algorithms: An empirical comparison. *Machine Learning, 6*, 111-143.

[3] Weiss, S.M., & Kapouleas, I. (1989). An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. *Processings of the eleventh international joint conference on artificial intelligence*, 688-693.

[4] Mitchell, T.M. (1997). *Machine Learning*. McGraw-Hill.

[5] Uribe, T., & Stickel, M.E. (1994). Ordered binary decision diagrams and the davis-putnam procedure. *Proocedings of the First International Conference on Constraints in Computational Logics*, volume 845 of Lecture Notes in Computer Science.

[6] Shavlik, J.W. (1994). A Framework for Combining Symbolic and Neural Learning, *Machine Learning, 14*, 321-331.

[7] Rehder, B., & Murphy, G.L. (2001). A knowledge-resonance (KRES) model of category learning. *Proceedings of the Twenty-third Annual Conference of the Cognitive Science Society*, 821-826.

[8] Pratt, Y.L. (1993). Discriminability-based transfer between neural networks. *Advances in Neural Information Processing Systems, 5*. 204-211.

[9] Silver, D. & Mercer, R. (1996). The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science Special Issue: Transfer in Inductive Systems*. 277-294.

[10] Jacobs, R.A., Jordan, M.I., Nowlan, S.J., & Hinton, G.E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*, 79-87.

[11] Mitchell, T.M., & Thrun, S.B. (1993). Explanation-based neural network learning for robot control. *Advances in Neural Information Processing Systems, 5*, 287-294.

[12] Eliasmith, C., & Thagard, P. (2001). Integrating structure and meaning: A distributed model of analogical mapping. *Cognitive Science, 25*, 245-286.

[13] Hummel, J.E., & Holyoak, K.J. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychology Review, 104*, 427–466.

[14] Erickson, M.A.,& Kruschke, J.K. (1998). Rules and exemplars in category learning. *Journal of Experimental Psychology: General, 127*, 107-140.

[15] Pratt, L., & Jennings, B. (1996). A survey of transfer between connectionist networks. *Connection Science, 8*, 163-184.

[16] Pulvermûller, F. (1998). On the matter of rules: Past-tense formation and its significance for cognitive neuroscience. *Network: Computational Neural Systems, 9*, R1-R52.

[17] Shultz, T. R., & Rivest, F. (2001). Knowledge-based cascade-correlation: Using knowledge to speed learning. *Connection Science, 13*, 43-72.

[18] Fahlman, S.E., Lebiere, C. (1989). The cascade-correlation learning architecture. *Advances in Neural Information Processing Systems, 2*, 525-532.

[19] Thivierge, J.P., & Shultz, T.R. (2002). Finding relevant knowledge: KBCC applied to DNA splice-junction determination. *Proceedings of the IEEE International Joint Conference on Neural Network 2002*, 1401-1405.

[20] Rivest, F., & Shultz, T.R. (2002). Application of knowledge-based Cascade-Correlation to vowel recognition. *IEEE International Joint Conference on Neural Networks 2002*. 53-58.

[21] Towell, G.G., & Shavlik, J.W. (1991). The extracton of refined rules from knowledge-based neural networks. *Machine Learning, 13*, 71-101.

[22] Thivierge, J.P., Rivest, F., & Shultz, T.R. (2003). A dual-phase technique for pruning constructive networks. *Proceedings of the IEEE International Joint Conference on Neural Networks 2003*.

[23] LeCun, Y., Denker, J., & Solla, S. (1990). Optimal Brain Damage. *Advances in Neural Information Processing Systems, 2*, 598-605.

[24] Hinton, G. E. (1986). Learning distributed representations of concepts. *Eighth Conference of the Cognitive Science Society*.

[25] Clark, A., & Toribio J. (1994). "Doing without representing?" *Synthese, 101*, 401–431.