Salzburg Interest Group on Integrated Systems

siGis-017-1999 June 11, 1999

Extraction of Compact Rule Sets from Evolutionary Designed Artificial Neural Networks

1999 IEEE Systems, Man, and Cybernetics Conference (SMC'99)
October 12–15, 1999, Tokyo, Japan

Helmut A. Mayer
helmut@cosy.sbg.ac.at

Karl Fürlinger
kfuer@cosy.sbg.ac.at

Marc Strapetz
mstrap@cosy.sbg.ac.at

Department of Computer Science
University of Salzburg

siGis
Salzburg Interest Group on Integrated Systems

Correspondence to:

Universität Salzburg
Institut für Computerwissenschaften
Jakob–Haringer–Straße 2
A–5020 Salzburg
Austria

Extraction of Compact Rule Sets from Evolutionary Designed Artificial Neural Networks

Helmut A. Mayer*, Karl Fürlinger*, Marc Strapetz*
***Department of Computer Science, University of Salzburg**
A–5020 Salzburg, Austria
{helmut}, {kfuer}, {mstrap}@cosy.sbg.ac.at

ABSTRACT

While artificial neural networks (ANNs) have proven their almost universal applicability in a broad variety of problem domains, many scientists (theorists and practitioners as well) are still worried by the opacity of neural problem solvers. As well a network may perform, it is often desirable, if not necessary, to know at least some general concepts the network bases its decisions upon. During the last years a number of ANN rule extraction (RE) algorithms have been proposed addressing the problem of ANN opacity. Most RE approaches demand specific preconditions on ANN type, structure, or training methods which often prohibit the use of these algorithms for extracting rules from ANN architectures generated by evolutionary algorithms (EAs). However, evolved ANN topologies with structures no human designer would consider might represent problem knowledge very efficiently. Therefore, we utilize two RE methods having (almost) general purpose properties, namely *MofN* and *VIA* (validity interval analysis), for the extraction of rules from evolved generalized multi–layer perceptrons being trained by error–back–propagation. In order to derive compact rule bases, ANN evolution is governed by a fitness function favoring networks of low complexity without loss of accuracy. We compare ANN classification with the performance of the extracted rule bases, and analyze network and rule structures by solving the MONK's problems which have become an RE benchmark.

1. INTRODUCTION

With the ever increasing number of ANN applications in science, business, and engineering, the need to provide an explicit declarative representation of the knowledge represented by a connectionist system has grown. Specifically, in safety–critical problem domains engineers cannot take the risk of employing a "black box" system regardless of its potential usefulness. Therefore, a variety of RE algorithms have emerged (a detailed survey is given in [1]). The extraction of rules from ANNs could also narrow the gap between connectionist and symbolic approaches, and lead to a fruitful symbiosis of both worlds (in a way this is accomplished by the human brain, as we are able to generate higher order concepts from neural substrates).

Besides the opacity of a trained ANN, there are other open research issues concerning the ANN design phase. For the construction of ANN architectures for real–world problems only "rules of thumb" exist [2], and the experience of a human designer is still the most important factor for an appropriate choice. However, due to the exponential nature of this design problem evolutionary design of ANN architectures [3, 4] has been found to be a valuable alternative and usually improves the overall performance of an ANN.

Many researchers also observed that networks of low complexity, i.e., few hidden neurons and/or sparse connectivity, improve the generalization ability of ANNs [5]. This property is of potential benefit to RE algorithms, specifically, for RE techniques analyzing the ANN structure (decompositional approach), as the extracted rule bases become smaller. The more compact set of rules then possibly comprises more general concepts with increased comprehensibility to the human observer.

Combining the above ideas we employed the netGEN system [6, 7] evolving the architecture of generalized multi–layer perceptrons by means of genetic algorithms. The evolved ANN structures are trained by standard error–back–propagation and the fitness of a network is given by a weighted sum of the performance of the trained ANN on unseen data and the complexity (here simply the number of connections) of the net. For experimental comparisons the MONK's problems [8] are utilized which enable an exact analysis of the extracted rules, as the target concepts of this artificial problem are explicitly defined.

2. RULE EXTRACTION

The knowledge represented by a trained ANN is encoded in the network's weights, activation function(s), and in its architecture. In order to decode this knowledge into symbolic rules a number of RE algorithms have been proposed [1]. Andrews et al. (1995) introduced a classification scheme of RE algorithms comprising the following categories: RE technique, rule type, rule quality, RE complexity, and portability. We briefly describe these categories and classify the RE algorithms *MofN* [9] and *VIA* [10] whose details are presented below.

The specific *RE technique* is either based on the internal structure of the net (decompositional), or the ANN is used to generate examples for a learning algorithm (pedagogical). MofN is decompositional, as it analyzes the incoming weights of each hidden or output neuron. VIA is termed pedagogical by Andrews et al. (1995), however, based on our results (Section 5) we would rather suggest it to be eclectic (a blend of decompositional and pedagogical).

The *rule type* categorizes the expressive power of extracted rules, e.g., propositional, boolean, probabilistic, or fuzzy rules. MofN generates modified boolean rules, while VIA extracts propositional rules.

The *rule quality* can be assessed by different measures depending on the desired properties of the extracted rules. Certainly, the classification accuracy of the rule set should be considered, but also the fidelity of the extracted rules, i.e., the number of (dis)agreeing decisions of ANN and rule set. Comprehensibility of the rule set, primarily based on rule set size, is another important rule quality factor. Both MofN and VIA exhibited classification accuracies close or equal to the trained ANN with high rule fidelity and could also outperform symbolic machine learning methods on certain benchmark problems [9, 8, 10].

The *RE complexity* of MofN can be estimated with the complexity of ANN training (as a retraining is necessary). VIA's complexity is dependent on its underlying linear optimization algorithm.

The *portability* of an RE algorithm is an indicator for the generality of the specific technique. This is an important category, as many RE algorithms demand specific training regimes or network structures which impedes application of these techniques to existing trained ANNs. VIA is a general purpose RE algorithm (according to Thrun (1993) even for recurrent networks), while MofN seems to be restricted to discrete problems with binary input vectors.

This section would not be complete, if we did not ask the most basic, but most important question: Why do we try to extract rules from ANNs, if we could extract rules directly from data describing the problem? The answer is based on the hypothesis that certain problems can (possibly) be solved easier (or better) in the domain of ANNs [9]. RE could then be viewed as the backtransformation (connectionist to symbolic) of a problem having been transformed from the symbolic and solved in the connectionist domain.

MofN – Modified Boolean Rules

Originally, MofN was designed for the extraction of rules from knowledge based ANNs (KBANNs), where existing problem knowledge is used for the design of the ANN architecture [9]. The main benefit of KBANNs for RE is the explicit (predefined) representation of certain concepts by hidden neurons. However, the expressive power of ANNs is at least partially credited to the distributed representation of concepts, and constraining the ANN topology space might exclude the best solutions. It was also reported that (intentionally) corrupted domain knowledge made it sometimes impossible for a KBANN structure to be trained correctly [9].

On the other side, evolution does not make any assumption on the structure (if unconstrained by the genetic encoding) and carves out very effective problem–adapted ANN architectures. In other words, we substituted the explicit injection by implicit evolution of knowledge represented by the network structure. Table 1 presents the details of the MofN algorithm.

An extracted rule might look like this: `Neuron D (fires), if` $5.5 < 3.0 \times numberTrue\{A, B, C\}$. This rule can be simplified to: `D, if 2 of` $\{A, B, C\}$. Note that MofN rules are a more compact representation of disjunctive or conjunctive normal forms.

MofN [9]
⇒ *Use existing problem knowledge to construct a knowledge based ANN (KBANN) and train the network* **Evolve ANN architecture using genetic algorithms network fitness is based on conventional training**
⇒ For each neuron cluster incoming weights
⇒ Set weights of each cluster to the cluster's average
⇒ Heuristic elimination of insignificant clusters based on *unit activation* /**network classification**
⇒ Freeze all connections' weights and retrain the altered ANN adapting only biases
⇒ For each hidden and output neuron form a single rule derived from the bias (threshold) and the remaining clusters (weighted antecedents)
⇒ If possible, simplify rules

Table 1: MofN rule extraction (*emphasized* indicates original algorithm, **bold** indicates our adaptations)

VIA – Validity Interval Analysis

The central idea of VIA is to assign real valued intervals of activation to all or a subset of neurons in a trained ANN and to analyze the consistency of these intervals with the network's structure and weights. This is specifically useful, when assigning intervals to input and output neurons. If the specific intervals are consistent with the network, they can be directly transformed to rules of the form `if` $\mathcal{I} \in \mathcal{H}$ `then class` $\mathcal{C}$, where $\mathcal{I}$ is the input vector, $\mathcal{H}$ is the hypercube built by the intervals of the input neurons, and $\mathcal{C}$ is the resulting class (indicated by the intervals at the output neurons).

The consistency check is performed by minimizing the initially assigned intervals. For this task Thrun (1993) uses linear optimization which obviously requires a linear formulation of the problem. Hence, the intervals are minimized stepwise from layer to layer in forward and backward direction as long as all intervals have converged. Due to the nonlinear nature of the complete ANN the consistency of intervals can only be proven (using linear optimization) by detecting the inconsistency of the complementary intervals. An inconsistency is manifested by at least one zero set interval (lower bound above upper bound). Table 2 gives the details of the VIA algorithm extracting rules from a trained ANN.

The technique of "growing" rules from training patterns is termed *specific–to–general* in [10]. It is achieved by randomly increasing an input interval and retaining or rejecting the new interval size depending on the detection of a consistency or an inconsistency, respectively. Evidently, the resulting hypercubes are non–deterministic and dependent on the order of changed intervals. This is the reason why we used all training patterns, as similar training patterns can generate hypercubes of rather different volume.

The opposite technique is called "general–to–specific" and starts with the most general rule(s) which are broken down

VIA [8]
⇒ *Train an MLP* **Evolve generalized MLP using GAs** **ANN fitness is based on conventional training** ⇒ Use **each** training pattern $\mathcal{I}$ as a (degenerated) hypercube $\mathcal{H}$ and (randomly) grow a consistent $\mathcal{H}$ of maximal volume *only use training patterns not in an already extracted* $\mathcal{H}$ ⇒ Form a rule for each $\mathcal{H}$ ⇒ **Simplify rules by eliminating hypercubes** **completely contained in others**

Table 2: VIA rule extraction (*emphasized* indicates original algorithm, **bold** indicates our adaptations)

to more specific rules, if an inconsistency is detected for the general rule(s).

3. EVOLUTIONARY ANN DESIGN

For the evolutionary design of the ANN architecture, specifically, generalized multi–layer perceptrons, we have employed the netGEN system [7]. This collection of tasks uses a genetic algorithm (GA) [4] to generate problem–adapted ANNs using a direct encoding method, i.e., each connection of the network is represented in the chromosome. Additionally, the use of *neuron markers* (a bit field in the chromosome) selectively switching on and off neurons increases the variability of ANN architectures ranging from no hidden neurons to a preset maximum number. The candidate networks are trained in parallel (one per machine and/or processor) by standard error–back–propagation. ANN fitness is based on the classification accuracy on a validation set (model fitness). In order to drive evolution towards networks of low complexity a complexity term $\mathcal{E}_c$ has been added to the fitness function (complexity fitness). The composite fitness function $\mathcal{F}$ is given by a weighted sum of model fitness and complexity fitness:

$$\mathcal{F} = \alpha_1(1 - \mathcal{E}_m) + \alpha_2 \frac{1}{1 + \mathcal{E}_c} \qquad (1)$$

with $\alpha_1 + \alpha_2 = 1.0$, and $\mathcal{E}_m$ being the model error. Earlier work [6] showed that α_2 in the range of $0.001 - 0.01$ is sufficient to evolve ANNs of low complexity. Thus, we set $\alpha_2 = 0.01$. The model error for classification tasks is given by $\mathcal{E}_m = \frac{e_v}{n_v}$, where e_v is the number of misclassifications on the validation set, and n_v its respective size. The complexity term is simply defined by $\mathcal{E}_c = |C|$ with C being the set of all connections of the network.

GA and ANN Parameters

The following GA and ANN parameters have been used with all the experiments in this paper:
GA Parameters: Population Size = 50, Generations = 50, Crossover Probability $p_c = 0.6$, Mutation Probability $p_m = 0.005$, Crossover = 2–Point, Selection Method = Binary Tournament.
ANN Parameters: Network Type = Generalized Multi–Layer Perceptron, Activation Function (all neurons) = Sigmoid, Output Function (all neurons) = Identity, Training = Error–Back–Propagation, Hidden Neurons (max 7), Learning Parameter η (max 1.0), Number of Training Epochs (max 1000).

4. BENCHMARK PROBLEM

The three MONK's problems are discrete classification problems, where the learning system is given the task to retrieve specific concepts describing states of an exemplary robot. The robot's six attributes and its values are: `head_shape(round, square, octagon)`, `body_shape(round, square, octagon)`, `is_smiling(yes, no)`, `holding(sword, balloon, flag)`, `jacket_color(red, yellow, green, blue)`, `has_tie(yes, no)`. For the input vector of an ANN this translates to 17 components, when each value of an attribute is associated with a boolean input neuron. This encoding constrains the number of possible input vectors, as the robot's attributes are always defined to have a single value, e.g., the jacket color cannot be red **and** green.
The target concepts for the three problems are:
M1: `(head_shape = body_shape) or (jacket_color = red)`
M2: Exactly two of the six attributes have their **first** value
M3: `(jacket_color = green and holding = sword) or (jacket_color = not blue and body_shape = not octagon)`
The decision of the network is based on the state of its single output neuron o. If $o > 0.5$ the target concept is described by the input pattern, otherwise it is not (o is trained to be 0 or 1). The training data set for M1, M2, and M3 contain 124, 169, and 122 patterns, respectively. The training data set for M3 had some training noise added to it (6 misclassified patterns). The validation data set comprises all possible concepts (432 patterns) [8].

5. EXPERIMENTAL RESULTS

For each of the three MONK problems we evolved networks in 25 evolutionary runs (each run started with different random seed). Usually, an ANN with perfect generalization has already been found in the first generations and from then on the evolution process pruned superfluous neurons and/or connections. It should be noted that neither in [8], nor in [11], a perfect network (classifying all test set patterns correctly) for M3 could be found, although, more advanced training algorithms have been employed in these works (we used standard back–propagation with evolved learning parameter η).
Out of each run's best ANN (a total of 25) we selected the networks with the minimal and maximal number of connections. For comparisons we also ran evolution 25 times without complexity regularization ($\alpha_2 = 0.0$) and chose the most complex network. No single evolutionary run for all three problems generated an imperfect network. In the following the evolved ANNs are labeled as ev$n$_c, where n and c are the number of hidden neurons and connections, respectively. Further, the fully connected one–hidden–layer networks given in [8] [1], labeled as th$n$_c, are also compared.

[1] Thanks to the author having listed every single weight of the trained networks.

Figure 1 presents the compared networks. Tables 3, 4, and 5, give ANN and rule base statistics and the most compact rule sets extracted by MofN and VIA.

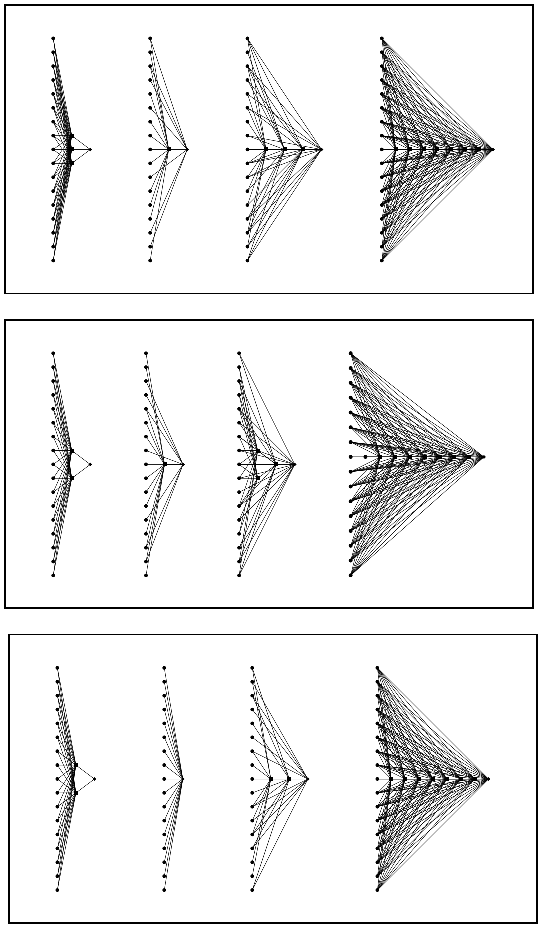

Figure 1: ANN architectures for the three Monk problems, from top to bottom *Monk–1* (th3-54, ev1-20, ev3-51, ev7-163), *Monk–2* (th2-36, ev1-21, ev3-48, ev7-163), *Monk–3* (th2-36, ev0-17, ev2-31, ev7-162).

With Monk–1 all four network architectures enabled the extraction of perfect rule sets (classifying the whole test set correctly). The leanest ANN ev1-20 generated the rule sets of the smallest size. While this is not surprising for the decompositional RE technique MofN, it shows that VIA is also dependent on the specific architecture. However, from above results it cannot be generalized that smaller networks result in smaller VIA rule sets. But it may be hypothesized that there exist well–pronounced structural optima, as the ANN ev3-30 (not shown in Table 3) resulted in a perfect VIA rule set containing 12 rules.
The formulation of the Monk–2 concept is well–suited for MofN expressions but less for VIA rules. This is reflected by the results shown in Table 4. Moreover, VIA could not find rule sets covering the whole test set space, as a certain number of patterns could not be classified. The incompatibility of VIA expressions (hypercubes) with the Monk–2 concept was also reported in [8]. The smallest and perfect MofN rule set was generated by the leanest network ev1-21. The more complex nets did not allow for

MONK1					
	Test Set Errors			Rule Sets	
	ANN	MofN	VIA	MofN	VIA
th3-54	0	0	0	19/34	15/37
ev1-20	0	0	0	11/20	19/57
ev3-51	0	0	0	14/17	18/49
ev7-163	0	0	0	26/70	16/43

Table 3: *Monk–1* Test set errors of ANNs and extracted rule bases and rule set sizes given as `number of rules / antecedents`.

MONK2					
	Test Set Errors			Rule Sets	
	ANN	MofN	VIA	MofN	VIA
th2-36	0	0	0_{18}	13/36	62/304
ev1-21	0	0	0_{28}	9/19	51/238
ev3-48	0	71	0_{39}	25/48	70/346
ev7-163	0	12	0_{21}	50/163	51/242

Table 4: *Monk–2* Test set errors of ANNs and extracted rule bases and rule set sizes given as `number of rules / antecedents` (the tiny numbers indicate patterns not classified by the VIA rule set).

the extraction of a perfect MofN rule set.
A similar picture is presented by the Monk–3 results. The evolved networks of low complexity give rise to perfect rule sets (note that the VIA rule set for th2-36 is slightly smaller, because it misses some target concepts due to the imperfect ANN). The MofN rule set for ev0-17 contains 6 antecedents which comes close to the minimal four antecedents in the target concept. Actually, the target concept can be derived from the MofN rule set by inspection as is shown in Table 6. This is only possible, because of the very compact rule set extracted from the low complex ANN.
The extracted VIA rule set presented in Table 7 even represents the exact target concept without any further transformation. Due to the nature of VIA separate rules for concepts agreeing (positive rules) and disagreeing (negative rules) with the target concept have been extracted. In order to enhance readability the extracted hypercubes have been rewritten with the monk attributes (e.g., `(jacket_color =`

MONK3					
	Test Set Errors			Rule Sets	
	ANN	MofN	VIA	MofN	VIA
th2-36	12	12	12_{12}	5/16	3/5
ev0-17	0	0	0	5/6	5/9
ev2-31	0	0	0	12/21	5/9
ev7-162	0	6	0_{3}	34/141	33/99

Table 5: *Monk–3* Test set errors of ANNs and extracted rule bases and rule set sizes given as `number of rules / antecedents` (the tiny numbers indicate patterns not classified by the VIA rule set).

MofN [9]
Extracted rule set
Out neuron true, if
$0.1 < -12.6 \times numberTrue\{15\} + 8.5 \times numberTrue\{14\} +$
$+3.0 \times numberTrue\{13\ 12\ 9\} + -9.3 \times numberTrue\{6\}$
Reformulation
True, if
$1of\{13\ 12\ 9\}\ AND\ (NOT\ 1of\{15\})\ AND\ (NOT\ 1of\{6\}$
$1of\{14\}\ AND\ (NOT\ 1of\{15\})\ AND\ (NOT\ 1of\{6\}$
$1of\{14\}\ AND\ 1of\{13\ 12\ 9\}\ AND\ (NOT\ 1of\{15\})$
$1of\{14\}\ AND\ 2of\{13\ 12\ 9\}\ AND\ (NOT\ 1of\{6\})$
Simplification
$(14\ AND\ 9)\ OR\ (NOT\ 15\ AND\ NOT\ 6) \equiv$ **M3**

Table 6: Extracted MofN rule set (M3, ev0-17) and its transformation to the exact target concept (the integer numbers correspond to the sequence of attributes given in Section 4).

`blue)` would be a hypercube with attribute 15 set to [1, 1] and all others to [0, 1]).

VIA [8]
Extracted positive rules $\equiv$ **M3**
`(body_shape = not octagon) and (jacket_color =`
`= not blue)`
`(holding = sword) and (jacket_color = green)`
Extracted negative rules
`(jacket_color = blue)`
`(body_shape = octagon) and (jacket_color = not green)`
`(body_shape = octagon) and (holding = not sword)`

Table 7: Extracted VIA rule set (M3, ev0-17) representing the exact target concept (positive rules).

Generally, the evolved ANNs of low complexity exhibit better generalization abilities than more complex networks which is an established ANN research fact. The two investigated RE algorithms are able to fully transfer the knowledge accumulated in these networks' structure (by evolution) and weights (by training) to a symbolic rule set. As a consequence the extracted rule sets are not only accurate, but also smaller than those derived of more complex ANNs which in turn enhances comprehensibility of the rule base.

6. OUTLOOK

Based on the observation that there might exist very specific ANN architectures for the extraction of rule sets of minimal size (maximal comprehensibility), we will continue this work towards ANN evolution with network fitness based on the accuracy and size of the extracted rule set. Also, RE should be compared to symbolic data mining methods in order to check the hypothesis that certain problems or problem classes can be solved easier in the ANN domain. Further research topics include utilization and comparison of other RE algorithms, ANN evolution and RE for real world problems, improvement of existing RE techniques, and experiments towards ANN–based RE, a process inspired by the way humans (arguably) derive their higher order concepts.

7. ACKNOWLEDGEMENTS

We wish to thank the *Austrian Center for Parallel Computation* (ACPC), Group Salzburg, for making available a cluster of DEC AXP workstations for this research.

8. REFERENCES

[1] R. Andrews, J. Diederich, and A. Tickle. A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks. *Knowledge Based Systems*, 0(0):0, 1995.

[2] Simon Haykin. *Neural Networks. A Comprehensive Foundation.* MacMillan, 1994.

[3] Xin Yao. Evolutionary Artificial Neural Networks. *International Journal of Neural Systems*, 4(3):203–222, 1993.

[4] Melanie Mitchell. *An Introduction to Genetic Algorithms.* Complex Adaptive Systems. MIT Press, Cambridge, MA, 1996.

[5] David E. Rumelhart, Bernard Widrow, and Michael A. Lehr. The Basic Ideas in Neural Networks. *Communications of the ACM*, 37(3):87–92, March 1994.

[6] Helmut A. Mayer, Reinhold Huber, and Roland Schwaiger. Lean Artificial Neural Networks - Regularization Helps Evolution. In *Proceedings of the 2nd Nordic Workshop on Genetic Algorithms and their Applications*, pages 163–172, 1996.

[7] Helmut A. Mayer. Symbiotic Coevolution of Artificial Neural Networks and Training Data Sets. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *5th International Conference Parallel Problem Solving from Nature*, pages 511–520, Berlin, Germany, September 1998. Springer.

[8] Sebastian B. Thrun. Extracting Provably Correct Rules from Artificial Neural Networks. Technical Report IAI–TR–93–5, University of Bonn, Institut für Informatik III, D–53117 Bonn, May 1993.

[9] Geoffrey G. Towell and Jude W. Shavlik. Extracting Refined Rules from Knowledge–Based Neural Networks. *Machine Learning*, 13:71–101, 1993.

[10] Sebastian Thrun. Extracting Rules from Artificial Neural Networks with Distributed Representations. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems 7*, Cambridge, MA, 1995. MIT Press.

[11] Jay A. Alexander and Michael C. Mozer. Template–Based Procedures for Neural Network Interpretation. *Neural Networks*, 0(0):0, 1998.