Bayesian Case-Based Reasoning with Neural Networks

Petri Myllymäki Henry Tirri*

University of Helsinki
Department of Computer Science
Teollisuuskatu 23
SF-00510, Helsinki, FINLAND
email: myllymak@cs.Helsinki.FI, tirri@cs.Helsinki.FI

Abstract

Given a problem, a case-based reasoning (CBR) system will search its case memory and use the stored cases to find the solution, possibly modifying retrieved cases to adapt to the required input specifications. In this paper we introduce a neural network architecture for efficient case-based reasoning. We show how Pearl's probability propagation algorithm [12] can be implemented as a feedforward neural network and adapted for CBR. In our approach the efficient indexing problem of CBR is naturally implemented by the parallel architecture, and heuristic matching is replaced by a probability metric. This allows our CBR to perform theoretically sound Bayesian reasoning. We also show how the probability propagation actually offers a solution to the adaptation problem in a very natural way.

1 Introduction

> *..., it must be insisted that the common-sense distinction between the informal process of learning to play a game and the formal process of learning its rules is valid.*
> *-Mortimer Taube*

Artificial intelligence research has focused on finding methods that allow computer programs to incorporate knowledge by manipulating high-level representations of relevant information. A typical example of this approach is expressing knowledge by rules. The underlying assumption is that these representations are compact abstractions of the known individual instances of the knowledge concept to be encoded. These abstractions can be static or dynamic, i.e., they can be compiled into the program directly, or adaptively synthesized during the application execution with a learning procedure. This approach is applicable for representing well-organized knowledge. However, such a summarization approach has faced substantial problems in applications where the concepts required for the knowledge are highly interconnected and have a large amount of irregularities (exceptions), e.g., "common sense".

One source of the current problems is the confusion between representing the constraints of the desired computation and the computation itself. For example it is easy to represent the rules of chess formally, but it is much harder to represent the knowledge to play chess well. So far the only method to represent the latter, less structured type of knowledge is essentially based on "memorizing" examples of parts of the computations, e.g., in the chess case board patterns. The computation itself then concentrates on efficiently manipulating this vast set of examples by pruning the search space, finding approximations etc. Interestingly this approach is actually consistent with some arguments of the AI critics [4], who argue that humans develop an expertise in areas that require "common sense" by an adaptive process that proceeds from abstract to concrete, i.e., from simple representations, such as context-free rules to complex case-based representations. Notable, this is the exact reverse of the traditional procedure of searching for more and more abstract concept representations. However, recently there has been a growing interest to so called case-based reasoning methods [3], notion originally introduced by Schank [15].

*This research was supported by Technology Development Center (TEKES) and Honkanen Foundation.

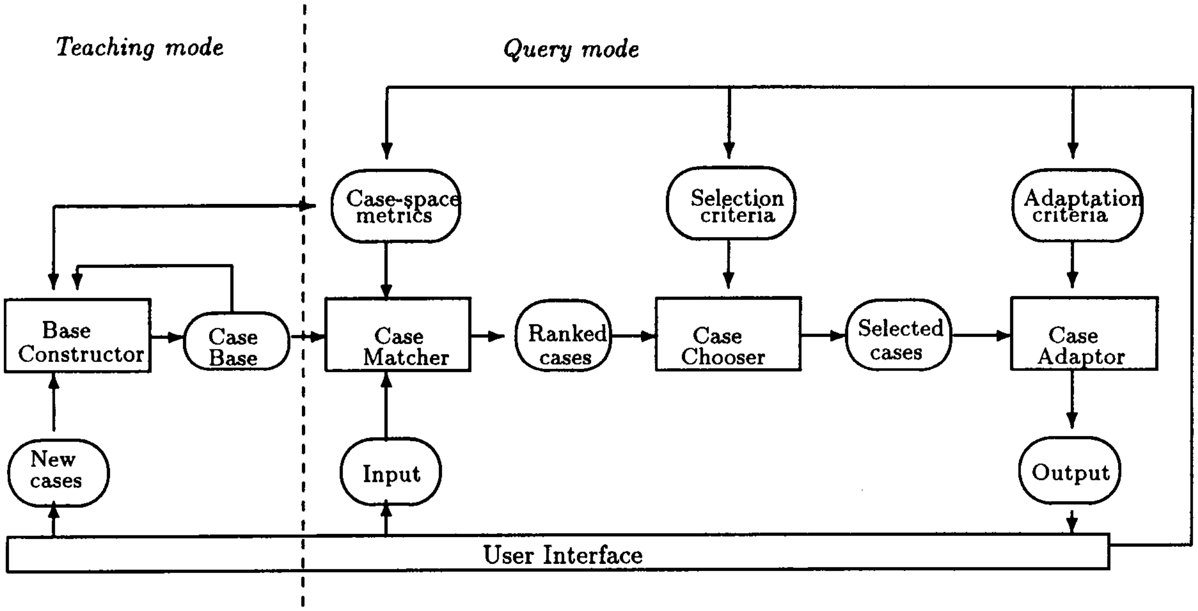

Figure 1: CBR system architecture

In case-based reasoning (CBR) paradigm the dynamic case memory is central to the reasoning process (see the process model in Figure 1) — learning is an inherent part of the process. Although the idea of using a case memory is simple in principle, there are many difficult problems related to case indexing, similarity metric in case matching and case adaptation. For example much of the published work has concentrated on applying machine learning methods for indexing cases in order to avoid costly comparison of the input with the large set of cases in the case memory. Similarly developing appropriate metric for matching has in practice lead to heuristic solutions which are hard to justify theoretically. In this paper we discuss the use of feedforward neural networks to implement a case based reasoning system with a probability metric, which allows us to do theoretically sound Bayesian reasoning. Consequently we propose solutions to three of the problems in case based reasoning: case indexing (by the parallel neural architecture), case matching (by probability metric) and case adaptation (by propagating probabilistic information back to the input vector).

2 Case-Based Reasoning and Neural Networks

During the past few years a rediscovered computational paradigm of neural computing has spawned a large amount of research activities [1, 6, 14]. Much of the excitement surrounding this approach has been inspired by the rich possibilities inherent in ideas of distributed representation of knowledge, reasoning based on constraint satisfaction, and neurally-realistic cognitive modeling. As connectionism has challenged the physical symbol system modeling of traditional artificial intelligence, there is an ongoing debate of the respective roles of these two approaches in explaining intelligence [13, 16]. However, in spite of the debate there are clear connections between both of the approaches. Case-based reasoning is one of the areas where one can combine the ideas from these apparently diverse approaches fruitfully.

What is the appeal in neural networks for case-based reasoning? First of all neural networks are constructed from a large amount of elements with an input fan order of magnitudes larger than in computational elements of conventional architectures. This means that they can be used for distortion tolerant storing of a large number of cases (represented by high dimensional vectors) by making single elements "sensitive" to a stored item, i.e., to produce a high output for particular subregions in the input space. Hence neural computing models offer an ideal architectural framework for the methods in case-based reasoning, which are based on using a set of high-dimensional cases stored in a knowledge base. Secondly, as there is nothing that resembles a shared memory, the neural computing architecture is inherently parallel, and each element can perform comparison of its input against the value stored in the interconnections independently from the others. This offers a linear speed-up in the comparison process with respect to the number of computational elements available. The case indexing problem is thus addressed directly at the architectural level where matching can be performed by using the available parallelism. This kind of a *memory-based reasoning* approach allows matching of the input against millions of stored cases efficiently [7].

In addition, as in many situations the result of the individual comparisons can be merged to achieve a lower dimensional output, such as a binary decision, the resulting computing structure is fault-tolerant as loss of a single element (case) does not in general have a great effect on the overall result. Consequently neural computing models offer an architecture that, at least in principle, matches already in its structure many of the requirements needed for representing case memory: efficient storing of cases and fast mechanisms for comparing high-dimensional patterns with tolerance for input distortions. On the other hand, neural architectures do not offer directly solutions to the problem of choosing proper metrics for case matching and case adaptation tasks. Thus, to achieve a full CBR implementation, we need to augment the neural architecture with additional concepts. To avoid resorting to ad hoc heuristic solutions in the reasoning process, we propose that one uses methods developed for Bayesian probabilistic reasoning systems [12], [10]. This approach, which we call *Bayesian case-based reasoning*, is discussed in the next section.

3 Bayesian Case-Based Reasoning

Recent progress in the theory of graphical belief network representations has made it possible to create rigorous new algorithms for belief updating (see e.g.[12, 10]). Unfortunately, the problem is in the general case known to be NP-hard [2]. Perhaps the most promising approach to overcome this problem is to use a stochastic simulation scheme called Gibbs sampling [5] for approximating the outcome of the updating process. In our earlier work [11, 9] we presented schemes for implementing Gibbs sampling on a neural network architecture. However, the problem of determining the so called annealing schedule has proven very hard in practise, resulting to slow convergence of the algorithm. On the other hand, for a restricted class of belief networks, singly-connected networks, there exists a polynomial time algorithm for belief updating, developed by Pearl [12]. This algorithm is exploited in another approach [8] to the general belief updating problem, where a given belief network is first transformed to a singly-connected network, which is then updated by using Pearl's algorithm. However, as the problem is NP-hard, the transformation process may take an exponential time. Here we suggest a different approach: we restrict ourselves to simple singly-connected belief networks, trees, and show how trees, in spite of their simple structure, can be used to represent the case-based reasoning framework. Furthermore, in the next section we show how to efficiently implement Pearl's belief updating algorithm for trees as a massively parallel feedforward neural network.

In our framework, the knowledge of the problem domain is coded using m attributes $A_1, \ldots, A_m$, where an attribute A_i has n_i possible values, $a_{i1}, \ldots, a_{in_i}$. Our observations of the world consist of binary vectors

$$(\underbrace{f_1(a_{11}), \ldots, f_1(a_{1n_1})}_{A_1}, \underbrace{f_2(a_{21}), \ldots, f_2(a_{2n_2})}_{A_2}, \ldots, \underbrace{f_m(a_{m1}), \ldots, f_m(a_{mn_m})}_{A_m}),$$

where the characteristic function $f_i(a_{ij})$ is 1 if A_i has value a_{ij}, otherwise $f_i(a_{ij})$ is 0. A case C_k is a "prototype" representation of a class of (in some sense) similar observations, and is coded as a vector

$$C_k = (P_k(a_{11}), \ldots, P_k(a_{1n_1}), P_k(a_{21}), \ldots, P_k(a_{2n_2}), \ldots, P_k(a_{m1}), \ldots, P_k(a_{mn_m})),$$

where $P_k(a_{ij})$ expresses the likelihood for attribute A_i to have value a_{ij} in the class k. Our case base C consists of l cases $C_1, \ldots, C_l$, each of which is provided with an unique label c_k. Here we do not address the problem of choosing the cases C_k, but we assume that they are, for example, defined by a human expert or derived from a large database of observations by statistical clustering methods.

In our Bayesian approach, the case attributes A_i are treated as random variables. Similarly, the case base can also be regarded as a random variable C, with possible values $c_1, \ldots, c_l$. We can now define $P_k(a_{ij})$ as the conditional probability $P(A_i = a_{ij} | C = c_k)$, which in the sequel will be written in an abbreviated form $P(a_{ij}|c_k)$. In addition to the values $P_k(a_{ij})$, each case must be provided with a prior probability $P(c_k)$.

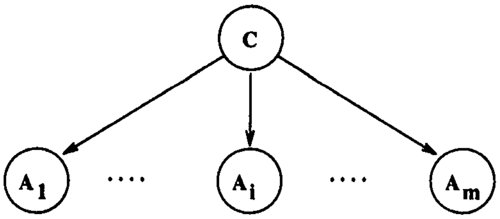

Figure 2: Belief network representation of the case base.

This probability can be estimated by the proportional number of occurrences of class k, if a database of observations is available. Similarly, the probabilities $P(a_{ij}|c_k)$ can be estimated by occurrences of the value a_{ij} inside class k.

We assume that the given cases are *complete*, i.e., all the values $P_k(a_{ij})$ are given for each case C_k. If the user is unable to provide complete cases, the missing probabilities can be filled in by using the uniform probability distribution (if we do not know the value of an attribute, we assume all the values to be equally probable). Alternatively, the user may also define another a priori distribution for the missing cases, if this kind of information about the attributes is available. After storing the labeled cases, we can obviously retrieve any value $P_k(a_{ij})$, if we know the label (index) of the case C_k. In the Bayesian framework this means that *all the variables A_i are conditionally independent of each other, given the value of the variable C*. What this means is that the Bayesian network corresponding to this representation is a tree, where variable C is the root of the tree, and variables A_i form the leaves (see Figure 2).

An input case is created by permanently setting ("clamping") the values of some attributes; let us denote the set of clamped attribute values by I. The rest of the values are initialized according to uniform (or some other a priori) distribution. Given this new case vector (which is of course usually not in our case base C), we can now use Pearl's algorithm to compute the probability $P(c_k|I)$ for each case C_k. Hence we are able to do the case matching task, using the probability measure as the metrics of our system. What is more, Pearl's algorithm allows us also to compute probabilities $P(a_{ij}|I)$ for all the unclamped values $a_{ij} \notin I$. Consequently, this algorithm can also be used for the case adaptation task of our CBR system.

4 Neural Implementation of Bayesian CBR

We now show how to construct a 6-layer feedforward neural network which performs the computations of Pearl's algorithm in parallel. In practise, it is possible to implement the system as a 3-layer network with feedback, but we use here a feedforward architecture for clarity. The structure of the network is determined by the number of the cases (l), the number of the attributes (m), and the number of the values of the attributes ($n_1, \ldots, n_m$) (see Figure 3). The total number of nodes in the resulting network is

$$3\sum_{i=1}^{m} n_i + 2ml + l,$$

and the number of arcs in the network is given by

$$l\sum_{i=1}^{m} n_i + lm + 2lm + l\sum_{i=1}^{m} n_i + 2\sum_{i=1}^{m} n_i = 2(l+1)\sum_{i=1}^{m} n_i + 3lm.$$

During the network computation process, each node X computes its activation value $S(X)$ using incoming messages, and sends the computed value further through the arcs leading to nodes in the upper layers. This activation propagation starts when the user sets the values $S(a_{ij})$ for the nodes in layer 1. According to the idea of *virtual evidence* [12], if there exists some initial evidence e for the value a_{ij} of the attribute A_i, the value $S(a_{ij})$ should be set equal to the probability $P(e|A_i = a_{ij})$. Total ignorance of the correct value of A_i

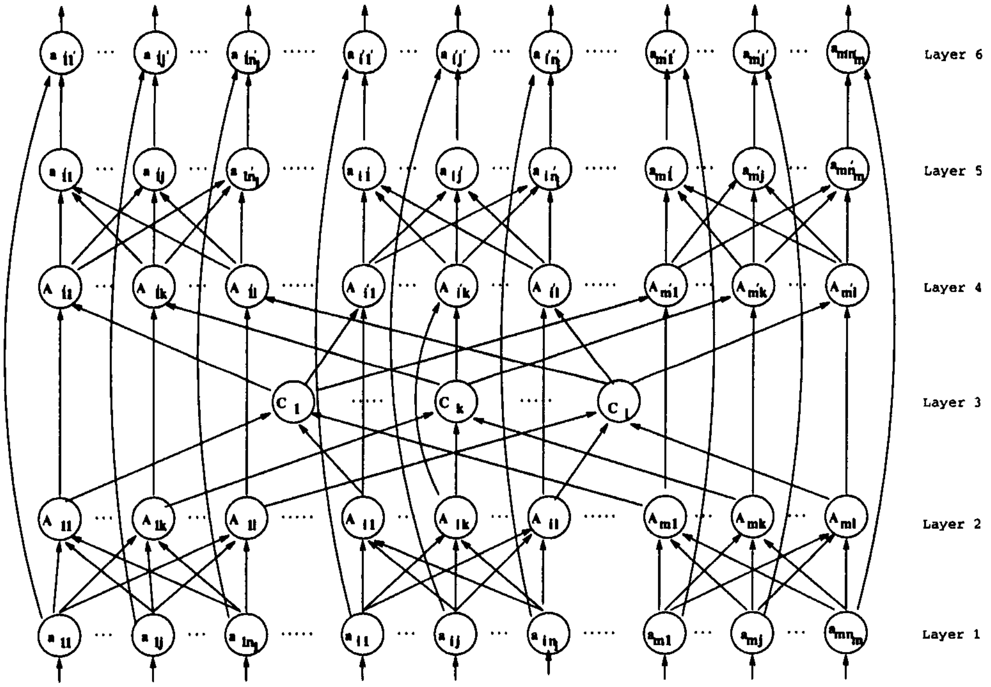

Figure 3: The neural network implementing Bayesian CBR.

is represented by setting all the values $S(a_{i1}), \ldots, S(a_{in_i})$ to be equal, for example 1. If the value of A_i is known to be a_{ih} for certain, then $S(a_{ih})$ should be set to 1, and the values $S(a_{ij})$ to 0, for all $j \neq h$.

Intuitively the computation consists of two phases. The initial phase, using the first three layers of the network, performs case matching. The third layer activation function gives a matching score for each of the cases (i.e., to nodes C_i), thus these activation values can be directly used for classification, if needed. In the second phase, the last three layers perform the probability propagation back to the attribute values, thus complementing the attribute value vectors based on the "winning" cases. In general this allows many stored cases to contribute to the adaptation by the amount justified by their original matching. In the general framework of Figure 1 this approach corresponds to situation, where the case-space metrics and adaptation criteria coincide.

In the following we illustrate more closely how the activation propagation process proceeds from layer to layer.

Layer 1: The first layer contains one node for each of the possible attribute values a_{ij}, altogether $\sum_{i=1}^{m} n_i$ nodes. The value $S(a_{ij})$ is either given by the user, or initialized to the defined a priori value. Thus we have restricted our discussion to attributes with discrete values. However, this is a very natural restriction in the context of expert systems, which currently is the main application area of CBR.

Layer 2: Layer 2 consists of m groups of nodes, each of which has l individual nodes $A_{i1}, \ldots, A_{il}$, making the total number of nodes in this layer ml. Each node A_{ik} has n_i arriving arcs from all the nodes $a_{i1}, \ldots, a_{in_i}$. The weight $W(A_{ik})_j$ from node a_{ij} to node A_{ik} is $P(a_{ij}|c_k)$, i.e., the conditional probability that the attribute A_{ik} has value a_{ij} given an observation from class C_k. The activation value of node A_{ik} is computed by

$$S(A_{ik}) = \sum_{j=1}^{n_i} W(A_{ik})_j S(a_{ij}).$$

Layer 3: In layer 3, there is one node for each of the l classes. Each node C_k receives input from m nodes in the layer 2, $A_{1k}, \ldots, A_{mk}$. The activation value of node C_k is computed by

$$S(C_k) = P(c_k) \prod_{i=1}^{m} S(A_{ik}).$$

This activation gives a score for each of the stored cases. The constant value $P(c_k)$ is assumed to be stored in the node C_k.

Layer 4: This layer is identical to layer 2, consisting of ml nodes. Each node A'_{ik} receives input from two nodes: the corresponding node A_{ik} in the layer 2, and the node C_k in layer 3. The activation value is computed by

$$S(A'_{ik}) = S(C_k)/S(A_{ik}).$$

Layer 5: Layer 5 is identical to layer 1. Each node a'_{ij} has incoming arcs from l nodes, $A'_{i1}, \ldots, A'_{il}$. The weight $W(a'_{ij})_k$ from node A'_{ik} to node a'_{ij} is $P(a_{ij}|c_k)$. The activation value is computed by

$$S(a'_{ij}) = \sum_{k=1}^{l} W(a'_{ij})_k S(A'_{ik}).$$

Layer 6: The last layer is identical to layers 1 and 5. Each node a''_{ij} receives two inputs, one from the corresponding node a'_{ij} in layer 5, and one from the node a_{ij} in layer 1. The activation value is computed by

$$S(a''_{ij}) = S(a'_{ij})S(a_{ij}).$$

This can be understood as a "correction" to the original values assuming that the matching cases have prediction value for unidentical, but similar cases.

Using the notation of Pearl in [12], the task of the layer 2 is to compute the m values $\lambda_{A_i}(c_k)$, for each of the l cases. As $\lambda(c_k)$ is defined as $\lambda(c_k) = \prod_{i=1}^{m} \lambda_{A_i}(c_k)$, the activation value of node C_k is $S(C_k) = P(c_k)\lambda(c_k)$. Pearl has proved that this is equal to $\alpha P(c_k|I)$, where α is a normalizing constant. The actual probabilities can now be retrieved easily by normalizing the values $S(C_k)$:

$$P(c_k|I) = S(C_k) / \sum_{h=1}^{l} S(C_h).$$

In a similar way, layer 4 computes the terms $\pi_{A_i}(c_k)$, layer 5 the terms $\pi(a_{ij})$, and the activation values of nodes in layer 6 are $S(a''_{ij}) = \lambda(a_{ij})\pi(a_{ij}) = \alpha P(a_{ij}|I)$, where α is again a (different) normalizing constant. The actual probabilities are retrieved again by normalization:

$$P(a_{ij}|I) = S(a''_{ij}) / \sum_{h=1}^{n_i} S(a''_{ih}).$$

Naturally, the two normalization tasks can also be performed in parallel on a neural network by using two extra layers of units.

5 Conclusion and future work

We have presented a neural network architecture for case-based reasoning by implementing Pearl's probability propagation as a multi-layer feedforward network. The immediate advantages of this approach are efficient case matching inherent in the parallel architecture, theoretically sound Bayesian interpretation of the case-space metrics and one possible solution to case adaptation via similar probability propagation as the one used for case matching. A prototype implementation of this kind of a CBR system is currently under construction. Our next goal is to study the problem of learning: what is the proper choice of the cases (in the presence of noise the best strategy is not necessary to store all the cases encountered), or how does one determine the conditional probabilities $P(a_{ij}|c_k)$? Initially it can be assumed that such probabilities are estimated by the expert, but it is clear that one can use various statistical clustering techniques for this purpose. We are also studying the use of more elaborate mechanisms for the case adaptation task, which may be useful in complex problem domains.

References

[1] J.A. Anderson and E. Rosenfeld (Eds.), *Neurocomputing.* Foundations of research. The MIT Press, 1988.

[2] G.F. Cooper, Probabilistic Inference using Belief Networks is NP-hard. Technical Report KSL-87-27, Stanford University, Stanford, California.

[3] DARPA, *Proceedings of the Workshop on Case-Based Reasoning 1988-1990.* Morgan Kaufmann, San Mateo.

[4] H. Dreyfus and S.Dreyfus, *Mind over Machine — The power of human intuition and expertise in the era of the computer*, Basil Blackwell, 1986.

[5] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 6 (1984), 721–741.

[6] R. Hecht-Nielsen, Neurocomputer applications. Pp. 445–451 in: *Neural computers* (R. Eckmiller and C.v.d.Malsburg Eds.), Springer-Verlag, 1987.

[7] H. Kitano and M. Yasunaga, Wafer Scale Integration for Massively Parallel Memory-Based Reasoning. Pp. 850 – 856 in: *Proc. of the Tenth National Conference on Artificial Intelligence (San Jose, July 1992)* AAAI Press/The MIT Press, Menlo Park, 1992.

[8] S. L. Lauritzen and D. J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Stat. Soc., Ser. B* 1989. Reprinted as pp. 415–448 in: *Readings in Uncertain Reasoning* (G. Shafer and J. Pearl, eds.). Morgan Kaufmann, San Mateo, 1990.

[9] P. Myllymäki and P. Orponen, Programming the Harmonium. Pp. 671–677 in: *Proc. of the International Joint Conf. on Neural Networks (Singapore, Nov. 1991), Vol. 1.* IEEE, New York, NY, 1991.

[10] R. Neapolitan, *Probabilistic Reasoning in Expert Systems.* Wiley Interscience, New York, 1990.

[11] P. Orponen, P. Floréen, P. Myllymäki, H. Tirri, A neural implementation of conceptual hierarchies with Bayesian reasoning. Pp. 297–303 in: *Proc. of the International Joint Conf. on Neural Networks (San Diego, CA, June 1990), Vol. I.* IEEE, New York, NY, 1990.

[12] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[13] S. Pinker and J.Mehler (eds.), *Connections and Symbols.* The MIT Press, Cambridge, England, 1988.

[14] D.E. Rumelhart and J.L.McClelland (Eds.),*Parallel distributed processing: explorations in the microstructures of cognition. Vol 1,2.* The MIT Press, 1986.

[15] R. Schank, *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge University Press, Cambridge, 1982.

[16] P. Smolensky, On the proper treatment of connectionism. *Behavioral and Brain Sciences* (11), pp. 1–74, 1988.