Pp. 144–154 in *Topics in Case-Based Reasoning*, edited by Stefan Wess, Klaus-Dieter Althoff and Michael M. Richter. Volume 837, Lecture Notes in Artificial Intelligence. Springer Verlag, 1994.

Massively Parallel Case-Based Reasoning with Probabilistic Similarity Metrics

Petri Myllymäki and Henry Tirri

University of Helsinki, Department of Computer Science*
P.O.Box 26, FIN-00014 University of Helsinki, Finland
email: Petri.Myllymaki@cs.Helsinki.FI, Henry.Tirri@cs.Helsinki.FI

Abstract. We propose a probabilistic case-space metric for the case matching and case adaptation tasks. Central to our approach is a probability propagation algorithm adopted from Bayesian reasoning systems, which allows our case-based reasoning system to perform theoretically sound probabilistic reasoning. The same probability propagation mechanism actually offers a uniform solution to both the case matching and case adaptation problems. We also show how the algorithm can be implemented as a connectionist network, where efficient massively parallel case retrieval is an inherent property of the system. We argue that using this kind of an approach, the difficult problem of case indexing can be completely avoided.

1 Introduction

In case-based reasoning (CBR) paradigm the dynamic case memory is central to the reasoning process (see the process model in Fig. 1) — learning is an inherent part of the process. Although the idea of using a set of representative instances as the basis for a reasoning system is simple in principle, there are many difficult problems related to constructing a case base from data (learning), case matching and case adaptation. Here we do not address the problem of choosing a suitable set of cases, but for our purposes we assume that they are defined by a human expert, or derived from a large database of observations by statistical clustering methods. We will focus on the central problems concerning the reasoning part of the CBR system: case matching and adaptation tasks. Furthermore, in this study we are only concerned with shallow, knowledge poor CBR where the knowledge is represented as an unstructured set of high dimensional case vectors. This type of *memory-based* or *instance-based* reasoning systems have recently shown very promising results in solving difficult real-world problems, such as speech-to-speech translation [5].

Much of the published work on CBR has concentrated on applying machine learning methods for case indexing, in order to avoid costly comparison of the input with the large set of cases in the case memory during the case matching

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

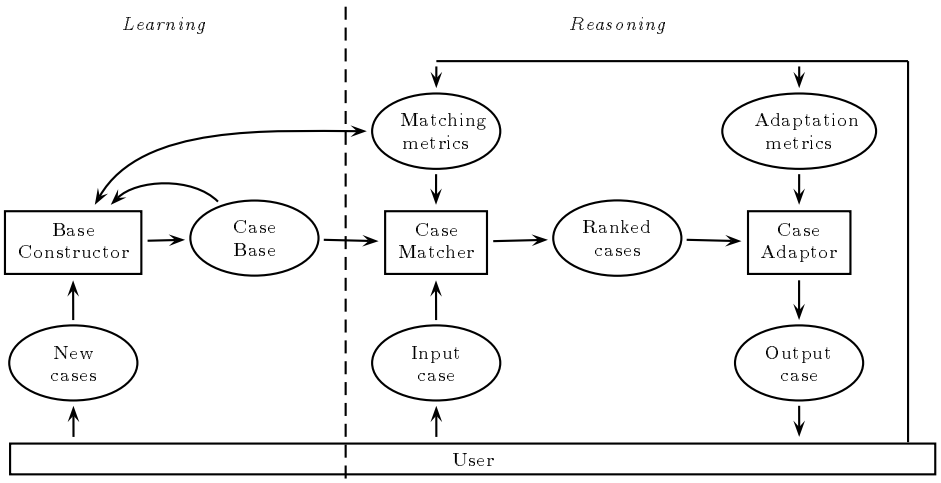

Fig. 1. The structure of our CBR system.

task [2]. We adopt an alternative approach and show how to construct a massively parallel implementation of CBR confirming to the so called *connectionist* architecture (see e.g. [14]). Connectionist 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 the set of connections of the elements can be used for distortion tolerant storing of 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. In our approach the case indexing problem is thus addressed directly at the architectural level where matching can be performed efficiently by using the available parallelism. On the other hand, chip level implementation of massive parallelism constraints the complexity of a single computing element to a limited set of operations and structurally simple local memory. Consequently case-based reasoning, with a knowledge base of high-dimensional cases as the basis for the reasoning process, offers a very natural computational framework for connectionist architectures. This is especially interesting as it has been recently argued that massive parallelism has an fundamental effect on the development of AI in general [5].

In addition to using connectionist models for avoiding the case indexing problem, we will also propose a uniform solution to the problem of choosing proper metrics for case matching and adaptation. Developing appropriate metrics for case matching and adaptation has in practice led to heuristic solutions which are hard to justify theoretically. The obvious disadvantages of such approaches are related to the difficulty of interpreting differences in the similarity of the various cases, and to the related problem of discovering the significance of the difference of the attribute values. For example in ad hoc solutions based on uncertainty

values, in most cases it is very difficult to interpret if e.g., the difference between values .9 and .99 is less important than the difference between .6 and .8. To avoid resorting to such ad hoc heuristic solutions in the reasoning process, we propose that one uses methods developed for Bayesian probabilistic reasoning systems [13], [11]. In Section 3, we show how our Bayesian reasoning framework, which we call *Bayesian case-based reasoning*, offers us a uniform similarity metrics for both the case matching and adaptation tasks. In Section 4, we present a massively parallel implementation of our Bayesian CBR system.

2 Massively Parallel Case-Based Reasoning

Our case base C is a collection of l case vectors, $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_l\}$, where each vector consists of n real-valued components, $\mathbf{c}_k = (c_{k1}, \ldots, c_{kn}), c_{ki} \in \mathbb{R}$. The CBR process starts when an *input case vector* $\mathbf{c}_0 = (c_{01}, \ldots, c_{0n})$ is presented to the system. The input case is usually incomplete, with at least one of the values c_{0i} missing. The purpose of the case matching module is to provide each case $\mathbf{c}_k$ with a *similarity rank* $S(\mathbf{c}_k)$ representing the similarity between the vectors $\mathbf{c}_k$ and $\mathbf{c}_0$. The case adaptation module then fills in the missing values of $\mathbf{c}_0$ with values consistent with the ranking of the cases (see Fig. 2).

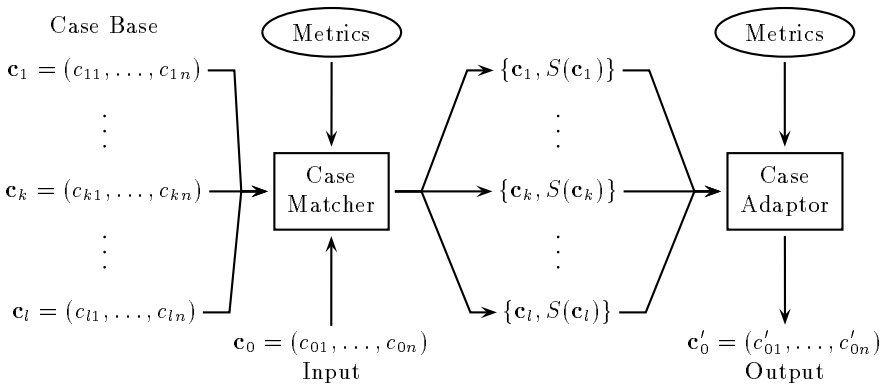

Fig. 2. The case matching and adaptation modules.

For a very general class of similarity metrics the case matching and case adaptation tasks can be implemented on a massively parallel connectionist architecture. As an illustrative example, let us consider a case matching task where the similarity rank for a case $\mathbf{c}_k$ is a function depending on the inner product of the case and input vectors: $S(\mathbf{c}_k) = F(\mathbf{c}_k \mathbf{c}_0)$. Let us now assign one processing *case unit* for each of the cases and one *value unit* for each of the components of the input case, and connect each value unit to all the case units with a directed arc. The weight of the connection from a value unit to a case unit is equal to the value of the corresponding component in the case definition. Hence each case

vector $\mathbf{c}_k$ is coded as a set of weights attached to the connections arriving to the case unit (see Fig. 3).

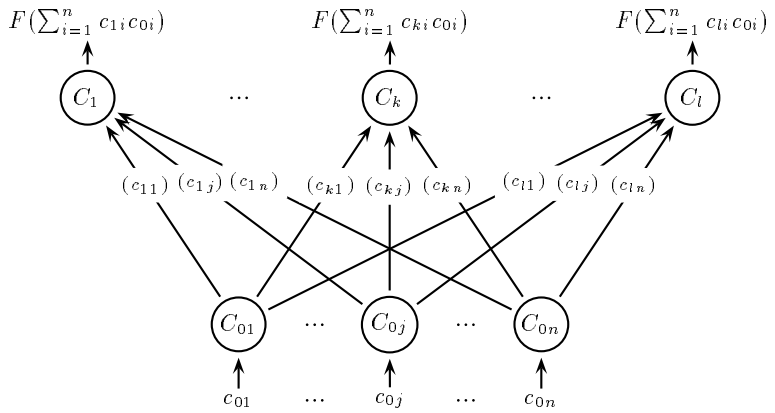

Fig. 3. An example of a massively parallel implementation of case matching.

Let the value of an input unit be either given in the input case assignment $\mathbf{c}_0$, or zero if the value is undefined. In our connectionist network, each unit sends its value to all the adjacent units, which sum all the incoming messages weighted by the connection strengths of the corresponding arcs. It is obvious that the total input of a case unit is the inner product $\mathbf{c}_k\mathbf{c}_0$. If each case unit now computes the function F using the input as the parameter, we have accomplished our goal: parallel computation of the case matching process. A similar construction can be presented for the case adaptation task. In Section 4 we present a massively parallel implementation of CBR where both the case matching and case adaptation tasks are performed using the same, undirected connectionist network.

As there is nothing that resembles a shared memory, the connectionist 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. This type of an approach allows matching of the input against millions of stored cases efficiently [4].

3 Bayesian Case-Based Reasoning

In our Bayesian framework, we assume that the problem domain knowledge is represented using values of m discrete attributes (random variables) $A_1, \ldots, A_m$. This is a very natural assumption in the context of expert systems, which currently is the main application area of CBR. If an attribute is not discrete, it can be discretized using standard quantization methods. Also all the cases $C_1, \ldots, C_l$

can be regarded as binary random variables, with $C_k = 1$ denoting the fact that case C_k is in question, $C_k = 0$ the opposite situation.

Let an attribute A_i has n_i possible values, $a_{i1}, \ldots, a_{in_i}$. A case vector $\mathbf{c}_k$ is coded as a vector

$$\mathbf{c}_k = (\underbrace{P_k(a_{11}), \ldots, P_k(a_{1n_1})}_{P_k(A_1)}, \underbrace{P_k(a_{21}), \ldots, P_k(a_{2n_2})}_{P_k(A_2)}, \ldots, \underbrace{P_k(a_{m1}), \ldots, P_k(a_{mn_m})}_{P_k(A_m)}),$$

where $P_k(A_i)$ expresses the probability distribution for the values of attribute A_i when case C_k is in question: $P_k(a_{ij}) = P(A_i = a_{ij} \mid C_k = 1)$. If all the components of a case vector are either 0 or 1, then this representation is identical to the normal deterministic attribute-value assignment type of a representation. However, in our model the components can also have values between 0 and 1, which means that a case can be regarded as a "prototype" representation of a class of (in some sense) similar instances.

Our case base can now be represented as a Bayesian belief network [13], consisting of variables $A_1, \ldots, A_m$ and $C_1, \ldots, C_l$ (see Fig. 4a). Intuitively speaking, a Bayesian network can be used to represent any probability distribution by showing all the direct dependencies between the random variables of the problem domain. The importance of Bayesian network representations lies in the way such a structure can be used as a compact representation for many naturally occurring distributions, where the dependencies between variables arise from a relatively sparse network of connections. In particular, given the Bayesian network representation of the dependencies (which implicitly define also the independencies), and for each random variable, a set of certain conditional probabilities, it is possible to construct algorithms for different probabilistic reasoning tasks.

The input to a Bayesian network is given by defining initial probability distributions for the values of each of the attributes. These initial probabilities $P_0(a_{ij})$ form the *input case* $\mathbf{c}_0$,

$$\mathbf{c}_0 = (P_0(a_{11}), \ldots, P_0(a_{1n_1}), P_0(a_{21}), \ldots, P_0(a_{2n_2}), \ldots, P_0(a_{m1}), \ldots, P_0(a_{mn_m})),$$

where according to the idea of *virtual evidence* [13], if there exists some initial evidence e for the value a_{ij} of the attribute A_i, the value $P_0(a_{ij})$ should be set equal to the probability $P(e \mid A_i = a_{ij})$. Total ignorance of the correct value of A_i is represented by using the uniform probability distribution. All the possible input case vectors can be regarded as values of an input case random variable C_0. Using this notation, we now define the metrics for the case matching and case adaptation tasks in our Bayesian framework as follows:

1. *Bayesian case matching*: The score for a case C_k is the conditional probability $P(C_k = 1 \mid C_0 = \mathbf{c}_0)$, i.e. the probability of case C_k being in question, given the current input value assignment.
2. *Bayesian case adaptation*: The components of the output vector are conditional probabilities of the form $P(A_i = a_{ij} \mid C_0 = \mathbf{c}_0)$. These components are computed using the probabilities produced by the Bayesian case matching task, and they can be understood as a "correction" to the original values of

the input case assuming that the matching cases have prediction value for unidentical, but similar cases.

In the general framework of Fig. 1, the Bayesian CBR approach corresponds to a situation where the case-space metrics and adaptation metrics coincide. For computing the probabilities defined as the output of a Bayesian CBR system, a family of rigorous algorithms provided by the theory of Bayesian network representations can be used (see e.g.[13, 11]). To be able to use these algorithms, we need to provide each arc from variable C_k to variable A_i with probabilities $P_k(a_{i1}), \ldots, P_k(a_{in_i})$. In our CBR framework, these are given as the components of the corresponding case $\mathbf{c}_k$. In addition to this, each case must be provided with a prior probability $P(C_k = 1)$. This probability can be estimated by the proportional number of occurrences of class C_k, if a database of observations is available. Similarly, the probabilities $P_k(a_{ij}) = P(A_i = a_{ij}|C_k = 1)$ can be estimated by occurrences of the value a_{ij} inside class C_k.

If the required conditional probabilities are provided, in principle we can solve the case matching and adaptation tasks using the probability measure as the metrics of our system. However, the network in Fig. 4a is not singly connected, which means that there are loops in the underlying network, if the direction of the arcs is disregarded. In this case, the problem of calculating the above mentioned probabilities can be shown to be NP-hard [1]. One approach to overcome this problem is to use stochastic simulation schemes such as Gibbs sampling [3] for approximating the outcome of the updating process. In our earlier work [7, 8, 12] we presented schemes for implementing Gibbs sampling on a connectionist network architecture. However, the problem of determining the so called annealing schedule has proven very hard in practice, resulting to slow convergence of the algorithm. On the other hand, for singly-connected networks, there exists a polynomial time algorithm for belief updating, developed by Pearl [13]. In the approach introduced in [6], 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. In the following, we show how our case base can be viewed as a simple singly connected network, a tree, in which case Pearl's belief updating algorithm can be applied directly.

Let us regard the cases $\mathbf{c}_1, \ldots, \mathbf{c}_l$ as mutually exclusive values of a single random variable, C. To be able to do this, all the cases must be *complete*, i.e., all the values $P_k(a_{ij})$ must be given for each case $\mathbf{c}_k$. 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 complete cases, we can obviously retrieve any probability $P_k(a_{ij})$, given a case $\mathbf{c}_k$. 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*[2]. What this means is that the

[2] Please note, however, that this does not mean that the attributes are independent.

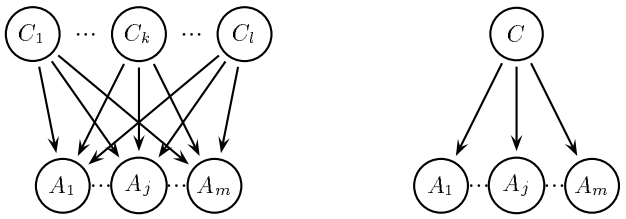

Fig. 4. Case base as a a) multiply connected Bayesian network with several case variables, and b) singly connected tree with a single case variable as the root of the tree.

Bayesian network corresponding to this representation is a tree, where a single variable C is the root of the tree, and variables A_i form the leaves (see Fig. 4b). To use this network for probabilistic reasoning, an arc to variable A_i must be provided with probabilities $P_k(a_{i1}), \ldots, P_k(a_{in_i})$, for all the cases $C_1, \ldots, C_l$. In the next section, we show how these probabilities can be stored as weights in a connectionist network, and used as part of a massively parallel probabilistic reasoning process.

4 Massively Parallel Bayesian CBR

We now show how to construct an undirected 3-layer connectionist network which performs the computations of Pearl's algorithm in parallel. In an earlier paper [9] we discussed a related directed 6-layer feedforward neural network architecture, which has a more complex structure than the connectionistic network presented here, but used simpler computational elements. In addition to the general idea presented in Section 2, we need a special intermediate layer, consisting of m groups of nodes, one for each attribute A_i. Each group has l nodes, one for each of the l cases (see Fig. 5). As the bottom layer contains one node for each of the possible attribute values a_{ij}, and the top layer one node for each of the l classes, the total number of nodes in the resulting network is $\sum_{i=1}^{m} n_i + ml + l$. The total number of arcs in the network is given by $\sum_{i=1}^{m} ln_i + lm$.

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 other layers. The weights on the arcs between the top layer and the middle layer are not needed, so they are assumed to be one. The weight $W(A_{ik})_j$ on an arc between a unit A_{ik} on the middle layer and a unit a_{ij} on the bottom layer is $P_k(a_{ij})$. Consequently, the cases of the case base are stored as weights between the bottom and the middle layer of the network.

The activation propagation starts when the user sets the initial values $S(C_k)$ and $S(a_{ij})$ for the nodes on the top and bottom layers. The initial values of the bottom layer nodes are identical to the components of the input case vector $\mathbf{c}_0$:

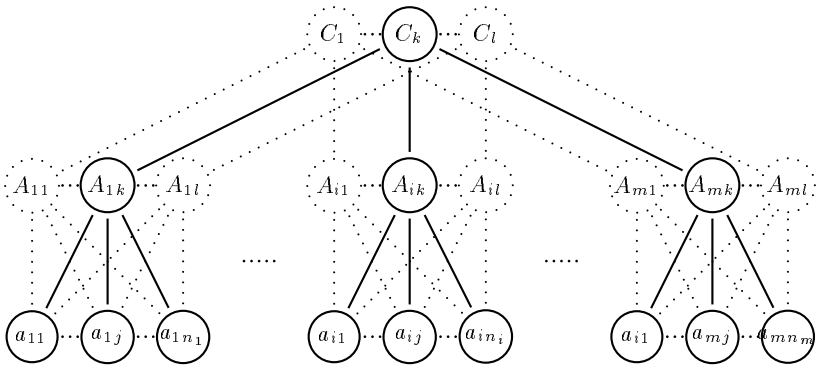

Fig. 5. A connectionist network implementing Bayesian CBR. The subnetwork corresponding to a case C_k is emphasized.

$S(a_{ij}) = P_0(a_{ij})$. The initial values of the case nodes are the apriori probabilities of the cases: $S(C_k) = P(C = C_k)$. The computation process consists of two phases. The first phase, corresponding to a bottom-up value propagation through the network, performs case matching (see Figs. 6 and 7). As the result, the third layer activation values gives us a matching score for each of the cases (i.e., to nodes C_k), thus these activation values can be directly used for classification, if needed.

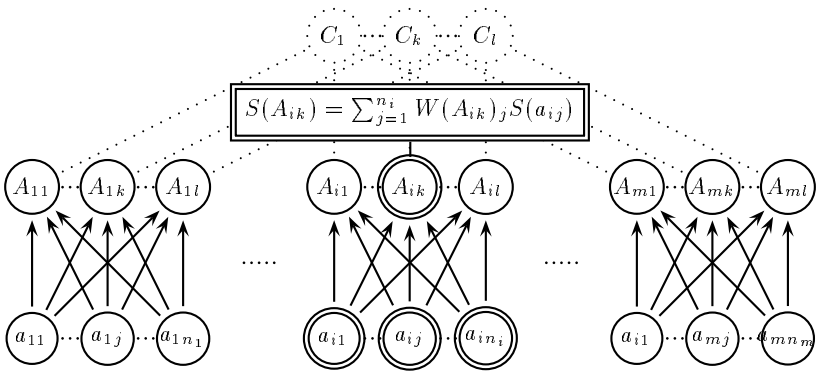

Fig. 6. Step 1 of the Bayesian case matching phase.

In the second phase of the computation process, the probabilities are propagated from the top down back to the attribute values, thus complementing the attribute value vectors based on the ranking of the cases (see Figs. 8 and 9). In general this allows all stored cases to contribute to the adaptation by the amount justified by their original matching.

Using the notation of Pearl in [13], the task of the step 2 is to compute

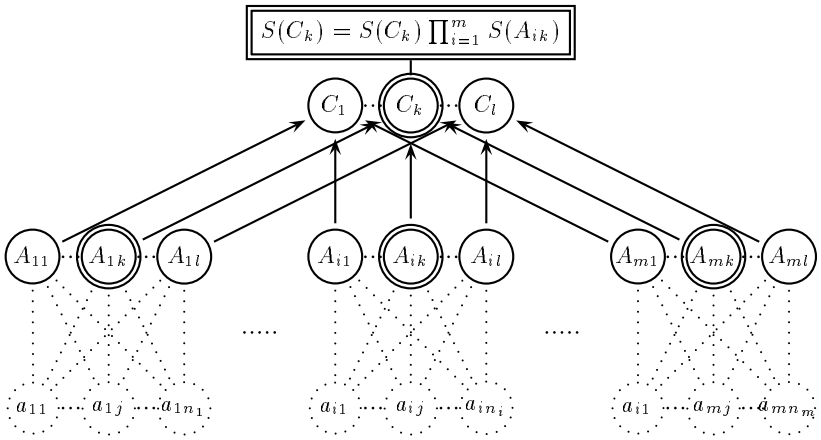

Fig. 7. Step 2 of the Bayesian case matching phase.

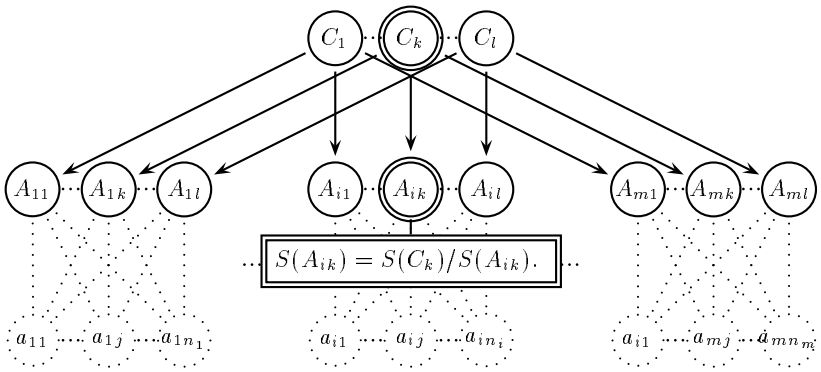

Fig. 8. Step 1 of the Bayesian case adaptation phase.

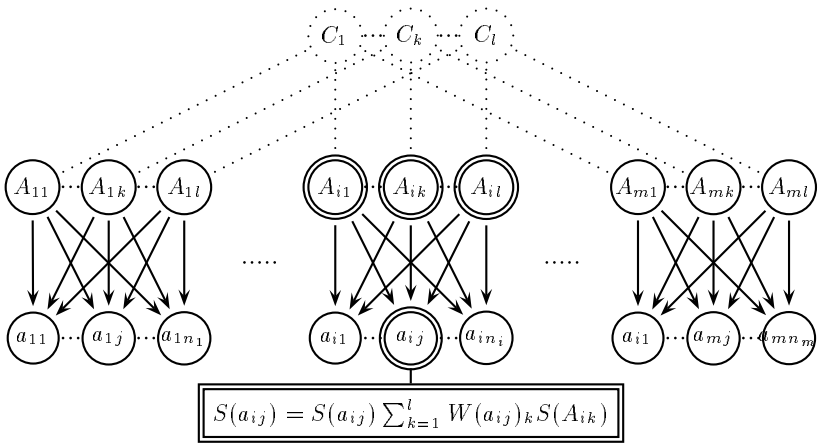

Fig. 9. Step 2 of the Bayesian case adaptation phase.

the m values $\lambda_{A_i}(\mathbf{c}_k)$, for each of the l cases. As $\lambda(\mathbf{c}_k)$ is defined as $\lambda(\mathbf{c}_k) = \prod_{i=1}^{m} \lambda_{A_i}(\mathbf{c}_k)$, the activation value of node $\mathbf{c}_k$ is $S(\mathbf{c}_k) = P(\mathbf{c}_k)\lambda(\mathbf{c}_k)$. Pearl has proved that this is equal to $\alpha P(\mathbf{c}_k \mid \mathbf{c}_0)$, where α is a normalizing constant. The actual probabilities can now be retrieved easily by normalizing the values $S(\mathbf{c}_k)$:

$$P(C = \mathbf{c}_k \mid C_0 = \mathbf{c}_0) = S(C_k) / \sum_{h=1}^{l} S(C_h).$$

In a similar way, step 4 produces the terms $\pi_{A_i}(C_k)$, and step 5 the values

$$\begin{aligned} S(a_{ij}) &= \lambda(a_{ij}) \sum_{k=1}^{l} \pi_{A_i}(C_k) P(A_i = a_{ij} \mid C = \mathbf{c}_k) \\ &= \lambda(a_{ij})\pi(a_{ij}) = \alpha P(A_i = a_{ij} \mid C_0 = \mathbf{c}_0), \end{aligned}$$

where α is again a (different) normalizing constant. As above, the actual probabilities can be retrieved by normalization:

$$P(A_i = a_{ij} \mid C_0 = \mathbf{c}_0) = S(a_{ij}) / \sum_{h=1}^{n_i} S(a_{ih}).$$

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

5 Conclusion

We have presented an approach where case-based reasoning can be implemented as a connectionist network architecture. The method is based on implementing Pearl's probability propagation as a 3-layer hierarchical network. The advantages of such an approach are twofold. In the first place, it provides an efficient solution to the case indexing problem based on the parallel architecture. Secondly, it offers a theoretically sound Bayesian interpretation of the case-space metrics and its successful application to both the case matching and adaptation via probability propagation. However, it is evident that there are several aspects left for further research. The most important of the questions to be addressed is the proper choice of the cases (observe that in the presence of noise the optimal strategy is not necessarily to store all the cases encountered). In addition, methods to determine the conditional probabilities $P(A_i = a_{ij} | C = \mathbf{c}_k)$ used by the reasoning algorithm should be investigated. Initially it can be assumed that such probabilities are estimated by the expert in a regular knowledge acquisition process, but it is clear that one can also use various statistical clustering techniques for this purpose. Initial attemps towards a theoretically solid learning algorithm for the Bayesian CBR model are reported in [10, 15].

References

1. G.F. Cooper, Probabilistic Inference using Belief Networks is NP-hard. Technical Report KSL-87-27, Stanford University, Stanford, CA, 1987.
2. DARPA, *Proceedings of the Workshop on Case-Based Reasoning 1988-1991.* Morgan Kaufmann, San Mateo, CA.
3. 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.
4. 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/MIT Press, Menlo Park, CA, 1992.
5. H. Kitano, Challenges of Massive Parallelism. Pp. 813–834 in: *Proceedings of IJCAI-93, the Thirteenth International Joint Conference on Artificial Intelligence (Chambéry, France, August 1993).* Morgan Kaufmann, San Mateo, CA, 1993.
6. 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, CA, 1990.
7. P. Myllymäki, Bayesian Reasoning by Stochastic Neural Networks. Ph. Lic. Thesis, Report C-1993-67, Department of Computer Science, University of Helsinki, 1993.
8. 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.
9. P. Myllymäki and H. Tirri, Bayesian Case-Based Reasoning with Neural Networks. Pp. 422–427 in: *Proc. of the IEEE International Conf. on Neural Networks (San Francisco, March 1993), Vol. 1.* IEEE, Piscataway, NJ, 1993.
10. P. Myllymäki and H. Tirri, Learning in neural networks with Bayesian prototypes. Pp. 60–64 in: *Proceedings of SOUTHCON'94 (Orlando, March 1994).*
11. R. Neapolitan, *Probabilistic Reasoning in Expert Systems.* Wiley Interscience, New York, NY, 1990.
12. 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.
13. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.
14. D.E. Rumelhart and J.L.McClelland (eds.),*Parallel distributed processing: explorations in the microstructures of cognition. Vol 1,2.* MIT Press, Cambridge, MA, 1986.
15. H. Tirri and P. Myllymäki, MDL learning of probabilistic neural networks for discrete problem domains. To be presented at the *IEEE World Congress on Computational Intelligence (Orlando, June 1994).*

This article was processed using the LaTeX macro package with LLNCS style