1990

Implementing Expert System Rule Conditions by Neural Networks

Henry Tirri

Report Number:
90-1050

IMPLEMENTING EXPERT SYSTEM RULE CONDITIONS BY NEURAL NETWORKS

Henry Tirri

CSD-TR-1050
December 1990

Implementing Expert System Rule Conditions by Neural Networks

Henry Tirri[*]
Purdue University
Department of Computer Sciences West Lafayette, IN47907, USA

December 3, 1990

Abstract

The relation of subsymbolic (neural computing) and symbolic computing has been a topic of intense discussion. We address some of the drawbacks of current expert system technology and study the possibility of using neural computing principles to improve their competence. In this paper we focus on the problem of using neural networks to implement expert system rule conditions. Our approach allows symbolic inference engines to make direct use of complex sensory input via so called detector predicates. We also discuss the use of self organizing Kohonen networks as a means to determine those attributes (properties) of data that reflect meaningful statistical relationships in the expert system input space. This mechanism can be used to address the difficult problem of conceptual clustering of information. The concepts introduced are illustrated by two application examples: an automatic inspection system for circuit packs and an expert system for respiratory and anesthesia monitoring. The adopted approach differs from the earlier research on the use of neural networks as expert systems, where the only method to obtain knowledge is learning from training data. In our approach the synergy of rules and detector predicates combines the advantages of both worlds: it maintains the clarity of the rule-based knowledge representation at the higher reasoning levels without sacrificing the power of noise-tolerant pattern association offered by neural computing methods.

Keywords: Expert System Architectures, Expert Systems with Complex Input, Neural Networks, Knowledge Compilation

[*] On leave from Department of Computer Science, University of Helsinki, FINLAND. This research is supported by Technology Development Center (TEKES) in Software Technology Programme(FINSOFT). Part of this work was done while the author was visiting AT&T Bell Laboratories.

1 Introduction

Currently expert systems are usually designed based on a particular knowledge representation framework (rule-based, frame-based, object-based etc.). How much of the expert system's problem-solving performance is restricted by the limitations of the representation technique chosen? It seems that many of the fundamental problems in expert system design are due to imperfect "knowledge compilation": part of the domain knowledge cannot be expressed in the format dictated by the representation framework. Expert systems perform inferences on symbolic (data structure) level, but in many applications the "knowledge" exists not only in the reasoning with concepts, but also in correctly coding input information to a symbolic form. "Is the monitored elevator traffic pattern showing rush hour behavior?" or "Is the concentration of sulphuric acid in the process within tolerance bounds?" Answering questions such as the ones above put the expert systems to a real test: how to compress complex high-dimensional sensory data into symbolic form which can be used by the inference engine, like the information received by human expert's sensory systems can be utilized at higher cognitive levels?

In spite of the underlying strong optimism in his book, Waterman [Wate 86] identifies several areas where programs have had little success of showing similar performance as human experts. Many of these problems are studied intensively in the artificial intelligence community, but we will address only two of them: direct use of complex sensory input and conceptual clustering by learning, i.e. how to "ground" symbols to the statistical information available to the application.

In our approach this "compilation" from sensory information into symbols is performed by neural networks [AnRo 88, RuMc 86]. These neural networks act as computing modules that perform noise-tolerant pattern recognition of input information. Although we adopt a purely engineering approach (as opposed to for example the competence modeling approach of [KeJo 86]), and do not try to mimic cognitive behavior of human experts, we still find an interesting starting point in studying areas where human expertise clearly is superior to the capabilities of modern expert systems.

Recent years have seen an impressive growth in neural computing research and many monographs have been published on the topic. As it is outside the scope of this paper to review the various theoretical models and their properties appearing in the literature, we suggest for an uninitiated reader to study for example the two-volume set by the PDP-group [RuMc 86] and the monograph by Kohonen [Koho 88]. Our use of neural computing principles differs considerably from the usual studies of neural networks as expert systems (see e.g. [BePe 87], [BLMW 88], [DuSh 88] and [Gall 88]) which concentrate solely on acquiring knowledge by learning processes. Our approach to expert system design is a hybrid one, the higher levels of reasoning are always described within the rule-based paradigm.

2 Managing complex sensory input information in expert systems

For pragmatic reasons we will restrict ourselves to represent symbolic knowledge in the form of rules [Wate 86]. Currently rules are the most common means to give rigorous formal expressions of knowledge about the problem domain.

Application examples

Rule-based expert systems manipulate symbols that represent ideas and concepts. However, in many application areas complex sensory data (visual data, auditory data, satellite data etc.) has to be transformed into symbols manipulated by the inference mechanism. In pure rule-based systems this translation process inevitably loses information, and depending on the application, that information may be crucial to a successful operation of the overall system. For a moment let us study some typical rules from existing expert systems (see Figure 1).

SPE is a typical example of an interpretive expert system whose purpose is to infer situation descriptions from sensory data, in this case analyze waveforms from a scanning densitometer to distinguish between different causes of inflammatory conditions in medical patients. The rule from TATR is an example of rules in planning expert systems that create plans of actions to perform a given goal. REACTOR falls into the category of monitoring systems that compare actual system (in this case nuclear reactor) behavior to expected behavior. For all of the example rules from these different types of systems there exists a need for translation from sensory data to a symbolic representation, and the correctness of this translation is essential to the viability of the rule ("TRACING PATTERN IS ASYMMETRIC GAMMA" ..., "AIRFIELD DOES NOT HAVE EXPOSED AIRCRAFT"..., ... "COOLANT SYSTEM IS INADEQUATE").

Although all the above notions "TRACING PATTERN IS ASYMMETRIC GAMMA", "DOES NOT HAVE EXPOSED AIRCRAFT" and "HEAT TRANSFER ... COOLANT SYSTEM IS INADEQUATE" have a precise meaning, problems arise when one attempts to give a formal definition, e.g. in logic, to describe the condition. The situation is further complicated by the fact that the sensory data is usually noisy, sparse or incomplete, in worst case all of them. In the case of one-dimensional data it is sometimes possible to approximate these conditions by giving symbolic boundary expressions such as toleration intervals

$$0.6 < \text{HEAT TRANSFER} < 0.9,$$

but this approach does not generalize well to higher dimensional data with more complicated relations, e.g. recognizing similarities of waveforms. In the literature there exist many studies on fuzzy logic [Zade 83] for approximate reasoning strategies which aim at a good estimate for uncertain data and imperfect rules.

Instead of describing the certainty factor of the truth value of a rule R, our approach is much more pragmatic, we simply implement a reliable detector for the predicate P in the condition part of the rule. We have adopted this approach as we have observed that very seldom the rules themselves are fuzzy, but in many cases the detectors for the condition predicates are hard to describe algorithmically. Thus the problem is how to preserve the relevant information when changing information into symbolic form rather than the impreciness of the the rules themselves.

3 Implementing data compilation with neural networks

Formally, a neural network N is a dynamical system which has a topology of a directed graph and carries out information processing by means of its state response to (continuous) input [Hech 87a]. One class of neural networks suitable for compiling sensory data into symbols are networks that directly approximate the target function

$$g : S \subseteq R^n \longmapsto S' \subseteq R^m$$

after self-adjustment in response to a finite descriptive set of example mapping pairs $\{(i_1, o_1), \ldots, (i_k, o_k)\}$ (where $o_j = f(i_j)+\eta$, η is a stationary noise process). Such layered networks are discussed for example in [Hech 87b] and [Werb 74].

The nodes in the networks are simple computational elements: a node sums k weighted inputs and passes the result through a nonlinearity f as shown in Figure 2. The topology of these networks is a DAG with usually less than three layers.

Neural networks as classifiers

Much of the interest in these networks is due to the observation that they can be used as pattern classifiers (see e.g. [DuHa 73]) in the d-dimensional feature space defined by the network inputs [Werb 74], [RuMc 86], [Lipp 87]. The decision boundary in the feature space is defined by the state of the network (connection topology, weight matrix W, function f and threshold value vector Θ). Although in principle the position of this decision boundary in the feature space could be "programmed" directly by setting arc weight values, the values of the weights are seldom known a priori. Hence the correct positioning of the boundary is approximated by a training process, where a set of examples of input instances $\{I_1, \ldots, I_k\}$ and a correct classifications $\{O_1, \ldots, O_k\}$ are presented to the network, and an algorithm called *learning rule* [RuMc 86] is used to calculate weight changes depending on network's performance with the current weights. The network input includes also the correct classifications, hence this type of training is called supervised learning [DuHa 73]. Finally, the complexity of the shape of the boundary (linear, convex etc.) that can be realized is dependent on the number of layers (see [Lipp 87]), however for our purposes it is enough to know that any realizable shape can be produced by a three-layer network of the above elements — at least in principle.

For the above layered networks many learning algorithms have been suggested [Lipp 87], [Werb 74], [RuMc 86], e.g. backpropagation and its variants [Werb 88]. In our case the details of the learning algorithms are not important, it is sufficient to know that for example backpropagation is an iterative gradient descent algorithm designed to minimize the mean square error between the actual and desired output of the multilayer network.

If neural network N solves a two-class classification problem, such as the question if an input **I** is regular/nonregular, in our knowledge representation formalism the network corresponds to a single data predicate P_l. However, in many cases a single network is capable of solving m-class problems and hence implements a set of (usually mutually excluding) predicates $P_l = \{P_{1l}, \ldots, P_{ml}\}$.

4 Knowledge representation with rules and detector predicates

In our case the knowledge base consists of three (sub)knowledge bases: a rule base R_b, a fact base F_b and a neural network base N_b. Following the common expert system terminology a rule is understood as a condition-action -statement

rule i: IF C_i THEN A_i;

with the obvious semantics (if the condition part C_i is evaluated to be true, the action part A_i will be performed). A condition C_i is an expression containing one or more predicates P_{ik}. For our purposes it is sufficent to make a distinction between *detector* and *non-detector* predicates. The former predicates (denoted by P_d) are implemented by neural networks in N_b.

An action A_i is a set of operations $\{o_{ij}\}$, each operation being either

- internal, i.e. it modifies the fact base F_b and/or internal variables,
- external, i.e. call to an external procedure (e.g. for an alarm signal) or
- adaptive, i.e. call to a neural network in N_b in training mode.

Internal operations allow storing deduced knowledge for further use and external operations provide the interface to the environment where the expert system is functioning. These two types of operations are usually found in all monitoring, planning or interpretive expert systems. The adaptive operations, which are related to the adaptiveness of the data detectors, will be discussed in more detail in the context of dynamic behavior of this expert system architecture.

Fact base

The fact base F_b consists of facts stating that a particular predicate P holds for certain objects s_i in the object domain S of the expert system. The predicates appearing in the fact base F_b can also be detector predicates. In this case the detector predicate on current input **I** has already been evaluated. Hence a fact base acts as the "memory" of the expert system. In addition to the static part that states the universal facts about the problem domain, it also contains dynamically changing knowledge about a particular execution that can be erased later on.

Neural network base

The neural network base N_b contains a set of neural networks $\{N_i\}$, each of which corresponds to one or more detector predicates. Like rules in the rule base, neural networks are active components of the system. They perform a classification operation that implements a predicate test on their input data and keep the latest data stored in an associated buffer. We call this process knowledge compilation, as the statistical relationships within the input data are translated into symbolic facts. In this sense N_b is analogous to the rule base R_b; the latter contains the knowledge for inference at the symbolic level, the former the knowledge for reasoning at the subsymbolic, stochastic data level.

Architecture of the system

The dynamic behavior of the expert system is illustrated in Figure 3. The inference engine performs the normal backward/forward-chaining of rules. However, the inference engine may encounter a rule which has detector predicates $P_{d1}(l,x), P_{d2}(j,y), \ldots$ in its condition part, and it cannot decide the value of a predicate from the facts in fact base F_b, it performs a call for the corresponding neural network(s) with predicate arguments as parameters. These parameters are not the actual input values to the neural network N, they define what classification result (l) is significant to the condition (since a multi-class classifier corresponds to a set of predicates $\{P_{di}\}$) and the address of the input device (x). The actual input values the neural network receives from the sensory data equipment directly, the arrival of the parameters acts only as a trigger to the classification process. In in its simplest form N returns a boolean predicate value. If the predicate is true, the action part A of the rule in question has an operation o_i that stores the corresponding fact in the fact base. If it is evaluated false, the system automatically stores the negative fact. This is necessary in order to prevent the subsequent encounters with the predicate in some other condition parts from invoking the knowledge translation process again, i.e. the system "memorizes" the fact. Naturally this stored information is query-dependent and is removed after the query is completed. In the case of graded classifier output the value itself is stored in the fact base, i.e. a graded predicate always "succeeds".

Adaptation

Layered neural networks, such as the ones we are using, often require an extensive training period with a sample set adjusting the weights to reflect a good approximation of the decision boundary in the feature space. Obviously the larger and statistically more representative this training sample set is of the whole input space S, the better the performance (i.e. the accuracy of the approximation) is. Unfortunately in most cases it is not possible to gather enough data samples in advance to create a truly representative set. Therefore the system described must be prepared to continue adjusting the network weight matrix while already functioning by using the adaptive operations oa in the action parts. If during the inference process there is substantial evidence of the fact $P_d(l,x)$ in F_b being incorrect, deduced either automatically or by human intervention an adaptive operation is performed. The adaptive operation is implemented as a call to the corresponding N with the correct classification and the request to train to perform this classification. Observe that we required our N modules to be able to store the latest input data whose classification was triggered, hence the symbolic reasoning module does not have to deal with the actual data at all. This adaptation mechanism gives a way to gradually improve the accuracy of the detector predicate implementations with real input data.

5 Self-organization for attribute selection

In practice one of the most difficult issues in the design of an expert system is the question of attribute selection for knowledge representation. As any real world process has infinitely many attributes, the problem is how to choose such a small attribute set for the knowledge base that it would be descriptive enough for the modeling purposes. This problem is present especially when machine learning methods (either neural or symbolic, decision tree based ones such as ID3 [Quin 79]) are used for knowledge acquisition.

Interestingly enough, one important organizing principle of sensory pathways in the brain is that the placement of neurons is orderly and often reflects some characteristic of the external stimulus being sensed [KaSc 85]. Inspired by this biological fact some of the neural network models and their associated learning algorithms promote self organization [Koho 88], [Gross 88]. As variants of these Kohonen networks can also be used directly as classifiers, they especially suitable for our purposes.

Self-organizing Kohonen networks

In Kohonen networks the learning algorithm generates a mapping of a higher dimensional input space S onto (usually two-dimensional) discrete lattice M of output nodes. The map is generated by establishing a correspondence between the inputs in S and output nodes in M, such that the topological neighborhood relationships among the input instances are reflected as closely as possible in the arrangement of the corresponding nodes in the lattice. As a result of this process, a non-linearly reduced two-dimensional version of the input space is found. This data structure can be used to cluster input attributes.

The correspondence is obtained as follows. Each input instance is represented by a vector $s \in S$. For each training cycle an input instance $s \in S$ is chosen randomly according to a probability distribution $Pr(s)$. Each location $m \in M$ has an associated vector $w_m \in S$. These vectors w_m map lattice locations m to points in S. For each training cycle the mapping is modified according to the following abstract algorithm:

A1. Determine lattice location c for which

$$||w_c - s|| = min_{m \in M} ||w_m - s||$$

where s is the input chosen for the training cycle.

A2. For all nodes m in the neighborhood of c modify

$$w_m(t+1) = w_m(t) + \alpha \delta_{mc}(s - w_m(t)).$$

Here $0 \leq \delta_{mc} \leq 1$ is the adjustment function for the distance $||w - s||$ and α is the learning step size.

By decreasing the step size α and the width of δ_{mc} slowly during training, the algorithm gradually yields values for the vectors w_m which define a discretized neighborhood conserving mapping between lattice nodes m and points of the input space S [Koho 88].

Self-organization for attribute selection

We now turn to the problem of using this self organization process for attribute selection. Let us assume that our input space S is d-dimensional, i.e. each input instance is a vector $\mathbf{v} = (v_1, v_2, \ldots, v_d)$. Let T be the training set, i.e. a set of such vectors. Further assume that the (output) nodes M are arranged as a grid (size k^2). In the training process an input instance enforces the sensitivity of the most responding node c (closest in d-space) and the nodes in its immediate neighborhood (defined by δ_{mc}), hence the resulting network has a tendency to form clusters of nodes that are sensitive to similar inputs (see Figure 4). After the completion of the training process, each cluster C_i is labeled with a meaningful attribute name B_i (semiautomatically) by finding an example set of vectors

T_i from the training set such that the nodes in C_i are sensitive to these input instances. These example vector sets T_i help giving a meaningful interpretation for the clusters C_i. Observe that this process resembles multivariate methods such as factor analysis, but is nonlinear in nature.

As the output of the cluster nodes c_i is graded, these attributes could directly be used in relational expressions to form detector predicates P_{di} discussed above. For example, "sensor object s is a rock" if detector predicate *is-rock(s)* holds, where *is-rock(i)* $\equiv$ ROCK $> .7$ (ROCK is an attribute defined by the clustering process). However, better results are obtained if this self-organization process is used as a pre-processing step for a more sophisticated classifier. The clustering and labeling process gives the number and type of the classes after which a neural implementation of a nearest-neighbor method called LVQ [Koho 88] can be used to tune the classifier with the same training set T. This tuned classifier gives an alternate neural implementation to a set of detector predicates to those based on layered networks in Section 3, and is more viable in the cases where the structure of the expert system input space is not well-understood in advance.

6 Example: expert system for automatic inspection

We demonstrate the ideas discussed by applying them to the design of an expert system for automatic inspection of circuit packs. We will focus on showing the benefits of our overall approach in this particular application, readers interested in the details of the problem and a comparison of the different solution methods adopted should consult [MoRT 89a],[Tirr 89].

Computer vision is playing an ever increasing role in assuring the quality of manufacturing processes by making available low cost, reliable inspection. A computer vision system placed in-line after the placement operation can catch errors before the soldering process, thereby also reducing the repair cost. One special problem that arises in electronic assembly is the component orientation error. Although in sometimes even hundreds of components are placed on a single circuit pack, the components must be loaded into their hoppers manually and the symmetry of the component allows an orientation error to occur. Presently there is no standardization of the orientation marks (notches, dots etc.) and even if marks are used, they are often very hard to detect with a computer vision system. Hence the only starting point for the orientation detection is the information printed on the electronic component.

Therefore let us focus on one particular rule in the inspection expert system, namely

IF ORIENT(CHIP$_i$) = ORIENT(CHIP$_i$) IN DESIGN-DB
THEN CHECK-PINS(CHIP$_i$);

that requires checking the component orientation before initiating the inspection of component pins. In principle one could use backward-chaining to solve the value of ORIENT(CHIP$_i$) and then compare it to the value accessed from the design database. One could imagine a set of rules that could be used to detect the orientation based on the features extracted from the image produced by the machine vision system. However, the viability of this solution is questionable if we consider more closely some of the key requirements for this text orientation problem:

- There is no advance knowledge of font style or size.
- As opposed to Optical Character Recognition there is no opportunity to use contextual information (dictionaries) to resolve difficult-to-detect characters.
- The printing is often poor quality, e.g. characters are touching, misaligned and may contain nonchacter symbols.
- Many characters are invariant or almost invariant to a 180 degree rotation (or when rotated resemble some other character without rotation). Hence the system must be prepared to output also an "indeterminate" response.
- Detection must be carried out very quickly (typically up to 100 characters/second).

A solution to this problem is to implement a detector predicate ORIENT() as a feedforward network presented in Figure 5, which can then be called when executing the corresponding rule.

The feed-forward neural network model used in this application is called Random Neural Networks (RNN) [Tirr 90]. RNN networks resemble structurally the ones used with gradient descent based learning methods such as back-propagation [Werb 74, RuMc 86, Werb 88]. However, it should be pointed out that with the RNN neural networks there is no iterative learning process. Computation in these RNN networks are based on a set of reference vectors $a_1, a_2, \ldots, a_s$ and $b_1, b_2, \ldots, b_s$ (symbols in normal orientation and rotated, respectively), and the weights are set only once when the reference vector set is stored. Adaptation is achieved by incrementally adding new nodes if new reference vectors become available. One should also notice that the effect of adding new reference vectors to the computed function is only gradual. Thus the undesirable interference properties frequently observed with many of the neural network learning algorithms [BrSm 89, McCl 88] can be avoided.

In RNN, the first active layer, consisting of "pattern units", computes Hamming distances $h(x, a_i)$ and $h(x, b_i)$; the next layer, consisting of "summation units", forms $p(x \mid a)$ and $p(x \mid b)$ as weighted sums, and finally the "output units" give the final decision by thresholding. As in this case computation of the conditional probabilities is based on calculating Hamming distances, the

RNN neural network model can also be understood as a generalization of the "Hamming Network" [Stei 61, StPi 63, Tayl 64]. As pointed out by Lippman [Lipp 87], it is well known that under the assumption of independent bit errors, the optimal minimum error classifier calculates the Hamming distance to a single reference instance (or codeword) in each concept and selects the vector with the minimum distance as the decision. A schematic of a network representing this mechanism is presented in [LiGo 87]. The RNN approach can be viewed as a natural generalization of that idea in that rather than taking the nearest reference instance, we compute the likelihood ratio from families of reference instances for a concept. However, it should be noted that the approach presented here is not restricted to Hamming metric; it can be generalized to any problem domain for which a metric can be defined on the input space. A more detailed discussion on the theoretical basis of RNN networks is outside the scope of this paper, and presented in [Tirr 90].

Figure 6 presents an example of the recognition process using RNN in a case of a rotated chip. For the reference vectors in this application bitmaps of the letters in Sun font library were used. In Figure 6 we can see the automatically extracted characters together with the corresponding likelihood ratios. The closer the likelihood ratio is to 0, the more certain the detector predicate is that the symbol is up-side down. More detailed discussion on implementation and the performance of the application can be found in [MoRT 89a, Tirr 90].

7 Example: respiratory and anesthesia monitoring

As a second example we will briefly compare the approach presented above to implementing an expert system with traditional rule-based techniques. Rather than inventing an artificial example of our own, we chose an example from the literature in the area of intelligent monitoring in medical environment. The original prototype system CAPS [RaCM 87], has been implemented by using various commercially available software tools for statistical analysis and expert system development.

One type of equipment malfunction during anesthesia is the accidental disconnection of the paralyzed patient from the life sustaining ventilator. If this occurs, no oxygen can be delivered to the patient and the carbon dioxide produced by the patient is not removed from the lungs. The result is a brain damage in less than five minutes. To prevent this from happening analyzers that produce carbon dioxide waveforms (capnograms) are used for monitoring purposes. A mass spectrometer produces a capnogram which is plotted against time and displayed on a CRT monitor together with some numerical information gathered. Unfortunately the capnogram and its relationship to physiological changes in the patient and to anesthesia equipment malfunction is not usually well understood by many anesthesiologists. Lack of experience with capnograms reduces the information gained and may cause misinterpretations. Therefore the work in [RaCM 87] aims at a real-time expert system for analyzing capnograms which could indicate whether the waveforms are normal or abnormal and suggest actions to fix the possible problem.

Before the design of CAPS rule base the authors used manually a statistical analysis package to analyze data stored in a capnogram database (each capnogram has also associated descriptive information about the case). Based on this analysis they chose 12 different capnogram types to be recognized with an associated feature set. In our approach this database would be used to train the self organizing network to find the initial clustering, i.e. the number of capnogram classes. In this case the input space would be d-dimensional, where d is the resolution of the waveform image stored in the database. This gives us a statistically rigorous method to determine different attributes that can be used to form detector predicates such as REBREATHING(CGRAM), NORMAL(CGRAM), CARDIOGENIC-OSCILLATION(CGRAM) etc. As discussed already in the previous sections, the LVQ-method could then be used to tune the classifier neural network that implements the set of detector predicates which describe the nature of the waveform.

In the original prototype system CAPS most of the 48 rules were used for deducing the classification from elementary features, i.e. they served pattern recognition purposes. Based on the experience with CAPS the authors predict that satisfactory operation in a delivered product would be achieved with 1000 rules! In our case the detector predicates perform the pattern recognition functions and free the rule base to describe only the necessary actions (less than 20 rules) and other higher level relationships based on this information. This considerable reduction in the number of rules has obvious performance benefits. As the real product system in the operating room should analyze several gases simultaneously (nitrogen, oxygen, isoflurane etc.) our rules could also easily express complex relations between different classifications such as

IF HYPERVENTILATION(CGRAM) AND NORMAL (NGRAM)
THEN CHECK(ABNORMAL(OGRAM))

and hence improve the expert systems ability to gain better evidence about the cause of a possible abnormality. Our claim is that the rule bases produced this way are easier to maintain and understand, since the low level reasoning for the pattern analysis is not mixed with the high level reasoning considering proper actions.

8 Conclusions

Neural networks can be used to implement the compilation from stochastic sensory data to symbols manipulated by the inference engines. The most obvious application areas are expert systems that make direct use of complex sensory input, however, the ideas presented are in fact much more general. We already mentioned the notion of self-organization as a means to determine those properties of data that reflect meaningful statistical relationships in the expert system input space. The mechanisms described can be useful in many related areas such as robotics (autonomous vehicles etc.), and image processing. As our examples show, many isolated components of the systems already exist (LVQ- network simulators, design tools for rule bases etc.) but currently the integration itself requires considerable additional effort.

References

[AnRo 88] Anderson,J.A. and E.Rosenfeld (Eds.), Neurocomputing. Foundations of research. The MIT Press, 1988.

[BePe 87] Becker,L.E. and J.Peng, Using activation networks for analogical ordering of consideration: one method for integrating connectionistic and symbolic processing. Proceedings of the IEEE International Conference on Neural Networks, San Diego, pp.367–371, 1987.

[BLMW 88] Bounds,D.G, P.J.Lloyd, B.Matthew and G.Waddell, A multi layer perceptron network for the diagnosis of low back pain. Proceedings of the IEEE International Conference on Neural Networks, San Diego, pp.481–489,1988.

[BrSm 89] Brousse,O. and P.Smolensky, Virtual memories and massive generalization in connectionist combinatorial learning. Report CS-431-89, Department of Computer Science, University of Colorado at Boulder, 1989.

[CaWK 84] Callero,M., D.A.Waterman and J.Kipps, TATR: a prototype expert system for tactical air targeting. Rand Report R-3096-ARPA, The Rand Corporation, Santa Monica, CA, 1984.

[DuHa 73] Duda,R.O. and P.E.Hart, Pattern classification and scene analysis, John Wiley & Sons, 1973.

[DuSh 88] Dutta,S. and S.Shekhar, Bond rating: a non-conservative application of neural networks. Proceedings of the IEEE International Conference on Neural Networks, San Diego, pp.443–450. 1988.

[Gall 88] Gallant,S.I., Connectionistic expert systems. Communications of the ACM (31), pp. 152–169, 1988.

[Gross 88] Grossberg,S.(Ed.), Neural networks and natural intelligence. The MIT Press, 1988.

[Hech 87a] Hecht-Nielsen,R., Neurocomputer applications. In R. Eckmiller and C.v.d.Malsburg (Eds.), Neural computers, Springer-Verlag, pp. 445 451, 1987.

[Hech 87b] Hecht-Nielsen,R., Counterpropagation networks. To appear in Applied Optics, 1987.

[KaSc 85] Kandel,E.R. and J.H.Schwartz, Principles of neural science, Elsevier, 1985.

[KeJo 86] Keravnou,E.T. and L.Johnson, Competent expert systems - a case study in fault diagnosis. McGraw-Hill, 1986.

[Koho 88] Kohonen,T., Self-organization and associative memory. 2nd Edition, Springer-Verlag, 1988.

[Lipp 87] Lippmann,R.P., An introduction to computing with neural nets. IEEE ASSP Magazine, pp. 4-22, 1987.

[LiGo 87] Lippmann,R.P., B.Gold and M.L.Malpass, A comparison of Hamming and Hopfield neural nets for pattern classification. Technical Report 769, MIT Lincoln Laboratory, MIT, 1987. Press, 1969, 463-502.

[McCl 88] McCloskey,M., and N.Cohen, Catastrophic interference in connectionist networks: The sequential learning problem. To appear in G.Bower (ed.): The psychology of learning and motivation: Volume 23, 1988.

[MoRT 89a] Morris,R.J.T., L.Rubin and H.Tirri, Neural network techniques for object orientation detection: solution by optimal feedforward network and learning vector quantization approaches. IEEE Trans. on Pattern Analysis and Machine Intelligence (12), pp. 1107-1115,1990.

[MoRT 89b] Morris,R.J.T., L.Rubin and H.Tirri, A comparison of feedforward and self- organizing approaches to the font orientation problem. Proceedings of the International Joint Conference on Neural Networks (IJCNN'89), Washington D.C., pp. 291–298, 1989.

[Nels 82] Nelson,W.R., REACTOR: an expert system for diagnosis and treatment of nuclear reactor accidents. Proceedings of AAAI, 1982.

[Quin 79] Quinlan,R., Discovering rules from large collections of examples. A case study. In D.Mitchie (ed.), Expert systems in the microelectronic age. Edinburgh University Press, 1979.

[RaCM 87] Rader,C.D.,V.M.Crowe and B.G.Marcot, CAPS: a pattern recognition expert system prototype for respiratory and anesthesia monitoring. Proceedings of Western Conference on Expert Systems, Anaheim, CA, pp.162–168, 1987.

[RuMc 86] Rumelhart,D.E. and J.L.McClelland (Eds.), Parallel distributed processing: explorations in the microstructures of cognition. Vol 1,2. The MIT Press, 1986.

[Stei 61] Steinbuch, K., Die Lernmatrix. Kybernetik 1, 36–45, 1961.

[StPi 63] Steinbuch, K., and U.A.Piske, Learning matrices and their applications. IEEE Trans. on Electronic Computers, 846–862, 1963.

[Tayl 64] Taylor, W., Cortico-thalamic organization and memory. Proc. Royal Society of London B 159, 466-478, 1964.

[Tirr 89] Tirri,H., Applying Neural Computing to Expert System Design: Coping with Complex Sensory Data and Attribute Selection. Proceedings of the Third International Conference on Foundations of Data Organization and Algorithms (FODO'89), Paris, pp. 474–489, 1989.

[Tirr 90] Tirri,H., Sparse and continuous random concepts in artificial intelligence. Submitted for publication, 1990.

[WaSh 82] Wallis,J.W. and E.H.Shortliffe, Explanatory power for medical expert systems: studies in the representation of causal relationships for clinical consultations. Meth. Inform. Med (21), pp.127–136, 1982.

[Wate 86] Waterman,D.A., A guide to expert systems. Addison-Wesley, 1986.

[WeKu 84] Weiss,S.M. and C.A.Kulikowski, A practical guide to designing expert systems. Rowman&Allanheld (NJ,USA), 1984.

[Werb 74] Werbos,P., Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard U. Committee on applied Mathematics, 1974.

[Werb 88] Werbos,P., Backpropagation: past and future. Proceedings of the IEEE International Conference on Neural Networks, San Diego, pp. 343–353, 1988.

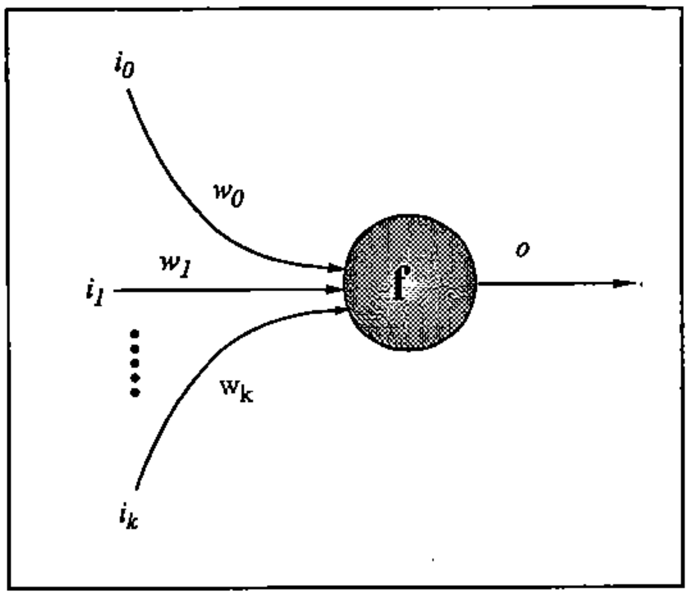
[Zade 83] Zadeh,L.A., The role of fuzzy logic in the management of uncertainty in expert systems. Fuzzy Sets and Systems (11), pp. 199–227, 1983.

[SPE [WeKu 84]] **IF** THE TRACING PATTERN IS ASYMMETRIC GAMMA AND THE GAMMA QUANTITY IS NORMAL
THEN THE CONCENTRATION OF GAMMAGLOBULIN IS IN NORMAL RANGE

[TATR [CaWK 84]] **IF** THE AIRFIELD DOES NOT HAVE AN EXPOSED AIRCRAFT AND THE NUMBER OF AIRCRAFT IN THE OPEN AT THE AIRFIELD IS GREATER THAN 0.25 TIMES THE TOTAL NUMBER OF AIRCRAFT AT THAT AIRFIELD
THEN LET EXCELLENT BE THE RATING FOR AIRCRAFT AT THAT AIRFIELD

[REACTOR [Nels 82]] **IF** THE HEAT TRANSFER FROM THE PRIMARY COOLANT SYSTEM TO THE SECONDARY COOLANT SYSTEM IS INADEQUATE AND THE FEEDWATER FLOW IS LOW
THEN THE ACCIDENT IS LOSS OF FEEDWATER.

Figure 1: Some typical rules from existing expert systems that illustrate the need for translation from sensory data to symbolic form.



$$o = f(\sum_{j=1}^{k} w_j i_j - \Theta)$$

where f is a nonlinearity.

Figure 2: The computational element in a layered neural network.

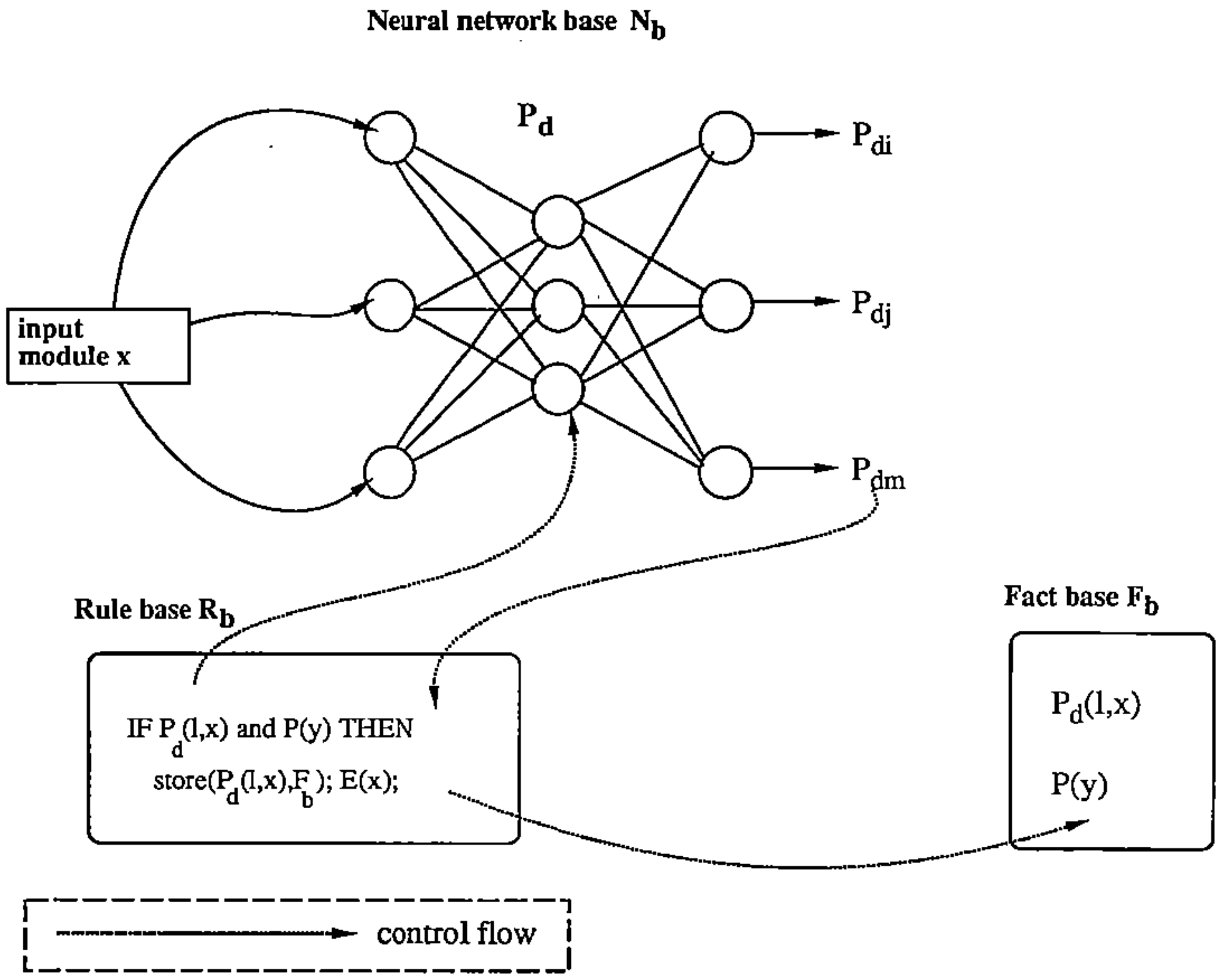


Figure 3: The behavior of a rule with a boolean detector predicate.

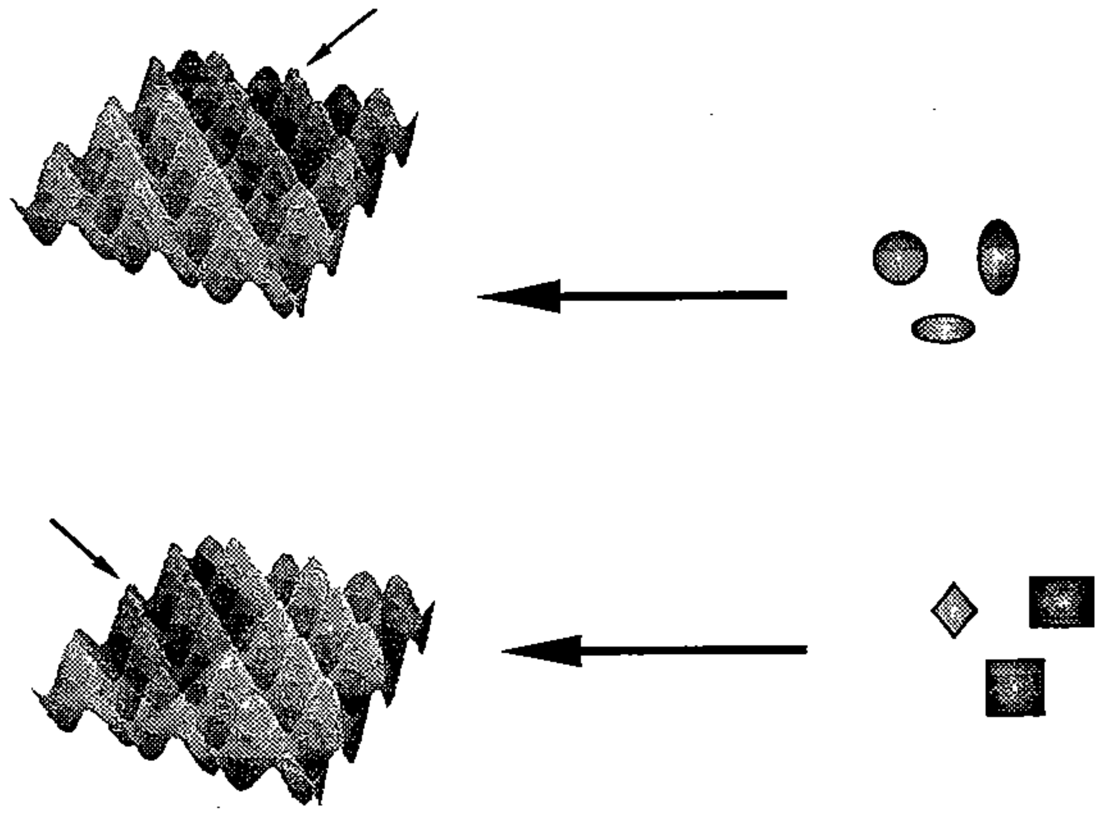


Figure 4: The self organization algorithm forms clusters among the output nodes. Each of the peaks on the surface represent a cluster center, i.e. the peak element and its neighbors are sensitive to a particular attribute value.

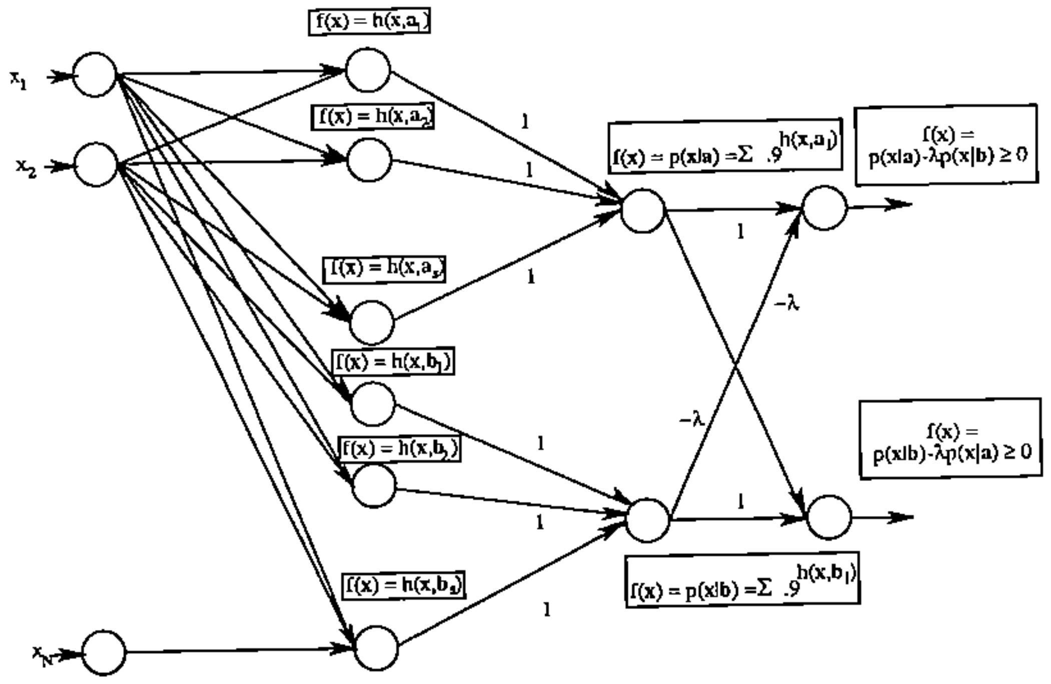
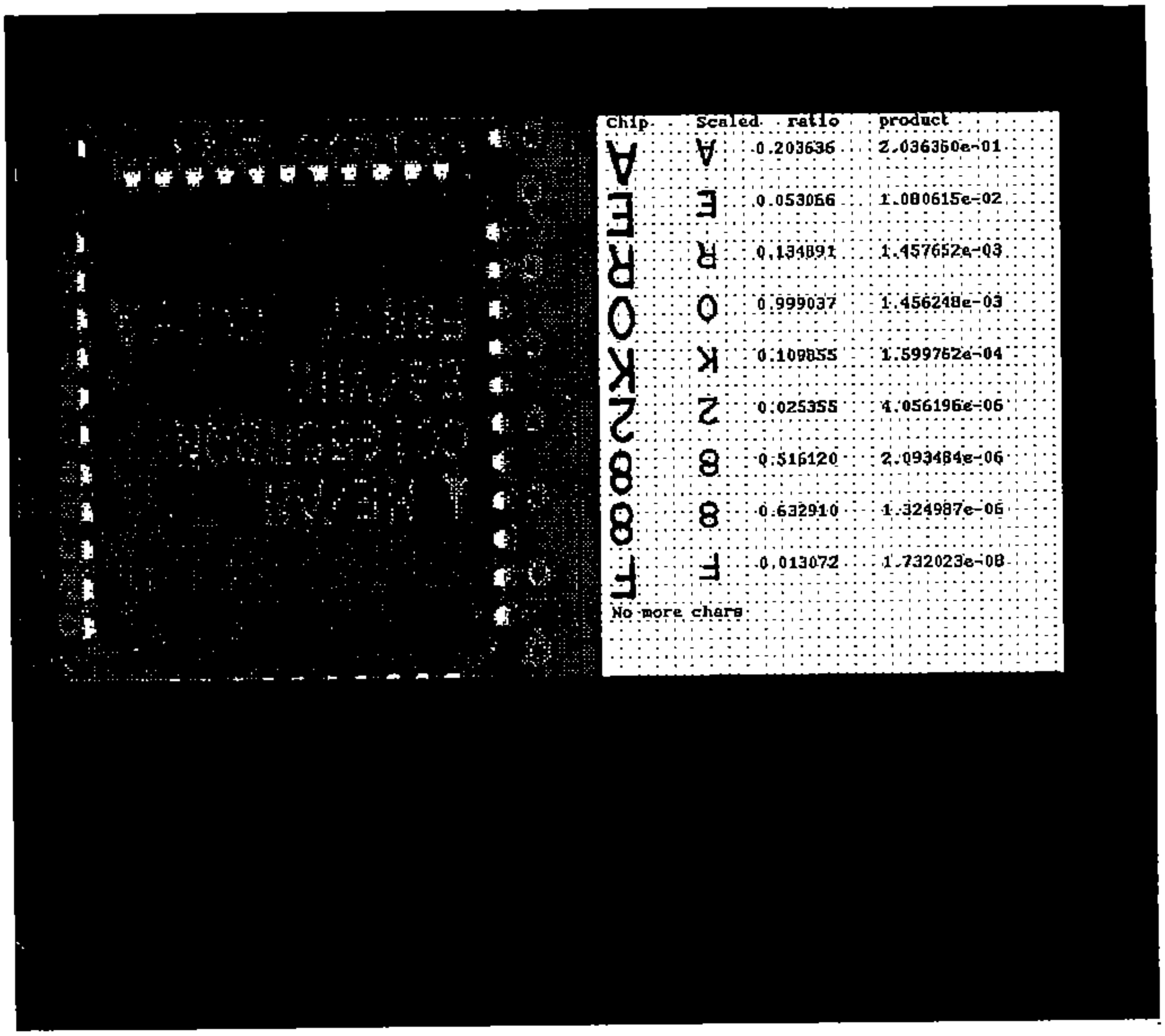

x_1
x_2
x_N
f(x) = h(x,a₁)
f(x) = h(x,a₂)
f(x) = h(x,a_s)
f(x) = h(x,b₁)
f(x) = h(x,b₂)
f(x) = h(x,b_s)
1
f(x) = p(x|a) = Σ .9^h(x,a_i)
f(x) = p(x|b) = Σ .9^h(x,b_i)
−λ
f(x) = p(x|a)-λp(x|b) ≥ 0
f(x) = p(x|b)-λp(x|a) ≥ 0

Figure 5: The RNN neural network implementing the detector predicate ORIENT().



```
Chip    Scaled  ratio      product
                0.203636   2.036360e-01
                0.053066   1.080615e-02
                0.134891   1.457652e-03
                0.999037   1.456248e-03
                0.109855   1.599762e-04
                0.025355   4.056196e-06
                0.516120   2.093484e-06
                0.632910   1.324987e-06
                0.013072   1.732023e-08
No more chars
```

Figure 6: An example of the recognition process of a chip on a circuit board. Low cumulative ratio on the right indicates that the chip is upside-down. The software is currently running on a Sun 3/260 workstation.