Knowledge Matrix ---- An Explanation & Knowledge Refinement Facility for a Rule Induced Neural Network

Daniel S. Yeung
Department of Computing,
Hong Kong Polytechnic, Hong Kong.
Fax : (852) 7642528
Email : csdaniel@hkpcc.hkp.hk

Hak-shun, Fong
Department of Computing,
Hong Kong Polytechnic, Hong Kong.
Fax : (852) 7642528
Email : cshsfong@comp.hkp.hk

Abstract

One of the major shortcomings of neural network as a problem solving tool lies in its opaque nature of knowledge representation and manipulation. For instance, the way that a learning algorithm modifies the connection weights of a network cannot be easily understood in the context of the application domain knowledge. Thus, the applications of neural networks is limited in areas where user's understanding of the situation is critical. This paper introduces a facility called knowledge matrix for a rule induced Neocognitron network. It represents the correlation between the knowledge stored internally in the network and the symbolic knowledge used in the application domain. Another facility called response matrix is developed to represent the network's response to an input. These two facilities are then employed cooperatively to generate symbolic interpretations of the network's response. Based on the interpretations, queries can be made against the network's responses and explanations can be provided by the system. Two detailed examples are discussed. It can be shown that the network knowledge can be refined evolutionarily without degrading its comprehensibility. An algorithm has also been formulated to adapt the system with respect to one type of recognition error.

Introduction

Although neural networks possess learning and generalizing capabilities, much of their internal knowledge representation and manipulation is incomprehensible. It is also difficult to make use of the ways they learn to refine the problem domain knowledge. These two shortcomings seriously hinder the use of neural networks in situations where a high degree of human interaction is required. To overcome these difficulties, one possible means is to build a neural network system with embedded high-level, symbolic domain knowledge.

Several hybrid systems which attempt to integrate neural networks and symbolic knowledge manipulations have been proposed (Fu & Fu 1990, Gallant 1988, Hayashi, Krishnamraju & Reilly 1991, Towell, Shavlik & Noordewier 1990, Towell & Shavlik 1992). Although some of them (Fu & Fu 1990, Towell, Shavlik & Noordewier 1990, Towell & Shavlik 1992) incorporate domain knowledge into the neural networks, they employ conventional learning algorithms to train the networks. Therefore, it is still difficult to understand why certain changes on the network have taken place. On the other hand, Hayashi (Hayashi, Krishnamraju & Reilly 1991) has proposed a hybrid architecture in which a "cooperative module" is employed to exchange knowledge between a neural network and an expert system. Unfortunately, very little information on this module is given. In this paper, a rule induced neural network for handwritten Chinese character recognition is proposed using a priori symbolic knowledge, i.e., a set of production rules. The construction of such a rule induced network is described in Section 2. A facility called knowledge matrix which provides symbolic interpretations of the network's response to inputs is presented in Section 3. Thus, it follows that the network knowledge can be refined based on information stored in the knowledge matrix. Section 4 gives the conclusion and possible future work.

Rule Induced Neural Network

The neural network being considered is a rule induced network (Yeung, Fong & Cheung 1992), which is so devised to represent the syntactic structure of a small set of seventeen Chinese characters. A syntax of attribute rules is formulated to describe the structural knowledge of these characters, and a mapping scheme is also established to program the network using these rules as a template. The rule induced network thus constructed is found to

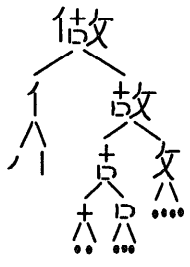

Figure 1 : Structure Decomposition Tree

Displacement from the expected location

	-2	-1	0	+1	+2
-2	.25	.25	.25	.25	.25
-1	.25	.75	.75	.75	.25
0	.25	.75	1.0	.75	.25
+1	.25	.75	.75	.75	.25
+2	.25	.25	.25	.25	.25

Figure 2 : Fuzzy Region

recognize some handwritten samples of the seventeen character categories. In this section, the rule syntax and the network architecture are briefly described.

Rule Representation

Every rule in our system specifies a character pattern in terms of its subpatterns. Each subpattern may successively be further decomposed into simpler ones (Figure 1). The decomposition process is repeated until the subpatterns being used are the commonly accepted primitives. The set of primitives chosen is called the basic stroke set. Similar stroke sets, with slight variations, are widely adopted in various Chinese dictionaries, e.g., Cihai (Shangwu-Yinshuguan 1979). The collection of all rules related to the decomposition of a particular character can be viewed as its structure decomposition tree. Such a tree description closely resembles the syntactic analysis approach of Chinese characters studies (Chen 1986, Stallings 1977).

The form of a general rule is given as follows :

If {A, [(s,t), fuz_1]}
and {B_i, [(u,v), fuz_2]}
and .....
then E,

where the consequence E indicates the character pattern whose structural knowledge is described by this rule. Each of the antecedents holds the geometric information of one subpattern. Currently, only one writing style in each character pattern serves as its template. All rules are of conjunctive types because no subpatterns are considered as optional. Since a disjunctive rule can be transformed into a number of conjunctive rules, it seems to be quite straight forward to extend our current rule forms to include disjunctive ones.

Every antecedent consists of two parts, namely, a component pattern and its positional attributes. The subpattern B_i stands for the i-th variant of the stroke class B, while A is a subpattern which may be a radical or a subcharacter. In the positional attribute part, two fields of information are maintained. The first field denotes the mostly expected integral coordinates of the subpattern when the character pattern represented by the consequence indeed exists. The coordinates of the subpatterns in the antecedent part are calculated as follows. The character pattern denoted by the consequence is supposed to fall on a 17x17 pixel matrix. Then, for each of its subpatterns, a reference location on the matrix is associated with it, which is often chosen at approximately the centre of that subpattern. The same is done for the consequence pattern too. By treating the reference location of the consequence pattern as the origin, the relative coordinates for the subpatterns are calculated accordingly. These relative coordinates are the mostly expected coordinates. The second field defines a fuzzy region enclosing the location specified by the first field. A sample fuzzy region is shown in Figure 2. Values in the region specify the plausibilities for that antecedent subpattern to exist at the corresponding locations. Table I lists the values in several fuzzy regions. In a fuzzy region, each value is inversely proportional to the distance between the location concerned, [r', c'], and the mostly expected location [r_e, c_e]. Here, "distance" is defined by the function MAX(|r'-r_e|, |c'-c_e|).

Fuzzy Region	Distance from the expected location					
	0	1	2	3	4	5
Fuz_1	1.0	0.50				
Fuz_2	1.0	0.75	0.25			
Fuz_3	1.0	0.85	0.50	0.15		
Fuz_4	1.0	0.90	0.65	0.35	0.10	
Fuz_5	1.0	0.93	0.75	0.50	0.25	0.07

Table I

System Architecture

In Figure 3, the architecture of our proposed neural network system is shown. A 65x65 square pixel matrix at the left hand side is the input pattern grid for receiving binary input images. An input, i.e., a handwritten Chinese character to be recognized, will turn on the corresponding pixels on the matrix.

A stroke extractor is then employed to locate all possible instances of the basic strokes contained in the input character. At the output layer of this extractor, there are twenty-two groups of neuron-planes corresponding to the twenty-two basic stroke classes (Yeung, Fong & Cheung 1992). There are three types of transformation applied to the basic stroke in each class, namely, scaling, skewing and rotation. With three choices allowed for each transformation, up to a total of twenty-seven variants can be generated for each stroke class. Thus, there are up to twenty-seven neuron-planes in each group. Every neuron-plane at the output layer of this stroke extractor is a 17x17 matrix arrangement of neuron-modules. The locations of the detected stroke variants are "quantized" to these 17x17 matrix locations. The firing score of a neuron-module, which is bounded between 0.0 and 1.0, reflects the degree of matching with a particular stroke variant at a particular location.

In the rule induced neural network, each stage holds the structural knowledge of several patterns which may be characters or radicals. A pattern in each stage can be constructed from its components as detected in the preceding stages. A neuron-plane, which is a 17x17 matrix arrangement of neuron-modules, is allocated for the detection of each pattern. All neuron-modules on a neuron-plane are mapped with the structural knowledge of the same pattern. This duplication of knowledge is to

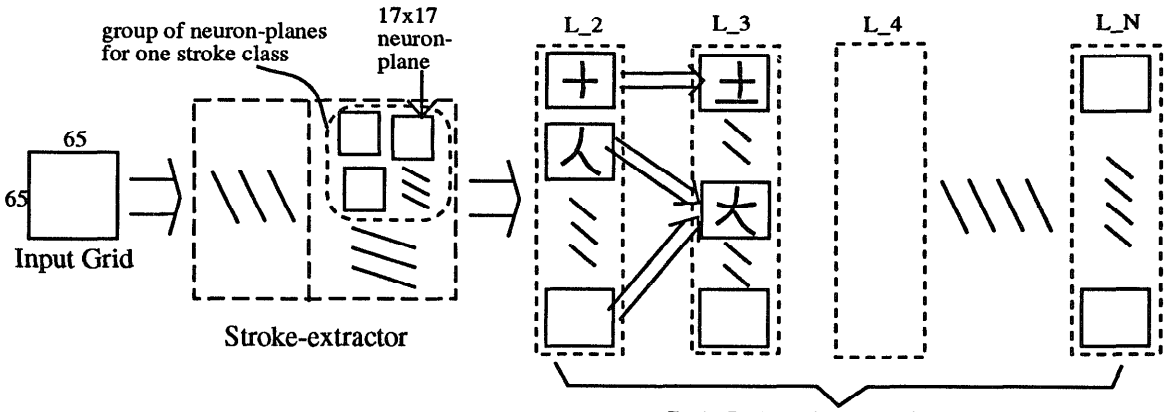

Figure 3 : System Architecture

facilitate positional shift tolerant detection of the pattern.

During the rule-mapping phase, the internal structure of a neuron-module is determined by the rule mapped onto it. There are two types of cells in each module, namely s-cell and p-cell. While there is always only one s-cell in each module, the number of p-cells in a module equals the number of antecedents in the mapped rule. Every p-cell is responsible for detecting the existence of a particular component in an acceptable region, by connection projected from the neuron-plane of that component pattern. This acceptable region is determined by both the location of the currently concerned module on the neuron-plane, and the positional attribute (section 2.1) of that component.

The rule induced network performs its recognition task stage by stage. A stage is labelled as a candidate-stage if it has a neuron-plane, which is associated to a character pattern, fire with non-zero response in one of its neuron-modules. After all stages have finished the processing, the last candidate-stage is checked. The input character is identified as the character pattern associated with the neuron-plane in that stage which fires with the highest score. If no candidate-stage is found at all, the input character is then rejected.

Knowledge Matrix

The knowledge matrix is introduced in this paper to function in two aspects. It is to facilitate reasoning of the network's response to an input character, and to drive modifications on the rule-base. The modified subset of rules can then be re-mapped onto the network to refine the network knowledge without degrading its comprehensibility. In the sections below, one will first see how the knowledge matrix and an auxiliary tool, namely the response matrix, are organized. Then, the ways these two matrices co-operate to perform the above functions are elaborated.

Matrix Representation

The knowledge matrix is a representation of the internal knowledge of the rule induced neural network (Figure 4). It has symmetric labels on its rows and columns. Every label is a 3-tuple, corresponding to a neuron-module in the network. For instance, a label (P,r,c) corresponds to the neuron-module at the location [r,c] on the neuron-plane for pattern P. Those rows (columns) which are closer to the upper (left) portion of the matrix are assigned labels corresponding to modules staying in earlier stages. Furthermore, those labels corresponding to the same pattern are grouped consecutively. Additionally, a non-diagonal, non-zero entry $K_{i,j}$ on row-i and column-j indicates a connection from module-j to module-i. The entry value is the maximally attainable excitation on that connection. In contrast, the diagonal entry-values indicate the highest firing scores attainable by the s-cells in the corresponding modules. Since the value-determination procedure for the entries in the knowledge matrix is quite complicated, those who are interested are suggested to refer to the authors' other paper (Yeung & Fong 1993).

According to this label-organization, the knowledge

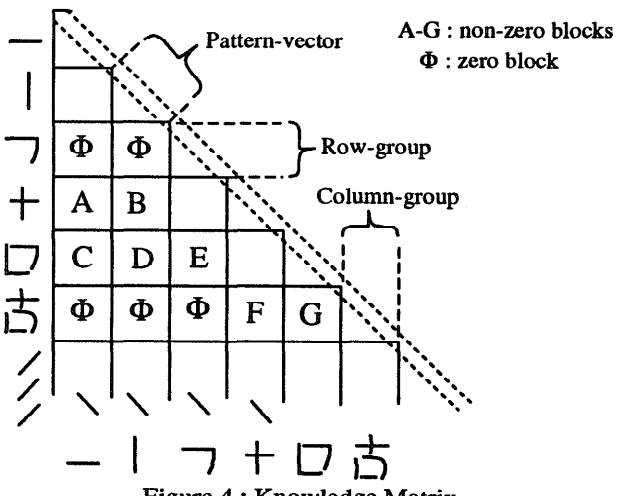

Figure 4 : Knowledge Matrix

matrix possesses several characteristics. Since the network allows connections only from lower stage to higher stage, all the entries at the upper-triangle of the matrix are set to zeros. Moreover, the rows (columns) of the knowledge matrix can be divided into *row-groups* (column-groups). All the row-labels (column-labels) in a row-group (column-group) correspond to modules from the same neuron-plane, whose associated pattern is thus called the *row-group pattern* (column-group pattern). The row-groups divide the diagonal on the knowledge matrix into partitions. Each of the partitions is named a *pattern-vector* of the associated row-group.

Another kind of unit on the matrix, called *block*, is also identifiable which is defined as the intersection region between a row-group and a column-group. Thus, a block is a submatrix associated with one row-group pattern and one column-group pattern. However, the following discussions concerning blocks should exclude those which cover the diagonal. A block is a non-zero matrix only if its row-group pattern possesses its column-group pattern as one component. Meanwhile, every row of entries in a block forms one *row-vector*. The entries in a row-vector express the maximum excitations on a set of connections, projected from a p-cell in the neuron-module associated to that row onto the neuron-plane of the column-group pattern.

While the knowledge matrix represents the static knowledge captured by the rule induced network, a response matrix is used to represent the network's run-time response to an input. The label-organization, definitions of row/column-group, pattern-vector, block and row-vector on the response matrix are identical to those in the knowledge matrix. The diagonal entries carry actual firing scores of each neuron-modules, and the non-diagonal ones carry actual excitations on the corresponding connections. The example of a response matrix is shown in Figure 5.

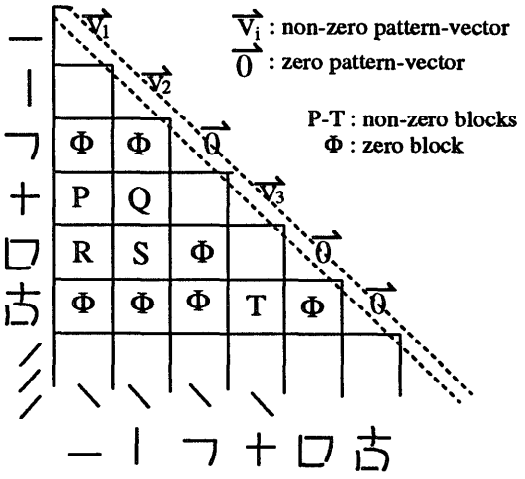

Figure 5 : Response Matrix

Explanation of the Network Response

With the help of the knowledge matrix (and also the response matrix), it is possible to query the network's response to an input in two ways :

(i) Why is pattern-A detected {at location [r,c]} ?
(ii) Why is pattern-A NOT detected ?

Note :The location portion {.} in query (i) is optional.

Two different algorithms are devised to handle these two types of queries.

Case (i) Why is pattern-A detected {at location [r,c]} ?
Algorithm A :
Steps

1) Check if pattern-A is really detected. If location is specified in the query, one can directly check if the diagonal entry on the corresponding row is non-zero. Otherwise, this can be done by tracing the row-group of pattern-A, on the response matrix, to see if it possesses a non-zero pattern-vector.
2) If the non-zero entry (or the non-zero pattern-vector) cannot be found in step 1), tell the user that pattern-A is NOT detected, or the query is invalid; then STOP.
3) If location is specified in the query, e.g., [r,c], then choose the target row Rr with row-label (A,r,c). Otherwise, in the row-group of pattern-A on the response matrix, locate the target row Rr with the highest diagonal entry-value. If more than one row possesses the highest value, choose one of them arbitrarily.
4) Extract the corresponding row, Kr, on the knowledge matrix.
5) For each of the non-zero row-vectors along Kr, find the corresponding row-vectors VRr_i on the response matrix.
6) On every row-vector VRr_i, do the steps below.
> Identify the highest excitation value. If more than one is found, choose one of them arbitrarily.
> Suppose the entry identified above is $M_{a,b}$. Extract row-b on the response matrix.

7) Suppose every row-b_i extracted in step 5) has a 3-tuple label (P_i, r_i, c_i), and the diagonal entry value on it equals V_i. Construct an explanation in the following format :

"Pattern-A is detected because
subpattern P_1 is detected at location [r_1, c_1] with score V_1, and
.....
subpattern P_i is detected at location [r_i, c_i] with score V_i, and
....."

8) STOP.

The user may continue the query on any particular pattern P_k above. However, if P_k is a stroke variant, stroke extraction stage is reached and the user should be notified that no more detail is available.

Case (ii) Why is pattern-A NOT detected ?
Algorithm B :

Steps

1) Check if the pattern-vector on the row-group of pattern-A on the response matrix is a zero vector. If not so, tell the user that pattern-A has been detected, or the query is invalid; then STOP.

2) In the row-group of pattern-A on the knowledge matrix, identify all the non-zero blocks. Then, locate the corresponding blocks Rm_j on the response matrix.

3) If some Rm_j are zero matrices, do as follows.

(i) Suppose each zero block Rm_i corresponds to column-group pattern P_i.

(ii) Construct an explanation in the following format :

"Pattern-A is NOT detected because
subpattern P_1 is NOT detected, and
.....
subpattern P_i is NOT detected, and
....."

(iii) STOP algorithm B.

4) Trace the row-group of pattern-A on the response matrix until a row is encountered along which all row-vectors in all Rm_j are non-zero vectors.

If such a row exists, give an explanation :

"Pattern-A is NOT detected because the degree of match is too low."

Otherwise, give the following explanation :

"Pattern-A is NOT detected because the positional information does not fit the input."

5) STOP.

The two types of queries can provide much information on the network's response. However, there are other types of queries which may be useful. In addition, details added onto the last explanation offered in step 4) of algorithm B will make it more informative. The situation can be understood better if one is informed that "the tolerance region of a subpattern *X* is too small", "the expected location of a subpattern *X* is too close", etc. Nevertheless, the algorithms presented above improve the transparency of the operation of a neural network. More work is to be done along this direction.

Rule Modifications

In this section, the knowledge matrix and the response matrix are adopted to address the problem of knowledge refinement. Most neural network based systems employ learning algorithms which bear no explicit relationship with the problem domain. In contrast, the proposed approach attempts to adjust the knowledge represented in the rule-base, according to the network's response to an input. The refined rules can then be used to update the network connections through mapping.

In refining our system, two major error types are handled, namely wrong-rejections and wrong-recognitions. A wrong-rejection error occurs when none of the neuron-planes responds to the input character which is expected to be recognizable. This type of error requires raising the tolerance (or fuzziness) of the related rules. An algorithm for this purpose has already been devised in another paper of the authors (Yeung & Fong 1993). Thus, we will focus on the second error type here, i.e., wrong-recognition.

There are two different cases for this type of error. One is that the relevant neuron-plane doesn't respond at all, while some other irrelevant ones do. Another case is that the response of the relevant neuron-plane is suppressed by the response from neuron-planes at this or higher stages of the network. For the first case, action adopted in handling the wrong-rejection error is taken to activate the relevant plane. Thus, the first case is transformed into the second case.

At least two situations are identified as the possible causes of the second case of wrong-recognition error. The noise generated by the stroke-extractor in the system, i.e., false stroke detections, may cause irrelevant patterns to be found. Since this is the problem of the stroke-extractor, it is not considered here. Another cause is that some rules (mapped onto the irrelevant neuron-planes) may be too sensitive to give "false" response to the input. A tuning algorithm is given below to tune these rules, so as to suppress responses from irrelevant neuron-planes. Before the algorithm is presented, several definitions employed in the algorithm are described beforehand.

Definitions

(1) Poorly-Done pattern, PD

The entry-values on a row-vector are calculated from the pattern-vector on the same column-group. If the maximum entry on a row-vector corresponds to an element in the pattern-vector which is smaller than a preset threshold (e.g., 0.1), the column-group is said to be PD. Adjustment on the structure knowledge of this column-group pattern will not be attempted.

(2) Low-Positional Fuzziness, LPF

An antecedent is said to possess LPF if the fuzzy region (associated with an antecedent in a rule) has been reduced to a preset lower bound (e.g., 1). Its fuzzy region in that rule should not be shrunk any more.

(3) Sit-Well row-vector

Each row-vector on the knowledge matrix corresponds to an antecedent of the rule associated to that row. If the maximum entry of a row-vector is found to be located at the centre of the fuzzy region of the corresponding antecedent, the row-vector is said to be Sit-Well.

(4) Innocent row-vector

When a row-vector possesses LPF or is Sit-Well, and its column-group pattern is PD, the row-vector will not be considered as a possible item whose changes may lead to any improvement of the situation.

Tuning Algorithm

Suppose the input belongs to the character category A whose structure knowledge has been mapped onto a plane at stage-N. Then, the row-groups on the response matrix are traced one by one, from the highest stage to stage-N. For any row-group G other than that for pattern A, if the pattern-vector is a non-zero vector, perform the routine

<Correct> below. One should note that, in the algorithm below, any adjustment on a rule is analogous to some alterations on the network. Thus, after every rule-adjustment, both the knowledge matrix and the response matrix must be updated before the tuning process proceeds.

<Correct>
(1) Select the row **r** on G corresponding to the maximum entry in the pattern-vector.
(2) Repeat
  (2.1) Record all innocent row-vectors on **r**.
  (2.2) Among the non-innocent row-vectors on **r**, choose one $\mathbf{r}_i$ which possesses the maximum entry.
  (2.3) Check if $\mathbf{r}_i$ is marked. If yes, perform 2.3a; else, follow 2.3b.
  (2.3a) Identify the maximum entry in $\mathbf{r}_i$, say, $RM_{a,b}$, and unmark $\mathbf{r}_i$. Then <Component-correct> is performed for the row **b**.
  (2.3b) Unmark all other marked row-vectors on **r** and mark $\mathbf{r}_i$. Reduce the fuzzy region radius by one of the antecedent associated to $\mathbf{r}_i$ if this row-vector does not possess LPF.
  Until {G possesses a zero pattern-vector, or all the antecedents of the associated rule possess LPF}
(3) Unmark all row-vectors on the response matrix.
(4) STOP.

<Component-correct>
Suppose the row concerned is **r'**, which is corresponding to a rule R_i.
(1) Record all innocent row-vectors on **r'**.
(2) Among the non-innocent row-vectors, choose one, e.g., $\mathbf{r}_i$', which possesses the maximum entry value.
(3) If $\mathbf{r}_i$' is marked but is PD, unmark it. Go to step (6).
(4) If $\mathbf{r}_i$' is marked, identify the maximum entry in $\mathbf{r}_i$', say, $RM_{s,t}$. Perform <Component-correct> for the row **t**.
(5) If $\mathbf{r}_i$' is not marked, unmark all other marked row-vectors on **r'** and mark $\mathbf{r}_i$'. Then, expand the fuzzy region of the antecedent in R_i associated to $\mathbf{r}_i$'.
(6) STOP.

It is possible that the pattern-vector of the row-group G cannot be suppressed completely after tuning. In this case, the algorithm is unable to resolve the ambiguity arisen between the input character and the structure rule associated to G.

Conclusion

In this paper, two shortcomings which hinder wider applications of neural networks are tackled. The neural network considered is induced by the structure rules of a set of Chinese characters. By introducing a facility called knowledge matrix, the network's responses can be explained to users. This type of explanation is seldom available in other neural network based systems. Additionally, the knowledge matrix is also employed to drive refinements on the rule-base. Re-mapping of the modified rules onto the network completes the adaptation cycle. This approach of rule refinement keeps the network comprehensible to users, which is another characteristic that cannot be easily achieved in learning algorithms adopted by most neural networks.

At present, only a few types of explanations with respect to the network's responses are offered. Explanations with more details are surely advantageous. Moreover, the performance of the rule refinement algorithms proposed needs to be studied. All these will require a more in depth investigation on the knowledge matrix.

References

Chen, K.J. 1986. Computational Approaches in Topological and Geometrical Descriptions for Chinese Characters. *Computer Processing of Chinese & Oriental Languages* 2(4): 234-242.

Shangwu-Yinshuguan 1979. *Cihai*. Beijing, Shanghai: Shangwu Yinshuguan (The Commercial Press).

Fu, L.M.; Fu, L.C. 1990. Mapping rule-based systems into neural architecture. *Knowledge-Based Systems* (3)1: 48-56.

Gallant, I. 1988. Connectionist Expert Systems. *Communications of the ACM* 31(2): 152-169.

Hayashi, Y.; Krishnamraju, V.; Reilly, D. 1991. An Architecture for Hybrid Expert Systems. In Proceedings of the IEEE International Joint Conference on Neural Networks, '91, Singapore, 2773-2778.

Stallings, W. 1977. Chinese Character Recognition. In Fu, K.S. eds. 1977. *Syntactic Pattern Recognition, Applications*, 95-123. New York : Springer-Verlay.

Towell, G.G.; Shavlik, J.W.; Noordewier, M.O. 1990. Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks. In Proceedings of the 8th National Conference on Artificial Intelligence '90, 2: 861-866.

Towell, G.G.; Shavlik, J.W. 1992. Interpretation of Artificial Neural Networks. In Moody, J.E. eds. 1992. *Advances in Neural Information Processing Systems Vol.4*, 977-984. Morgan Kaufmann Pub..

Yeung, S.; Fong, H.S.; Cheung, K.F. 1992. A Neocognitron-based Chinese Character Recognition System. In Proceedings of the International Joint Conference on Neural Networks '92, Beijing, 3: 617-622.

Yeung, S.; Fong, H.S. 1993. A Knowledge Matrix Representation for a Rule-Mapped Neural Network. (To appear in) *Neurocomputing*.