34

Rule-based Dating of Artefacts

Kazumasa Ozawa*

34.1 Introduction

Many types of expert systems have recently been developed. We face many traditional problems to which expert systems seem applicable. Empirically acquired knowledge has played a key-role in solving such types of problems. A famous and successful example of an expert system is MYCIN which handles the problem of diagnosing blood diseases (Shortliffe 1976).

Knowledge or expertise is basically regarded as a vague continuous body. The first step towards an expert system is to segment such a continuous body of expertise into unit rules. Then a knowledge-base is given by a collection of rules to solve problems in a given domain.

Empirically acquired knowledge often plays a very important role in archaeological dating. When the dating objectives are limited to a small class of artefacts, a huge amount of knowledge is not necessarily required so that a computer may be used in rule-based dating operations instead of an archaeologist. An artefact invloves a large number of dating components, each of which manifests an individual trait of temporal change. Archaeological knowledge is approximately equivelent to a set of dating rules referring to those temporal changes. This paper presents a model of the archaeological dating process and its application to building a dating expert system.

34.2 Modelling

Archaeological dating methods can be classified into two types, i.e. absolute dating and relative dating. As is well-known, absolute dating methods have been established on the physical basis as seen in radio-carbon dating. On the contrary, most relative dating operations have been based on the knowledge and inference of archaeologists. In Japanese archaeology, relative dating such as seriation has also played a very important role in the understanding of temporial traits of artefacts along with absolute dating. This paper will address such relative dating as based on archaeological knowledge and inference, termed *rule-based dating*.

Rule-based dating expert system should be designed referring to the typical process of archaeologists' dating operations. In the following, a model of the dating process is presented, which will be referred to for our system implementation.

An artefact involves a number of dating components. Let **x** be an artefact belonging to such a properly limited class A as associated with a constant tuple of N dating components, designated 1 through N. Then we define **x** as

$$x = (x_1, x_2, \ldots, x_N). \qquad (34.1)$$

where $x_1, x_2, \ldots, x_N$ are values assigned to dating components $1, 2, \ldots, N$, respectively.

Suppose that A is a properly limited class of pottery such that its dating components include decoration, colour, source material, technical skill, style, find spot and

* Osaka Electro-Communication University
Neyagawa-shi,
Osaka 572,
Japan

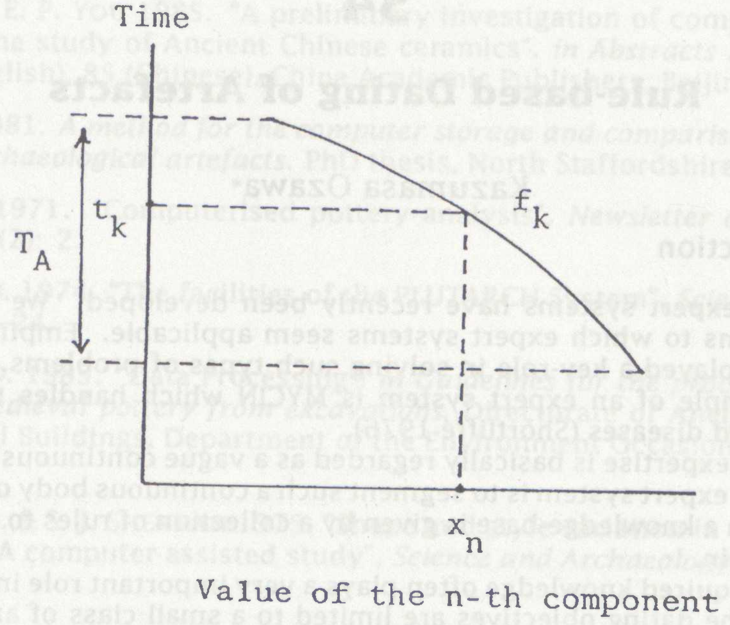


Figure 34.1: A mapping function of a single variable

others. Let a pot **x** in A be written by (34.1). Each of $x_1, \ldots, x_N$ is either numerical or non-numerical; e.g. when a component n is non-numerical 'colour', its value x_n would be red, brown or black. In general, the archaeological dating operation can be regarded as a mapping of the values $x_1, \ldots, x_N$ into a time range associated with the class: For **x** in (34.1), we define such a mapping function $f_k (k = 1, \ldots, K)$ as

$$f_k(x_1, \ldots, x_N) = t_k \; in \; T_A \tag{34.2}$$

where t_k and T_A are an estimate of f_k and the time range of A, respectively.

Fig. 34.1 illustrates (34.2) by employing an assumed function f_k of a single numerical variable x_n. In Fig. 34.1, the estimate t_k is presented by a point in the time range. But t_k would mostly be a time interval, termed *a temporal segment*, in the time range: for a non-numerical dating component like 'colour', f_k can be given by a set of rules such as 'black → early', 'brown → middle' and 'red → late'.

For an artefact **x** with N values as is in (34.1), we have a tuple of K estimates through the mapping procedure defined in (34.2). Symbolically, we have

$$M(x_1, x_2, \ldots, x_N) = (t_1, t_2, \ldots, t_K). \tag{34.3}$$

where M denotes an assumed operation that integrates the whole of individual operations by mapping functions $f_1, \ldots, f_k$

Another operation succeeding to M is the unification that balances or compromises all the estimates $t_1, \ldots, t_K$ and, eventually, unifies them into a single value t_x, i.e. the estimated date of **x**. This operation can be written by

$$U(t_1, t_2, \ldots, t_K) = t_x. \tag{34.4}$$

From (34.1), (34.3) and (34.4), we have a compact notation as follows:

$$UMx = t_x \tag{34.5}$$

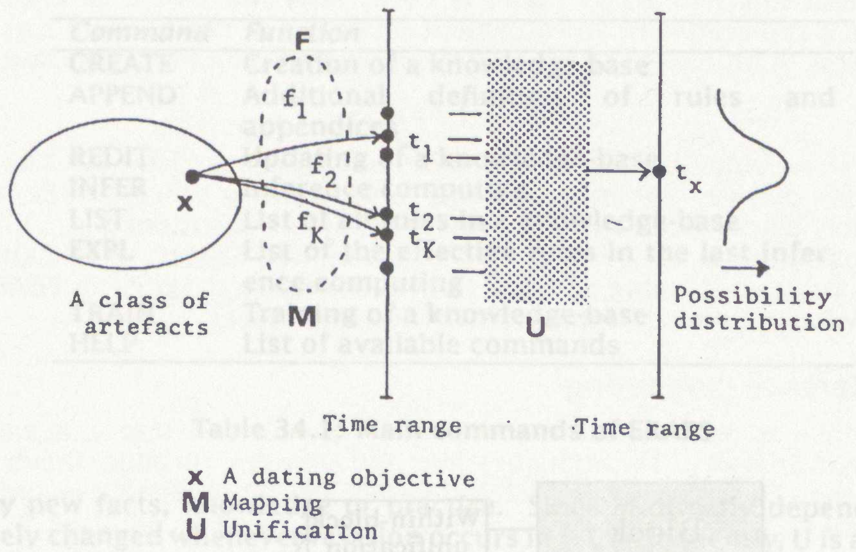


Figure 34.2: A model of rule-based dating

This model means that for a given artefact **x** in A, by operating M first and U second, we obtain its estimated date t_x. Fig. 34.2 illustrates the model of rule-based dating given by (34.5). Where, the knowledge-base F is nothing but a set of mapping functions. Symbolically,

$$F = \{f_1, f_2, \ldots, f_K\}. \tag{34.6}$$

In principle, our model has been built on the basis of the tuple representation defined as (34.1). All the values $x_1, \ldots, x_N$ should practically be evaluated by inspection or action on the artefact: This will be referred to as *evaluation process.*

The operation M is carried out in cooperation with the knowledge-base F. The relation between M and F can be compared to that of cooking and a cookbook; when given materials, cooking produces a course of dishes according to the cookbook. Practically, every mapping function can be given by a collection of if-then rules, so that F can also be regarded as a set of rules.

All the mapping functions in F are necessaily not independent: Several functions are often related to the same aspect of artefacts. It follows that F should be partitioned into the blocks of functions each of which concerns a special aspect. Partition of F is closely related to the organisation of U. A possible understanding is that U may be in charge of two stages of unification; *within-block* and *inter-block* unification. At the stage of within-block unification, all estimates determined by mapping funcitons within a block may be mediated and reduced into a block-representative estimate, termed a *B-estimate.* Then all B-estimates may be compromised and unified into the final value t_x; this stage is that of inter-block unification. Fig. 34.3 illustrates such two stages of unification.

Another problem to be noted concerns the learning or self-organizing capability. Every archaeologist appears to be in a ceaseless learning process: Whenever a new fact occurs, his dating procedure may be revised and his knowledge may also be updated. When wrong knowledge is detected in practice it may immediately be corrected. On the other hand, good knowledge gradually gains in importance through practice. Such a special quality can be discussed in terms of our model in Fig. 34.2: The knowledge-base F is the pivot of learning . Functions or rules in F are constantly

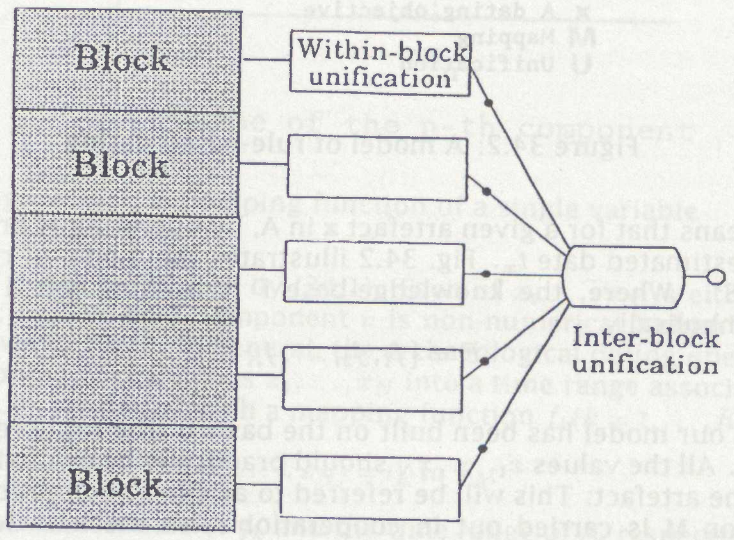


Figure 34.3: Two stages of unification. • and ∘ denote a B-estimate and the final conclusion, respectively.

Command	*Function*
CREATE	Creation of a knowledge base
APPEND	Additional definition of rules and appendices
REDIT	Updating of a knowledge-base
INFER	Inference computing
LIST	List of all rules in a knowledge-base
EXPL	List of the effective rules in the last inference computing
TRAIN	Training of a knowledge-base
HELP	List of available commands

Table 34.1: Main commands of ERAPS

revised by new facts, knowledge or practice. Since M directly depends on F, M is immediately changed whenever revision occurs in F. Consequently, U is also changed, especially at the stage of within-block unification. In fact, learning capability plays a key-role in keeping up-to-date dating operatons. Our archaeological knowledge will never be constant, but will be in ceaseless change.

34.3 System implementation

An archaeological dating expert system, RAPS, built in 1983, provides the rule-based dating operations on a special type of Japanese ancient tombs, termed *Keyhole tombs* (Ozawa 1988). Since RAPS is, however, specialized only in the dating of the Keyhole tombs, it has never been applied to other dating problems.

To make it extend to general use, ERAPS (the Essential core of RAPS) has been organised (Ozawa 1986). In other words, ERAPS is an expert system building tool with the same inference engine as RAPS. Athough ERAPS provides a receptacle for knowledge it is scarcely an expert system until a knowledge-base is created. Hereafter, we discuss inplementation of a rule-based dating expert system using ERAPS in the light of our model stated previously.

34.3.1 Knowledge-base

ERAPS is available using the Franz Lisp interpreter or compiler under UNIX (4.2 bsd). Table 34.1 shows the main commands provided by ERAPS. The first step is building a dating expert system is to create a knowledge-base as a set of rules. Basically every rule should be written by

$$\text{if } R \text{ then } Q \text{ with } p. \tag{34.7}$$

where R, Q and p are *condition, conclusion* and *possibility*, respectively. The possibility p is given by a real constant between 0 and 1, which quantitatively indicates the confidence in the validity of Q when R is absolutely true. To make this clear, p will be referred to as *P-constant*. In this paper, the term 'possibility' will be used, instead of 'probability', to express a subjective measure of confidence or certainty.

The condition R is given by a logical combination of dating components; for example, we have

R = 'colour is red' and 'diameter is smaller than 20cm'.

In most cases, Q is an estimate corresponding to (34.2), associating with the P-constant p.

Every original rule written by (34.7) is necessarily not stored in ERAPS as it is. Sometimes it is replaced with an equivalent collection of two or more rules in terms of optimization. A unit rule stored in ERAPS is written in the S-expression such as

$$(I((R)((Q.p))).w). \tag{34.8}$$

Where I and w are rule-identifying number and weight, respectively. The weight w, discussed later, plays a very important role in the training process of ERAPS.

As previously discussed, a knowledge-base should be partitioned into blocks, each of which consists of all the rules related to a common aspect. Practically we have a number of rules concerning an aspect such as 'technical skill'. In ERAPS, we can define a knowledge-base so as to be partitioned into blocks.

34.3.2 Inference computing

The condition R of each rule is successively transformed into a question to be presented on the display. Through questions and answers, certainty factors of all the conditions are to be evaluated by a user of ERAPS. Certainty factor *a* is a real number in the interval [0, 1] that indicates quantified certainty of a condition R; when R is false, *a* is set to 0 and when R is absolutely true, *a* is set to 1. Frequently, the certainty factor takes an intermediate value between 0 and 1, because most artefacts are not complete so that dating involves uncertainty. The maximum uncertainty takes place when dating components related to R are absent; in this case, the certainty factor should be set to 0.5.

Inference computing in ERAPS has been carried out, basically, in the same manner as EMYCIN (van Melle *et al.* 1981). Our inference computing has been divided into two stages, i.e. *within-block* and *inter-block* computing. The within-block computing eventually determines a block-representative estimate, termed a *B-estimate*. Practically a B-estimate is not a single value, but a tuple of temporal segments with possibilities. For example, suppose that we have three temporal segments 'early', 'middle' and 'late' in the time range. Then a B-estimate can be written by $(P_E,\ P_M,\ P_L)$; in this case , P_E, P_M and P_L denote the possiblilities of the three temporal segments, each of which is derived from its related rules within the block.

For simplicity, consider the within-block computing in relation to a temporal segment in a B-estimate. Fig. 34.4 presents a very simple example to illustrate our computing procedure. From all rules concerning a common temporal segment, an estimate of the possibility derived through AND/OR operations. AND operations are mostly applied to condition parts of rules. On the other hand, OR operatins mainly act to mediate the estimates, providing possibilities of temporal segments.

In case of rule A in Fig. 34.4, its condition part R is given by an AND combination of two conditions denoted by the two bottommost nodes. Suppose that for the two nodes, certainty factors *a* and *a'* are assigned as shown in Fig. 34.4. Then the estimate $p_A{}^*$, derived from rule A, is computed as follows:

$$p_A * = p_A \cdot w_A \cdot Min\ \{a, a'\} \tag{34.9}$$

where p_A and w_A are the P-constant and weight of rule A, respectively. 'Min' denotes the AND operation that selects the minimum among certainty factors involved in the condition part of rule A.

The concluding possibility of a temporal segment is determined by an OR operation. for the three rules shown in Fi. 34.4, we have the following OR operation:

$$P = Max\ \{p_A{}^*,\ p_B{}^*,\ p_C{}^*\} \tag{34.10}$$

Namely OR operation selects the maximum among the estimates derived from all rules associated with a temporal segment. We call a sequence of the rules that provide

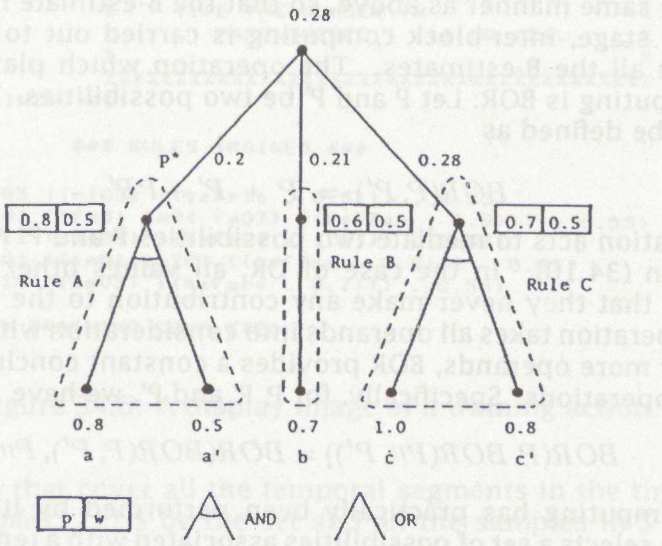
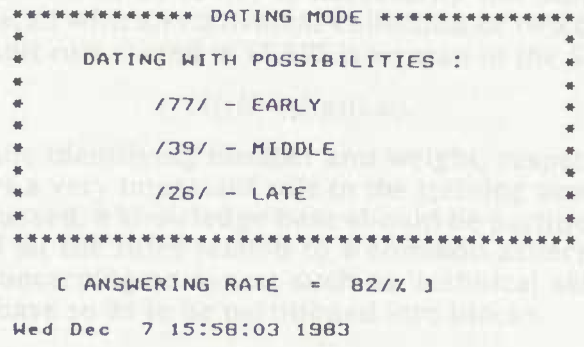


Figure 34.4: AND/OR operations in a simple block



```
************* DATING MODE **************
*                                      *
*  DATING WITH POSSIBILITIES :         *
*                                      *
*           /77/ - EARLY               *
*                                      *
*           /39/ - MIDDLE              *
*                                      *
*           /26/ - LATE                *
*                                      *
****************************************

        [ ANSWERING RATE  =  82% ]
      Wed Dec  7 15:58:03 1983
```

Figure 34.5: An example of dating conclusion

the maximum as *effective rules*. Possibilities of the other temporal segments are also computed in the same manner as above, so that the B-estimate may be determined.

As the second stage, inter-block computing is carried out to determine the final conclusion from all the B-estimates. The operation which plays a key-role in the inter-block computing is BOR: Let P and P' be two possibilities. Then BOR operation for P and P' can be defined as

$$BOR(P, P') = P + P' - PP'. \tag{34.11}$$

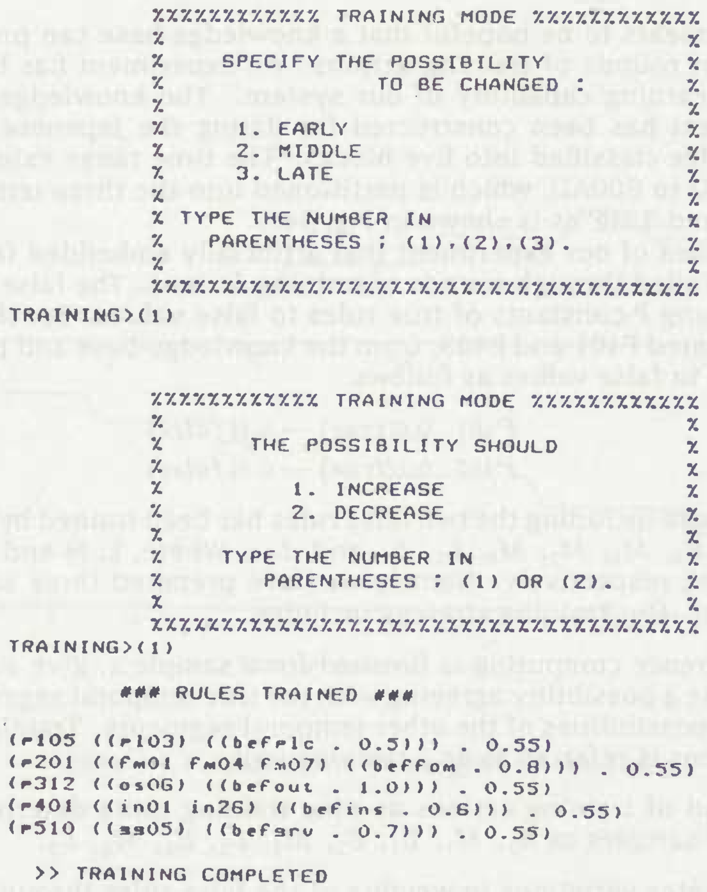
The BOR operation acts to mediate two possibilities P and P' more positively than OR as defined in (34.10). In the case of OR, all values other than the maximum are rejected, so that they never make any contribution to the final conclusion. In contrast, BOR operation takes all operands into consideration without rejection. Note that for three or more operands, BOR provides a constant conclusion independently of sequence of operations. Specifically, for P, P' and P', we have

$$BOR(P, BOR(P'', P')) = BOR(BOR(P, P'), P''). \tag{34.12}$$

Inter-block computing has practically been performed by iteration of the BOR operation which selects a set of possibilities associated with a temporal segment from all the B-estimates. Then, for the first two possibilities P and P'' in the set, the BOR operation is performed as in (34.11). Next, $BOR(P, P')$, thus obtained, and another possibility P'' in the set are operated, as is seen in the right hand of (34.12). This procedure is to be iterated until all are finished. consequently, the final possibility of the temporal segment is determined. This iteration is also applied to the other temporal segments so that the final possibilities of all the temporal segments may eventually be determined. Fig. 34.5 shows an example of such dating conclusion presented by ERAPS; in this case, the time range consists of three temporal segments, i.e. 'Early', 'Middle' and 'Late'.

34.3.3 Training procedure

As seen in Fig. 34.5, for a given dating objective, ERAPS presents a tuple of temporal segments with possibilities as a conclusion. It should be noted that such a conclusion as provided by a dating expert system is not constantly reasonable. A reason for this problem is that the knowledge-base is not always properly controlled, especially as the initial setup often involves useless rules, false definition of P-constants and other irrelevances. The TRAIN command is available to tune up the existing knowledge-base, asymptotically, though the training procedure as follows: prepare a set of



```
                %%%%%%%%%%%% TRAINING MODE %%%%%%%%%%%%
                %                                     %
                %    SPECIFY THE POSSIBILITY          %
                %               TO BE CHANGED :       %
                %                                     %
                %         1. EARLY                    %
                %         2. MIDDLE                   %
                %         3. LATE                     %
                %                                     %
                % TYPE THE NUMBER IN                  %
                %    PARENTHESES : (1),(2),(3).       %
                %                                     %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TRAINING>(1)

                %%%%%%%%%%%% TRAINING MODE %%%%%%%%%%%%
                %                                     %
                %    THE POSSIBILITY SHOULD           %
                %                                     %
                %        1. INCREASE                  %
                %        2. DECREASE                  %
                %                                     %
                %    TYPE THE NUMBER IN               %
                %      PARENTHESES : (1) OR (2).      %
                %                                     %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TRAINING>(1)

          ### RULES TRAINED ###

(r105 ((r103) ((befrlc . 0.5))) . 0.55)
(r201 ((fm01 fm04 fm07) ((beffom . 0.8))) . 0.55)
(r312 ((os0G) ((befout . 1.0))) . 0.55)
(r401 ((in01 in26) ((befins . 0.6))) . 0.55)
(r510 ((ss05) ((befsrv . 0.7))) . 0.55)

   >> TRAINING COMPLETED
```

Figure 34.6: A display image of a training action

the known samples that cover all the temporal segments in the time range, termed *training set of samples.* Let S be the set and all the samples in S be designated in temporal order.

First, select a sample x from S and make ERAPS perform its inference computing. Then we have a dating conclusion in terms of a tuple of temporal segments with possibilities. Here we can put TRAIN in action: If the conclusion is not resonable, for example, the possibility of the true temporal segment is relatively too small, then we can give such instruction as to increase it. On the contrary, sometimes, we give an instruction to decrease another possibility opposed to the truth. According to such instruction, ERAPS automatically modifies weights of the rules that were effective in the last inference computing. As a matter of course, if the conclusion is reasonable, we have nothing to do. Fig. 34.6 shows a display image just before the training action on the conclusion presented by Fig. 34.5. In the same way as above, our training actions are taken by the other samples one after another. A round of training actions is completed when all samples in S have been employed.

From a technical point of view, a training instruction given to the system is to be executed by adding 0.05 or -0.05 to the weight of every rule effective in the last inference computing. Where weights of all rules are initially set with 0.5, it is prohibited that they increase or decrease beyond the interval [0, 1] thorough the training process.

Practically it appears to be hopeful that a knowledge-base can properly be tuned through sufficient rounds of training actions. An experiment has been carried out to examine the learning capability of our system. The knowledge-base employed for our experiment has been constructed for dating the Japanese ancient tombs, including 126 rules classified into five blocks. The time range extends for the 300 years from 300AD to 600AD, which is partitioned into the three temporal segments 'Early', 'Middle', and 'Late' as is shown in Fig. 34.5.

It is the basic idea of our experiment that artificially embedded false rules would properly be controlled through rounds of training actions. The false rules have been created by changing P-constants of true rules to false values. Specifically we select two rules, designated P401 and P403, from the knowledge-base and puposely change their P-constants to false values as follows:

$$P401 \quad 0.6(true) \rightarrow 0.1(false)$$
$$P403 \quad 0.1(true) \rightarrow 0.6(false)$$

The knowledge-base including the two false rules has been trained by employing nine samples E_1, E_2, E_3, M_1, M_2, M_3, L_1, L_2 and L_3. Where, E, M and L means 'Early', 'Middle' and 'Late', respectively. Namely we have prepared three samples for each temporal segment. Our training strategy includes:

I. Whenever inference computing is finished for a sample x, give such instructions as to increase a possibility agreeing with the true temporal segement of x and as to decrease possibilities of the other temporal segments. Training by this series of instructions is referred to as a *training unit*.

II. Define a round of training actions as nine training units determined by such a sequence of samples as E_1, M_1, L_1, E_2, M_2, L_2, E_3, M_3, L_3.

Fig. 34.7 illustrates variations in weights of the false rules through five rounds of training (45 units). Note that the weight of P401 favourably increases in proportion to times of training. This means that based on (34.9), the system acts to compensate the false P-constant of P401 by increasing its weight. On the contrary, for P403, compensation is done by decreasing the weight. P209, seen in the figure, is one of the properly defined rules, of which weight is almost uniform except for periodical small spikes.

Fig. 34.7 appears to suggest that the learning capability of our system is so favourable that the knowledge-base can properly be controlled through rounds of training actions. In fact, as is seen in Table 34.2, two different conclusions for each sample, computed respectively after and before five rounds of training, suggest that our training actions have obvously improved the knowledge-base, providing a better conclusion for each sample than before the training.

34.4 Conclusion

In this paper, emphasis has been placed on modeling of the archaeological rule-based dating operation and its application to an expert system implementation. Archaeologists' dating operation would actually involve more complicated processes than those of the proposed model. It follows that other types of modeling would be required and would also be possible independently of ours. Different models bring forth different expet systems. From this point of view, ERAPS can be regarded as a tool to build dating expert systems provided that the expertise can approximately be written in the form of '*if* R *then* Q *with* p'.

Two knowledge-bases, TOMB and E-HANIWA, have been established in ERAPS; TOMB serves for dating the Keyhole tombs, which are frequently referred to in this paper.

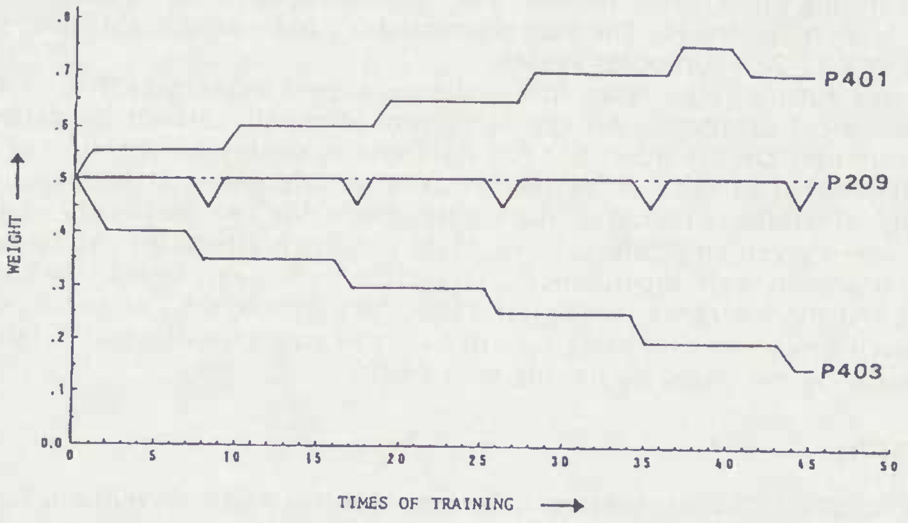


Figure 34.7: Variations in weights of the rules

Tomb	Early	Middle	Late
		Possibilities	
E_1	0.95(72)	0.26(39)	0.21(39)
E_2	0.86(78)	0.72(67)	0.71(57)
E_3	0.93(68)	0.28(42)	0.26(35)
M_1	0.38(52)	0.92(72)	0.46(64)
M_2	0.41(60)	0.97(81)	0.27(47)
M_3	0.38(52)	0.95(77)	0.34(59)
L_1	0.29(29)	0.26(48)	0.95(79)
L_2	0.39(61)	0.38(48)	0.91(68)
L_3	0.14(14)	0.36(48)	0.78(55)

Table 34.2: Two different conclusions for each sample. Every possibility computed before training is shown in parentheses

'Haniwa' are a set of all the terra-cotta figures which stand on the Keyhole tomb mounds. The knowledge-base E-HANIWA has been created for the dating of a special class of Haniwa, i.e. cylinder-shaped terra-cotta.

As mentioned previously, ERAPS is available using the Franz Lisp interpreter or compiler running under UNIX 4.2 bsd. The system needs about 70 KB memory space excluding knowledge-bases. The experimental work presented in this paper has been done on a VAX 11-750 computer system.

One of our future tasks is to link a dating expert system such as ERAPS with an archaeological database. An expert system is usually driven by data provided from the outside. On the other hand, a database is simply a collection of objective facts, independent of such an active function as inference. It appears that every archaeological database includes many null values due to uncertainty of attributes. However, when given an artefact, some of its unknown attributes can sometimes be evaluated based on their dependency on the other known attributes. Such evaluation is nothing but the inference that can possibly be performed by an expert system. It is hoped that unknown chronological attributes in an archaeological database could automatically be evaluated by linking with ERAPS.

Bibliography

OZAWA, K. 1986. "Expert systems", *Systems Control and Information*, 30 (4): 213–219. (in Japanese).

OZAWA, K. 1988. "An archaeological research support system databases of Japanese ancient tombs and geographies". paper presented at Cologne Computer Conference.

SHORTLIFFE, E. H. 1976. *Computer-based medical consultation: MYCIN.* American Elsevier, New York.

VAN MELLE, W., E. H. SHORTLIFFE, & B. G. BUCHANAN 1981. "EMYCIN: A domain-independent system that aids in constructing knowledge-based consultation programs". *in Machine Intelligence*, Infotech State of the Art Report 9. 3.