6

Interval Rule Matrices for Decision Making

Chenyi Hu

Department of Computer Science, University of Central Arkansas, 201 Donaghey Avenue, Conway, AR 72035-0001, USA. chu@uca.edu

In this chapter, we present a decision-making system using an interval rule matrix. Section 6.2 introduces the rule matrix model. Section 6.3 reports practical algorithms that establish an interval rule matrix. Section 6.4 describes how to use an interval rule matrix to make decisions according to environmental observations.

6.1 Introduction

Every day, numerous decisions are made for various reasons. Although many of these decisions are astute enough and return results as expected, there are others that can be made better to avoid unwanted consequences. How to make a good, a better, or even the best decision is a practical question asked frequently.

A decision problem usually consists of a set of possible states of nature (environment), a set of possible actions (decisions), and a benefit function that measures the results of a decision in a given environment. To make a right decision, one certainly needs to know the current environment and related knowledge. Through matching input data (relevance of each environment feature) with a certain set of rules, one may make a decision accordingly. However, due to uncertainty and incomplete information, the best one can do is to estimate the benefit for a decision. Usually, the actual best decision can only be known afterward. Therefore, historical data are often useful for people to discover rules for decision making.

As the world becomes more complex, the number of features involved in decision making can be unmanageable for human beings. Modern computers collect massive datasets and perform trillions of calculations in seconds. Nowadays, knowledge-based agents are designed, implemented, and embedded into computers as automated decision-making systems. These systems apply decision theories and algorithms to generate rules based on statistical/probabilistic, stochastic, fuzzy systems [5, 11, 12, 15], neuro-fuzzy systems

C. Hu et al. (eds.), *Knowledge Processing with Interval and Soft Computing*,
DOI: 10.1007/978-1-84800-326-2_6,

[9], and so on. In this chapter, we specifically study an interval rule matrix model for automated decision making and reasoning about possible courses of actions based on an environmental observation.

6.2 The Rule Matrix Model

6.2.1 A Simple Example

Decisions are mostly made according to observations of the current environment and existing knowledge. Here is a simple example.

Example 1. Bob needs to decide if he should carry his umbrella and coat before leaving home for work. He may carry (a) both his umbrella and coat, (b) his umbrella but not his coat, (c) his coat but not an umbrella, and (d) neither. Unfortunately, he has no way to check the forecasts and must make a quick decision by observing the current weather conditions. Based on his knowledge, Bob uses the following table to make his decision.

Environment/decision	a	b	c	d
Rain or very likely	Yes	Yes	No	No
Current temperature	Below 40^{o}F	Above 40^{o}F	Below 40^{o}F	Above 40^{o}F

The first row of the table lists all possible decisions for Bob and the first column lists environmental parameters that Bob takes into consideration for decision making. By matching the environmental observations with the table, Bob can easily make his decision.

6.2.2 Rule Matrices

In general, an environment e may contain m features, and there can be n possible different decisions, $d_1, d_2, \ldots, d_n$, that could be made based on the presence of the environment features. Let $e = (e_1, e_2, \ldots, e_m)^T$ be an observation of the environment; that is, e denotes the degree to which certain features of an environment are present. Then a knowledge-based agent may select a specific decision by matching the input e with column vectors of the m by n matrix P:

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ & \vdots & & \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix}.$$

If an observation vector e matches P_j, (i. e., the j-th column of P), then the j^{th} decision d_j should be selected. A benefit function can be associated with the decision as well. We call P a *rule matrix* for the decision-making process, and the model is called a rule matrix model.

In practice, a rule matrix should be interval-valued, as suggested by de Korvin, Hu, and Chen in [4]. This is mainly because of the following two reasons. The first one is that a decision is usually selected according to ranges of attribute values rather than matching a point exactly. In addition, an environment observation often imprecise. A rule matrix generated from such imperfect data should allow an error bounds. By specifying the lower and upper bounds of feature presence in an interval rule matrix, we take uncertainty into consideration appropriately [3, 4, 14].

We denote an interval rule matrix by the boldface uppercase letter $\boldsymbol{P}$ whose entry $\boldsymbol{p}_{ij}$ is an interval for each $1 \leq i \leq m$ and $1 \leq j \leq n$. By the same token, an environment observation is also interval-valued and is denoted by boldface as $\boldsymbol{e}$.

6.2.3 Another Example

For a more sophisticated example, we describe an interval rule matrix to enhance network intrusion detection systems (IDSs). In addition to user authentication, cryptography, and firewalls, IDSs are embedded into network management systems to monitor network states and perform control functions.

Two main types of intrusion detection approach are *misuse detection* [8] and *anomaly detection* [13]. Anomaly detection compares activities with established normal usage patterns (profiles) and determines if the network state deviates over some threshold from normal patterns. This approach is recognized as being capable of catching new attacks. However, normal patterns of usage and system behaviors can vary wildly. Therefore, anomaly detection usually produces a very large number of false alarms, see [2] and [10] and hence is impractical. By representing normal patterns in an interval-valued matrix, we should be able to reduce false alarms caused by a small change in normal behavior or a slight deviation from a pattern derived from the network audit data.

To generate an interval rule matrix for intrusion detection, we may assume the availability of previous network state data and intrusions associated with them. These can be automatically collected by network monitoring software. If we consider m features and n possible interval-valued network states, then we have an $m \times n$ interval rule matrix. We represent the likelihood of an intrusion with 0 (certainly not) and 1 (absolutely yes). Then the likelihood of an intrusion can be represented as a mapping from the network state to the interval $[0, 1]$.

To simplify the example, in [6], Duan, Hu, and Wei considered only three network state attributes: the average packet delay for a Transmission Control Protocol (TCP) flow, the bandwidth utilized for the TCP flow, and the number of TCP flows arriving at one router ingress port (shown as the first, second, and the third rows of the following matrix, respectively). An artificial interval matrix for the network state can be as follows:

$$\begin{pmatrix} [0.8, 1.8] & [1.6, 2.1] & [2.2, 3.1] \\ [3.5, 5.2] & [4.8, 6.5] & [6.9, 7.4] \\ [8, 10] & [9, 12] & [13, 16] \end{pmatrix}.$$

By counting intrusions associated with each of these states, we can obtain an empirical probability of intrusions d_j associated with the j-th column. For a current network state observation $\boldsymbol{e}$, we can estimate the likelihood of a intrusion (or intrusions) as follows:

- *Input:* Current network state $\boldsymbol{e}$.
- *Estimation:* Compare $\boldsymbol{e}$ and column vectors of $\boldsymbol{P}$.
  Case 1: If $\boldsymbol{e} \subseteq \boldsymbol{P}_j$, the likelihood of an intrusion is d_j.
  Case 2: If $\forall j \in \{1, 2, \ldots, n\}$, $\boldsymbol{e} \cap \boldsymbol{P}_j = \emptyset$, then the network state is very abnormal when compared to historical states. An intrusion alarm should be sent to the network administrator.
  Case 3: If $\boldsymbol{e} \cap \boldsymbol{P}_j \neq \emptyset$ for more than one $j \in \{1, 2, \ldots, n\}$, then one may select the greatest d_j or use other heuristics.

By using the empirical probability and cost function, the system can estimate the expected average damage for a possible intrusion. In [6], Duan, Hu, and Wei further suggested that the interval rule matrix model could be integrated with data collection, policy generation, and policy application. According to the current likelihood of network intrusion, network control can then identify risk levels and automatically take actions.

The example constructed earlier shows the potential of interval rule matrix in various kinds of application. In the rest of this chapter, we discuss ways to establish an interval rule matrix and to how apply it to decision making.

6.3 Establishing an Interval-Valued Rule Matrix

The main purpose of this section is to design practical algorithms that construct an interval rule matrix $\boldsymbol{P}$ from a known dataset E. We assume that E contains N environment-decision pairs. Because we have used e_k to indicate the k^{th} feature of an environment, we use a superscript e^k to denote the k^{th} observation of the environment. By the term environment-decision pair $[e^k, d_{k^*}] \in E$, we mean that under a given environment e^k, $1 \leq k \leq N$, the *desired* decision should be d_{k^*} for a particular k^* with $1 \leq k^* \leq n$.

A naive way to determine the j^{th} column of P, P_j, is to let $P_j = e^k$ if $j = k^*$. This certainly ensures that the j^{th} decision will be selected if the environment is e^k. However, this will not work appropriately, since the same decision d_j may be taken for different environment observations. In fact, n is usually much less than N. By using an interval rule matrix like the one constructed in the previous section, one may come to the same decision even with different values of environment observations.

6.3.1 A Straightforward Approach

Our objective is to extract an interval-valued rule matrix $\mathbf{P} = \{\boldsymbol{p}_{ij}\}_{m\times n} = \{[\underline{p}_{ij}, \overline{p}_{ij}]\}_{m\times n}$ from a dataset $\boldsymbol{E} = \{(\boldsymbol{e}^k, d_{k^*}) | 1 \le k \le N\}$ without any previous knowledge except the training dataset itself. A straightforward approach would be as follows.

For any given $j \in \{1, 2, \ldots, n\}$, we find an interval vector $\boldsymbol{P}_j$ such that $\forall k \in \{1, 2, \ldots, N\}$, $\boldsymbol{e}^k \subset \boldsymbol{P}_j$ if $d_{k^*} = j$. It is ideal if $\boldsymbol{P}_j \cap \boldsymbol{P}_k = \emptyset$ whenever $j \ne k$. Then these mutually exclusive n column interval vectors form an interval rule matrix that fits the training dataset. In the following example, we presorted the training dataset according to the decisions. By taking a hull for each feature corresponding to the same decisions, we easily obtain an interval rule matrix.

Example 2. An environment consists of three features, and there are three possible decisions: a, b, and c. Construct an interval rule matrix from the following collection of desired environment-decision pairs:

```
([0.8, 1.1] [1.1, 1.2] [0.8, 1.7], a);
([0.7, 0.9] [1.3, 1.4] [0.9, 1.1], a);
([0.8, 1.0] [1.3, 1.5] [1.0, 1.8], a);
([0.7, 0.9] [1.2, 1.4] [0.9, 1.0], a);
([0.8, 0.9] [1.2, 1.4] [0.9, 1.0], a).
([0.1, 0.4] [2.0, 2.1] [0.2, 0.5], b);
([0.2, 0.3] [2.0, 2.3] [0.2, 0.4], b);
([0.1, 0.4] [2.0, 2.1] [0.2, 0.5], b);
([0.3, 0.4] [2.1, 2.2] [0.3, 0.4], b);
([0.0, 0.2] [2.2, 2.5] [0.4, 0.5], b);
([1.5, 3.3] [0.7, 0.9] [5.8, 6.5], c);
([1.6, 3.0] [0.7, 1.0] [4.8, 5.0], c);
([2.0, 3.2] [0.9, 1.0] [5.1, 6.3], c);
([3.2, 4.0] [0.4, 1.0] [6.2, 6.4], c);
([1.5, 4.0] [0.5, 1.0] [4.8, 6.0], c);
```

Let us consider the interval vector $\boldsymbol{P}_a$ first. The first element of the interval vector $\boldsymbol{P}_a$ should contain the intervals [0.8, 1.1], [0.7, 0.9], [0.8, 1.0], [0.7, 0.9], and [0.8, 0.9]. Hence, the union of these five intervals, [0.7, 1.1], works. Similarly, we get the second and the third elements of $\boldsymbol{P}_a$ as [1.1, 1.5] and [0.8, 1.8]. Therefore,

$$\boldsymbol{P}_a = \begin{pmatrix} [0.7,\ 1.1] \\ [1.1,\ 1.5] \\ [0.8,\ 1.8] \end{pmatrix}.$$

We can find $\boldsymbol{P}_b$ and $\boldsymbol{P}_c$ similarly and then construct an interval rule matrix as

$$\begin{pmatrix} [0.7,\ 1.1] & [0.0,\ 0.4] & [1.5,\ 4.0] \\ [1.1,\ 1.5] & [2.0,\ 2.5] & [0.4,\ 1.0] \\ [0.8,\ 1.8] & [0.2,\ 0.5] & [4.8,\ 6.5] \end{pmatrix}$$

From the above example we can see that the rule matrix can be updated easily. Whenever a new environment-decision pair becomes available, a union operation on interval vectors takes care of the updating. Let $([0.9, 1.7], [1.2, 1.6], [1.0, 1.5]; a)$ be a newly available pair. Then the first column of the above rule matrix can be adjusted as

$$\begin{pmatrix} [0.7,\ 1.1] \\ [1.1,\ 1.5] \\ [0.8,\ 1.8] \end{pmatrix} \cup \begin{pmatrix} [0.9,\ 1.7] \\ [1.2,\ 1.6] \\ [1.0,\ 1.5] \end{pmatrix} = \begin{pmatrix} [0.7,\ 1.7] \\ [1.1,\ 1.6] \\ [0.8,\ 1.8] \end{pmatrix}.$$

This updating scheme also suggests a way to construct an interval rule matrix starting from an empty rule matrix (i.e., from an initial rule matrix in which every element is an empty interval).

Let us write the straightforward approach as an algorithm that constructs an $m \times n$ interval rule matrix with a known dataset that contains N environment-decision pairs.

Algorithm 3 (Straightforward construction of an interval rule matrix)

- *Initialize an empty $m \times n$ rule matrix $\boldsymbol{P}$ such that $\boldsymbol{p}_{ij} = \emptyset$, $\forall i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.*
- *For each environment-decision pair $(\boldsymbol{e}^k, d_{k^*})$, where $k \in \{1, 2, \ldots, N\}$ and $k^* \in \{1, 2, \ldots, n\}$, update the rule matrix as follows:*

$$\boldsymbol{p}_{ik^*} \leftarrow \mathit{hull}(\boldsymbol{p}_{ik^*}, \boldsymbol{e}_i^k) \quad \text{for } i = 1, 2, \ldots, m$$

where

$$\mathit{hull}([\underline{x}, \overline{x}], [\underline{y}, \overline{y}]) = [\min\{\underline{x}, \underline{y}\}, \max\{\overline{x}, \overline{y}\}]$$

is the smallest interval containing both $\boldsymbol{x} = [\underline{x}, \overline{x}]$ and $\boldsymbol{y} = [\underline{y}, \overline{y}]$.

For each environment-decision pair, one column of the rule matrix needs to be updated. Therefore, Algorithm 3 is $\mathcal{O}(mN)$. As we can see in this algorithm, it is unnecessary to presort a training dataset.

The straightforward approach is simple enough, with a relatively low computational cost. However, it may not always result in mutually exclusive columns. We say that an interval rule matrix is "fat" if two or more columns have a nonempty intersection. For example, let us assume that a new data item $([0.3, 0.8][1.4, 2.1][0.4, 0.9]; b)$ becomes available for updating the above rule matrix. It would result in the second column of the rule matrix being

$$\begin{pmatrix} [0.0,\ 0.4] \\ [2.0,\ 2.5] \\ [0.2,\ 0.5] \end{pmatrix} \cup \begin{pmatrix} [0.3,\ 0.8] \\ [1.4,\ 2.1] \\ [0.4,\ 0.9] \end{pmatrix} = \begin{pmatrix} [0.0,\ 0.8] \\ [1.4,\ 2.5] \\ [0.2\ 0.9] \end{pmatrix}.$$

The intersection of the updated column with the original first column is nonempty since

$$\begin{pmatrix} [0.7,\ 1.7] \\ [1.1,\ 1.6] \\ [0.8,\ 1.5] \end{pmatrix} \cap \begin{pmatrix} [0.0,\ 0.8] \\ [1.4,\ 2.5] \\ [0.2\ 0.9] \end{pmatrix} = \begin{pmatrix} [0.7,\ 0.8] \\ [1.4,\ 1.6] \\ [0.8\ 0.9] \end{pmatrix} \neq \emptyset.$$

If an observation falls into the intersection, say

$$\boldsymbol{e} = \begin{pmatrix} [0.72,\ 0.75] \\ [1.4,\ 1.5] \\ [0.83\ 0.89] \end{pmatrix},$$

then it is unclear whether one should select a or b as the decision. For this reason, we need to study more sophisticated algorithms.

6.3.2 A Divide-and-Conquer Approach

We now present an alternative approach, with the divide-and-conquer approach, to establish an interval rule matrix $\boldsymbol{P}$ from a given training dataset. In real applications, observed environment features at a particular time are mostly thin intervals. Instead of starting with decisions as in the straightforward approach, let us begin with the consideration of feature parameters.

To avoid "fat" interval rule matrices, we first subdivide the range of each feature parameter into narrow subintervals. Then by picking subintervals from each of the m-features, we construct an m-dimensional tube (interval vectors whose component intervals are narrow). Thus, if the interval for the i-th feature is subdivided into j_i subintervals for each i, $1 \leq i \leq m$, the total number of tubes will be $\prod_{i=1}^{m} j_i$.

For each environment-decision pair in the training dataset, we try to fit the pair with one or more tubes according to its environment value. If a pair can be put inside a tube completely, we increment the frequency of the decision(s) associated with the tube. If the environment of a pair covers several adjacent tubes, we increment the frequency of the decision for each tube that the environment covers. After doing so for all pairs in E, we have a decision frequency list associated with each tube. (Some of the frequencies can be zero.) If these tubes are narrow enough, then each of them may contain only a single decision, say d_j.

In the conquer stage, we select one of the tubes with the highest frequency for d_j as the base for $\boldsymbol{P}_j$, the j-th column of the rule matrix. Adjacent tubes associated with the same decision d_j can then be combined to form the j-th column of the rule matrix. On the other hand, if a tube contains different decisions, it can be subdivided further. Of course, the frequency should be recounted in the latter case.

The conquer stage not only consists of combining those adjacent tubes associated with the same decision but also of dealing with tubes that are not

associated with any decisions at all. In addition, a decision may be associated with nonadjacent columns. Those tubes that are not associated with any decisions we call *empty tubes*. These empty tubes can be either removed or kept inactive for possible future use. (Using a predetermined frequency threshold, one may computationally filter out statistically insignificant tubes as empty.)

A decision may be associated with multiple disconnected tubes (i.e., the tubes do not share a common boundary). (Geometrically, there are other tubes between tubes associated with the same decision.) A separation of two disjoint tubes with the same decision is *removable* if the tubes in between are empty. By taking a hull, one may remove such removable separations.

However, there are nonremovable separations, which means that there exists a tube between them but associated with another decision. In such a case, we have to associate the decision with multiple columns. However, this will cause a contradiction with the assumption that the rule matrix has only n columns. To resolve this, we can use multiple rule matrices, such that each rule matrix is on a separate "page." In this way, the rule matrix is no longer a two-dimensional array. The column of a rule matrix can be viewed as a pointer that points to multiple interval vectors on different pages. Obviously, the number of pages should be the same as the maximum number of disjoint tubes that are associated with a single decision.

Let us summarize the above idea in an algorithmic format.

Algorithm 4 (Divide and conquer scheme to construct an interval rule matrix)

1. *Divide*
   a) *Initialization: Subdivide each domain of the m-features into a predetermined number of subintervals to form a set of tubes T. For each $t \in T$, associate a frequency zero with it for each possible decision.*
   b) *For each $(\boldsymbol{e}^k, d_k^*) \in E$, if $\boldsymbol{e}^k \cap t \neq \emptyset$ for some $t \in T$ and there are no decisions other than d_k^* associated with t, add 1 to the frequency count of d_k^* on t. Otherwise, if there are decisions other than d_k^* associated with t, then subdivide t for decision separation.*
2. *Conquer*
   a) *For those tubes with the same decision, combine them if they are adjacent or the separations are removable.*
   b) *Form the j-th column of the rule matrix with the tube associated with the decision d_j. Note: There can be multiple separated columns associated with the same decision d_j.*

Notice that the frequency count can be used as an indicator for the strength of a rule. By ignoring very low frequencies, it can be used to avoid tube refinements that are too detailed. To control the output rule matrix with Algorithm 4, one may apply (a) a predefined tube size, (b) a frequency cutoff, and (c) multiple pages. By associating a decision with disconnected columns

on multiple pages, the "fat" rule matrix problem in Algorithm 3 may be eliminated.

The computational complexity of Algorithm 4 can be much higher than that of Algorithm 3. For an m-feature environment, if one performs s subdivisions for each feature, then T consists of s^m tubes. Each matching requires $m \log s$ comparisons with binary searching. Even without the tube refinement, the algorithm requires $\mathcal{O}(Nm \log s)$ comparisons. With the refinement, say performing up to h bisections for each feature, the complexity will be $\mathcal{O}(Nmh^m \log s)$. In real applications, the number of features of a complex environment can be very large.

A reasonable approach to make Algorithm 4 more practical is to control the number of features under consideration. Feature selection itself is a computational decision-making problem [7]. To control the number of features, one may rank features according to their correlations with respect to decisions. By selecting subsets of features based on reliably assessing the statistical significance of the relevance of features to a given predictor, one may build more compact feature subsets. Also, applying a singular value decomposition (SVD), one can form a set of features that are linear combinations of the original variables, which provide the best possible reconstruction of the original data in the least squares sense.

In practice, instead of using all of the data in a training dataset to select features, one may first sample the training dataset with Algorithm 4 to form an initial rule matrix. With the initial rule matrix, features can then be selected. Applying Algorithm 4 with these selected features, one can adaptively update the rule matrix. Hopefully, these selected features are good enough for most data, and only a few environment-decision pairs need to have more features than those in the selected sample when updating the rule matrix. Some heuristics such as sample selection, feature granularity, and cutoff threshold are needed in applications.

The above practical approach also has other advantages. It allows one to make a decision with the current rule matrix without completing the training. The correctness of the decision selected can be used as a feedback to adjust the rule matrix. We call this capability the ability to do "online" dynamical training. The selected features can also be updated adaptively. Rules and significance of a feature for decision making can be changed from time to time. An adaptive approach can update them as well. Also, we should point out that domain knowledge can and should play a very significant role in rule matrix generation and feature selection.

6.4 Decision Making with an Interval Rule Matrix

The purpose of establishing an interval rule matrix is to apply it to making decisions. Since a multipage rule matrix can be processed page by page, we consider only single-page interval rule matrices here.

Making a decision based on an environment observation $\mathbf{e}$ according to an interval rule matrix requires to determine a j such that $\boldsymbol{e}$ "matches" the j-th column of the rule matrix. By the word "match", we mean that the two interval vectors should somehow be "close." To determine the closeness of two interval vectors, we first need to define the distance between two intervals.

Definition 1. *Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two intervals. The distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as*

$$\operatorname{dist}(\boldsymbol{a},\boldsymbol{b}) = \begin{cases} 0 & \text{if } \boldsymbol{a} \subseteq \boldsymbol{b} \text{ or } \boldsymbol{b} \subseteq \boldsymbol{a} \\ \min\{|a-b|+1;\ \forall a \in \boldsymbol{a}, b \in \boldsymbol{b}\} & \text{if } \boldsymbol{a} \cap \boldsymbol{b} = \emptyset \\ 1 - \dfrac{w(\boldsymbol{a} \cap \boldsymbol{b})}{\min\{w(\boldsymbol{a}), w(\boldsymbol{b})\}} & \text{otherwise.} \end{cases}$$

The above definition implies the following properties:

1. The distance is reflexive: The distance from an interval $\boldsymbol{a}$ to another interval $\boldsymbol{b}$ is the same as from $\boldsymbol{b}$ to $\boldsymbol{a}$.
2. The distance between two intervals is zero if one is a subset of the other.
3. The minimum distance between two disjoint intervals is greater than or equal to 1.
4. The distance between two partially overlapped intervals is between 0 and 1.
5. The distance between two disjoint intervals is in fact the same as

$$|m(\boldsymbol{a}) - m(\boldsymbol{b})| - \frac{w(\boldsymbol{a}) + w(\boldsymbol{b})}{2} + 1,$$

where $m()$ and $w()$ are midpoint and width functions, respectively.

The proof is straightforward.

Note that Definition 1 is appropriate for knowledge processing, but it differs from other commonly used measures of distance, such as the Hausdorff distance [1, p. 11], used within the interval mathematics community.

Applying the concept of distance between intervals, we can define the distance between two interval vectors as follows.

Definition 2. *Let $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m)$ and $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m)$ be two m-dimensional interval vectors. Then the l-distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as*

$$\operatorname{dist}_l(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{1 \le i \le m} \operatorname{dist}^l(\boldsymbol{x}_i, \boldsymbol{y}_i) \right)^{\frac{1}{l}}.$$

By computing a distance, say $l = 1$ or $l = \infty$, between an environment observation vector $\boldsymbol{e}$ and each column of an interval rule matrix $\boldsymbol{P}$, one can then make a decision accordingly. It seems reasonable to select the d_j such that $\operatorname{dist}(\boldsymbol{e}, \boldsymbol{P}_j) = \min_{1 \le i \le n} \operatorname{dist}(\boldsymbol{e}, \boldsymbol{P}_i)$, where $\boldsymbol{P}_i$ is the i-th column of $\boldsymbol{P}$. This is

because the distance between $\boldsymbol{e}$ and $\boldsymbol{P}_i$ is viewed as the strength indicator of the i-th decision. The smaller the distance is, the stronger the decision should be. In other words, the larger the distance is, the less the decision should be selected. Zero distance indicates the strongest match between two interval vectors.

It is possible that an observation vector may have the same (or almost equal) minimum distance to multiple columns of the rule matrix. This can happen even when the observation vector is a thin interval vector. To make a reasonable decision from multiple best matches, instead of a random pick, we define the expected value of a decision as follows.

Definition 3. *Let ρ_j and v_j be the probability and the benefit value of a decision j, respectively. Then, the expected value of the decision j is defined as $exp_j = \rho_j v_j$.*

In constructing an interval rule matrix from a training dataset, the decision frequencies have been recorded. They can be considered as empirical probabilities for each of the decisions. Also, historical data can provide the average return on a decision. Therefore, one may select the decision with the highest expected values among the multimatched columns.

Instead of applying the concept of interval distance, an earlier alternative approach is to select a decision based on an interval rule matrix with fuzzy logic and values of possibility and necessity functions. Readers may refer to [4] and Chapter 2 of this book for more information.

6.5 Conclusions

We have studied an interval rule matrix model for establishing decision-making systems. Using a training dataset consisting of environment-decision pairs, we have proposed two algorithms to generate an interval rule matrix. The straightforward approach has a time complexity of $\mathcal{O}(mN)$. The divide-and-conquer approach may use adaptive modification, "online" training, and feature selection for more practicality.

With an interval rule matrix, making a decision for an environment observation becomes finding the minimum distance between interval vectors. Interval rule matrices have potential applications in rule-based automated decision-making systems.

Acknowledgment: This work is partially supported by the U.S. National Science Foundation under grants CISE/CCF-0202042 and CISE/CCF-0727798.

References

1. Alefeld, G., Herzberger, J.: Introduction to Interval Computations. Academic Press Inc., New York (1983). Translation by J. Rokne from the original German "Einführung In Die Intervallrechnung"
2. Bace, R., Mell, P.: NIST special publication on intrusion detection system. Technical report, NIST (National Institute of Standards and Technology) (2001). http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf.
3. Berleant, D., Cheong, M.P., Chu, C.C.N., Guan, Y., Kamal, A., Sheble, G., Ferson, S., Peters, J.F.: Dependable handling of uncertainty. Reliable Computing 9(6), 407–418 (2003)
4. de Korvin, A., Hu, C., Chen, P.: Generating and applying rules for interval valued fuzzy observations. In: Z.R. Yang, R.M. Everson, H. Yin (eds.) Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science, Vol. 3177, pp. 279–284. Springer-Verlag, Heidelberg (2004)
5. de Korvin, A., Hu, C., Sirisaengtaksin, O.: On firing rules of fuzzy sets of type II. Applied Mathematics 3, 151-159 (2000)
6. Duan, Q., Hu, C., Wei, H.C.: Enhancing network intrusion detection systems with interval methods. In: Proceedings of the ACM Symposium on Applied Computing, pp. 1444–1448 (2005)
7. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Machine Learning Research 3, 1157–1182 (2003)
8. Ilgun, K., Kemmerer, R.A., Porras, P.A.: State transition analysis: A rule-based intrusion detection approach. Software Engineering 21(3), 181–199 (1995)
9. Jang, J.S.R.: ANFIS: Adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics 23, 665–684 (1993)
10. Julisch, K.: Clustering intrusion detection alarms to support root cause analysis. ACM Transactions on Information and System Securroty 6(4), 443–471 (2003)
11. Mendel, J.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Prentice-Hall, Upper Saddle River, NJ (2001)
12. Pawlak, Z.: Rough sets and fuzzy sets. Fuzzy Sets and Systems 17, 99–102 (1985)
13. Seleznyov, A., Puuronen, S.: Anomaly intrusion detection systems: Handling temporal relations between events. In: Proceedings of Recent Advances in Intrusion Detection, West Lafayette, IN (1999). http://www.raid-symposium.org/raid99/PAPERS/Seleznyov.pdf
14. Shary, S.P.: A new technique in systems analysis under interval uncertainty and ambiguity. Reliable Computing 8(5), 321–418 (2002)
15. Zadeh, L.A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man, and Cybernetics SMC-3, 28–44 (1973)