On the Parallel Spaces of Knowledge and Experience Based on the Concept of "Dark-matter"

Xing Chen,[1] Yasushi Kiyoki[2]

[1] Kanagawa Institute of Technology, Department of Information & Computer Sciences, Kanagawa, Japan
chen@ic.kanagawa-it.ac.jp
[2] Keio University, Graduate School of Media and Governance, Tokyo, Japan
kiyoki@sfc.keio.ac.jp

This paper explores the phenomenon of inapplicability of experience, which occurs when we make mistakes by applying past experiences to current problems. The paper aims to create a knowledge model to analyze this phenomenon based on the concept of "dark-matter," which represents time-related data. This model uses two-dimensional matrixes to represent both time-related and non-time-related data. In this paper, the authors propose a concept of "parallel spaces" for the expression and processing of knowledge based on the concept of "dark-matter." Case studies are used to illustrate how knowledge is generated and expressed in this model, including examples of the phenomenon of inapplicability of experience. The contribution of this paper is the presentation of a new concept of "parallel spaces" based on the concept of "dark-matter," and the exploration of the relationship between the parallel spaces and knowledge and experience. Based on the case studies, the reason for the phenomenon of inapplicability of experience is revealed, providing insight into how we can more effectively use our past experiences to solve current problems.

Keywords:
knowledge, knowledge presentation, knowledge generation, machine learning, semantic space, spatiotemporal space

University of Maribor Press

DOI https://doi.org/10.18690/um.feri.5.2023.17
ISBN 978-961-286-745-4

1 Introduction

People make judgments and decisions based on experience and knowledge to solve problems. However, it often happens that wrong judgments and decisions are made based on experience and knowledge. For example, in the stock trading process, people will determine the current trade based on the stock's past ups and downs and trading experience. However, decisions based on experience are not always correct. After you decide to buy, the stock may fall, or after you decide to sell, the stock may rise. This phenomenon is referred to as "the phenomenon of inapplicability of experiences" in this paper. The aim of this paper is to create a knowledge model to analyze this phenomenon. It is known that knowledge and experience build up over time. In other words, if we want to express knowledge and experience in computers, we need to create a time related knowledge model. In our previous research, we proposed a knowledge model based on a concept of "dark-matter" [1].

The concept of "dark-matter" stems from existing research on semantic computing models [2-5]. These models use semantic spaces to represent the meaning of data. By mapping data into target semantic spaces and presenting them as points, these models calculate Euclidean distances between them to perform semantic calculations. For instance, to conduct semantic queries, query data or keywords are mapped to a semantic space and summarized as a point. The same applies to retrieval candidate data, before computing Euclidean distance between the point of the query and the points of the retrieval candidates to determine which relative retrieval candidate should be a retrieval result.

In our previous researches, two methods, Mathematical Model of Meaning (MMM) [4, 5], and Semantic Feature Extracting Model (SFEM) [2, 3] are used in creating semantic spaces. In MMM, a semantic dataset such as the English-English dictionary is used to create the semantic space. On the other hand, SFEM creates semantic spaces based on defined data sets relevant to specific applications. Mapping matrixes are required to map input data to the semantic space. SFEM mapping matrixes are defined according to the model's application [6, 7]. Various techniques have been developed to construct mapping matrixes of MMM for semantic information retrieval, classification, extraction, and analysis on cause and effect [8-14]. Moreover, a technique has been developed to construct the mapping matrixes through deep-learning [15]. In MMM and SFEM, the elements of the matrixes that represent semantic spaces are predefined, which distinguishes them from other models, such as the artificial neural network model and deep-learning artificial neural network model [16-19].

A mechanism based on the semantic space model is developed to implement basic logic computation to determine true and false judgments [20], which is a fundamental mechanism required for machine learning. This mechanism is applied to simulate unmanned ground vehicle control [21]. Case studies are used to illustrate why the phenomenon of inapplicability occurred. A model is presented for temporal data processing, with the word "matter" used to represent non-temporal elements and "dark-matter" for temporally changing elements [1]. An exploratory research is presented on knowledge expression and generation processes based on the concept of "dark-matter" [22].

This research creates a new knowledge model to analyze the phenomenon of inapplicability and introduces the concept of "parallel spaces" based on previous studies. The concept of "dark-matter" which is used in the analysis of the phenomenon of inapplicability is discussed in Section 2. Section 3 describes the relationship between knowledge and dark-matter, and Section 4 analyzes the phenomenon of inapplicability, and provides case studies to illustrate the issue. In section 4, the paper goes on to present the concept of "parallel spaces" and its applications in examples before concluding the study.

2 The machine learning model created based on the concept "dark-matter"

This section offers a brief review of a machine learning model known as the dark-matter learning model, which is based on the concept of dark-matter. In this model, matrix multiplication is referred to as mapping or space mapping. If a two-dimensional matrix **X** represents a space, matrix **X** can be decomposed into matter and dark-matter matrix. The former refers to the first column of **X** that correlates with sensor data and is referred to as visible data, while the latter refers to the matrix's second to the last column that is randomly filled and defined as chaotic space. Figure 1 shows an example of the space matrix **X**. According to the paper [1], knowledge expression and knowledge generation are linked to dark-matter.

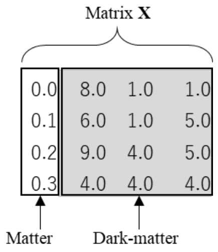


Figure 1: **An example of a space matrix X**
Source: own.

The learning process in the dark-matter learning model is concerned with transforming chaotic space into ordered space. The learning model resembles a state machine where states transition from one to another. A state transition diagram is employed to illustrate this process. Antimatter space is defined as the inverse of the matrix **X**, and it is used to create a new matrix **C** from **E** that represents actions taken by agents. Mass and energy equivalent equation explain that mass is visible and correlates with sensor data and refers to the amount of matter an object contains. Energy is represented by a vector **E** and is measurable. Equation (1) shows the relationship of these three matrixes. Figure 1 shows an example of matrix **X**.

$$\mathbf{E} = \mathbf{X}{*}\mathbf{C} \tag{1}$$

As mentioned in the paper [22], the creation and development of knowledge is linked with what is called "dark-matter." A "dark-matter learning model" grounded on this concept is also presented in the paper. The goal of the learning process is to shift from a chaotic space to an ordered space wherein the learning model is viewed as a state machine. The state machine's state changes from one to another, referred to as state transition, as illustrated by state transition diagrams such as Figure 2. In Figure 2, A circle and a number identify a state, while an arrow signifies the state's transition from one to the next. Numbers beside arrows indicate the condition needed for the transition, like for state "0.0" to transition to state "0.1" when the input data is "0.0."

Figure 3 displays the creation of an "ordered space" from the original "chaotic space" through state transitions using the example of the state transition diagram in Figure 2. The start state is "0.0" and the first step is the transition from "0.0" to "0.1". The chaotic space is shown in Figure 3 (a) and the first step is shown in Figure

3 (b), with the transition from "0.1" to "0.3" shown in Figure 3 (c). The full state transition diagram is represented in Figure 3 (d), creating an "ordered space" from the original "chaotic space." The "matter" refers to the first column of the matrix and the grey-painted second, third, and fourth columns represent the "dark-matter" matrix.

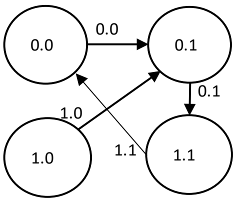
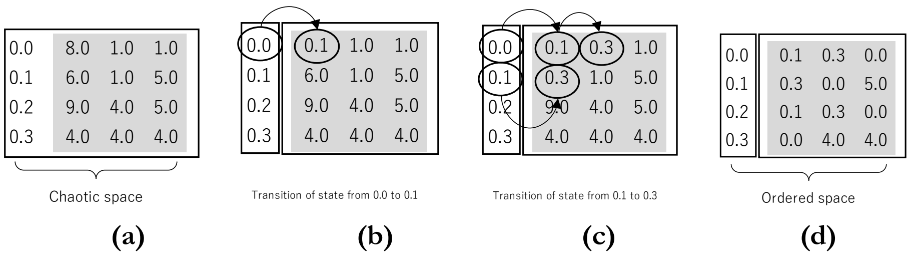


Figure 2: An example of state transition
Source: own.


Chaotic space
Transition of state from 0.0 to 0.1
Transition of state from 0.1 to 0.3
Ordered space
(a)
(b)
(c)
(d)

0.0	8.0	1.0	1.0
0.1	6.0	1.0	5.0
0.2	9.0	4.0	5.0
0.3	4.0	4.0	4.0

0.0	0.1	1.0	1.0
0.1	6.0	1.0	5.0
0.2	9.0	4.0	5.0
0.3	4.0	4.0	4.0

0.0	0.1	0.3	1.0
0.1	0.3	1.0	5.0
0.2	9.0	4.0	5.0
0.3	4.0	4.0	4.0

0.0	0.1	0.3	0.0
0.1	0.3	0.0	5.0
0.2	0.1	0.3	0.0
0.3	0.0	4.0	4.0

Figure 3: **Creating an "ordered space" from a "chaotic space"**
Source: own.

If $\mathbf{X}^{-1}$ is an inverse matrix of the matrix **X**, the matrix $\mathbf{X}^{-1}$ is referred to as an "***antimatter space***." By applying antimatter space to a matrix **E**, which represents actions of agents, a new matrix **C** is created as shown in equation (2). The calculation presented by the equation (2) is referred to as "*learning*" or "*training*" calculation.

$$\mathbf{C} = \mathbf{X}^{-1} * \mathbf{E} \tag{2}$$

In physics, "*mass*" is the measure of the amount of matter an object contains and can be sensed by sensors. "*Matter*" is defined as a vector that correlates with sensor data and is similar to the concept of "*mass*". Energy in physics is a vector **E**, and the elements of the vector **E** and the mass in matrix **X** are measurable. Thus, the measurable values of **E** correspond to energy in physics and the measurable values of mass in physics correspond to "matter".

To analyze the relationship between equation (1) and the "energy mass equivalent equation", we will take an example. Suppose that space **X** is a five-by-five matrix, as shown in Figure 4 (a). This means that there are five different types of matter in space. The elements in the first column of the matrix are the mass of matter. In the example, the values of the "matter" and "dark-matter", which are the elements of the matrix **X**, are shown in Figure 4 (a). The elements of the vector **E** are shown in Figure 4 (b). The inverse matrix of **X**, $\mathbf{X}^{-1}$, is shown in Figure 4 (c).

$$\begin{pmatrix} 6.00 & 14.16 & 72.84 & 49.42 & 26.72 \\ 3.00 & 83.37 & 75.00 & 36.21 & 28.21 \\ 9.00 & 27.55 & 32.70 & 59.96 & 20.56 \\ 2.00 & 28.83 & 34.27 & 35.39 & 74.98 \\ 7.00 & 55.43 & 86.68 & 38.21 & 87.91 \end{pmatrix} \quad \begin{pmatrix} 600.00 \\ 300.00 \\ 900.00 \\ 200.00 \\ 700.00 \end{pmatrix} \quad \begin{pmatrix} -0.09 & -0.07 & 0.11 & -0.15 & 0.16 \\ -0.02 & 0.01 & 0.01 & 0.00 & 0.00 \\ 0.02 & 0.00 & -0.01 & -0.01 & 0.00 \\ 0.01 & 0.01 & 0.01 & 0.02 & -0.03 \\ -0.01 & -0.01 & 0.00 & 0.01 & 0.01 \end{pmatrix}$$

mass dark-matter $\quad$ **E** $\quad$ $\mathbf{X}^{-1}$

(a) Space X: created with five types of matter **(b) Energy vector** **(c) Inverse matrix of X**

Figure 4: **An example of a space with five different types of matter**
Source: own.

Another matrix, vector **C** is also used to analyze the relationship between the energy and "mass equivalent equation". The values of the element of the vector **C** are calculated based on equation (2), multiplying $\mathbf{X}^{-1}$ by **E**. if the value of the first element of vector **C** is a non-zero value, and all other elements in vector **C** are zero, as that shown in Figure 5, the elements of vector **E** can be calculated by multiplying vector mass and only the first value of vector **C**, that is the dark-matter is not utilized in this computation process, equation $e=mc^2$ is appropriate for this scenario where the first value of vector C is presented as c^2.

$$\underset{\mathbf{C}}{\begin{pmatrix} 100.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{pmatrix}} = \underset{\mathbf{X}^{-1}}{\begin{pmatrix} -0.09 & -0.07 & 0.11 & -0.15 & 0.16 \\ -0.02 & 0.01 & 0.01 & 0.00 & 0.00 \\ 0.02 & 0.00 & -0.01 & -0.01 & 0.00 \\ 0.01 & 0.01 & 0.01 & 0.02 & -0.03 \\ -0.01 & -0.01 & 0.00 & 0.01 & 0.01 \end{pmatrix}} * \underset{\mathbf{E}}{\begin{pmatrix} 600.00 \\ 300.00 \\ 900.00 \\ 200.00 \\ 700.00 \end{pmatrix}}$$

Figure 5: Multiplying $\mathbf{X}^{-1}$ by E to calculate the rule vector C
Source: own.

The vector **E** can be calculated from **C** using equation (1), as shown in Figure 6 (a). Let c^2 represent the first value of **C**. Then, the vector **E** can be calculated by multiplying the first column of **X** by c^2, as shown in Figure 6 (b).

$$\underset{\mathbf{E}}{\begin{pmatrix} 600.00 \\ 300.00 \\ 900.00 \\ 200.00 \\ 700.00 \end{pmatrix}} = \underset{\text{mass} \qquad \text{dark-matter}}{\begin{pmatrix} 6.00 & 14.16 & 72.84 & 49.42 & 26.72 \\ 3.00 & 83.37 & 75.00 & 36.21 & 28.21 \\ 9.00 & 27.55 & 32.70 & 59.96 & 20.56 \\ 2.00 & 28.83 & 34.27 & 35.39 & 74.98 \\ 7.00 & 55.43 & 86.68 & 38.21 & 87.91 \end{pmatrix}} * \underset{\mathbf{C}}{\begin{pmatrix} 100.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{pmatrix}}$$

(a) Calculating E with vector C^2

$$\underset{\mathbf{E}}{\begin{pmatrix} 600.00 \\ 300.00 \\ 900.00 \\ 200.00 \\ 700.00 \end{pmatrix}} = \underset{\textbf{mass}}{\begin{pmatrix} 6.00 \\ 3.00 \\ 9.00 \\ 2.00 \\ 7.00 \end{pmatrix}} * \underset{c^2}{100.00}$$

(b) Calculating E with the scalar value c^2

Figure 6: The calculated result of E with the vector C^2 and the scalar value c^2
Source: own.

In this example, if e is the i-th element of the vector **E** and m is also the i-th element of the vector **mass**, it can be found that $e = mc^2$. For example, for the first element of E and the first element of mass,

$E = 600;$
$m = 6;$
$c^2 = 100$, where, $c = 10;$
$600 = 6 * 10^2.$

That is,

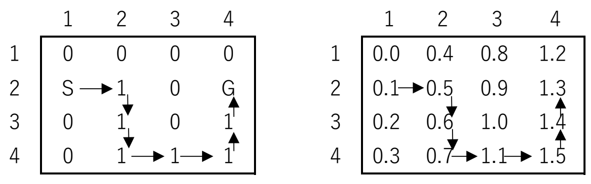
$$e = m\mathrm{c}^2.$$

To summarize, the dark-matter learning model employs matrix multiplication and defines dark-matter as chaotic space. It uses a state machine to transform chaotic space into ordered space. The concept of antimatter space is introduced, and the mass and energy equivalent equation explains the relationship between mass and visible object and measurable energy. A comprehensive example illustrates how vector **C** and vector **E** are computed. The relationship between mass and energy is described by the equation $e = mc^2$.

3 A knowledge representation method with "dark-matter"

The relationship between dark-matter and knowledge is explained in [22]. In the example shown in Figure 7, an agent's state is defined as its position in the maze. There are 16 possible positions, so the state space is represented by a 16-element vector. The agent's actions are defined as the four cardinal directions: up, down, left, and right. The start position is marked with the character "S" and the goal position is marked with the character "G". The agent will go along the path shown by the arrow-mark "→". The agent will go through the points (2,1), (2, 2), (3, 2), (4, 2), (4, 3), (4, 4) and (3, 4) and reach the goal position (2, 4), as shown in Figure 7 (a).

A space matrix can be defined by representing the agent's states as its positions on the maze matrix. There are 16 possible positions for the agent on the maze matrix, so the state space is represented by a 16-element matrix as shown in Figure 7 (b). There are 16 values from 0.0 to 1.5 which are used as the index of positions.



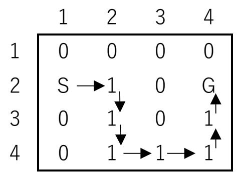
(a) A maze matrix **(b) The position index values of the maze matrix**

Figure 7: A maze matrix shown for an agent moving from "S" to "G"
Source: own.

If the agent does not have a position sensor, but it has a laser sensor, the output of the sensor shows which direction the agent can move. The states of the agent can be calculated through the output values of the laser sensor. However, the states of the agent cannot always be calculated without dark-matter. Let's take an example to explain it. The output values of the sensor are defined as follows:

- "1000" - Up
- "0100" - Down
- "0010" - Left
- "0001" – Right

The direction in which the agent can move at each position is shown in Figure 8 (a). The sensor's output values for each direction are shown in Figure 8 (b) and the sensor's output values when the agent is in different positions are shown in Figure 8 (c). For example, at the position (3, 2), the agent can move to "Up" direction to go back from the current position to its previous position., and "Down" directions to a new position, therefore, the output value of the sensor is "1100". If the agent is at the point (2, 2), the agent can move to two different positions, the agent can move back to the start position (2, 1) when it moves "Left". It can also move "Down" to the next position (3, 2). The output value of the sensor at the position (2, 2) is "0110", which is the sum of the two directions "0010", "Left" and "0100", "Down".



(a) The maze matrix

Up	Down	Left	Right
1000	0100	0010	0001

(b) The output of the laser sensor

	1	2	3	4
1	0	0	0	0
2	0001	0110	0	0100
3	0	1100	0	1100
4	0	1001	0011	1010

(c) The output of the laser sensor in the maze

Figure 8: The output of the laser sensor of the agent
Source: own.

As shown in Figure 8 (c), the sensor output values for the agent's positions (3, 2) and (3, 4) are both "1100". This means that the agent can move "Up" or "Down". If a function with the sensor value as the input of the function is used to retrieve the agent's action, two different actions, "Moving Down" and "Moving Up", will be returned. Therefore, without dark-matter, it is impossible to find a unique action for the agent. Dark-matter is used to calculate the best action to take.

The path from the start position to the goal position is not always known at the start position. For example, when reinforcement learning is used to find the path, the path is unknown at the start position. The path is found after many trials. Rewards are assigned to the found paths. The path with the shortest length is assigned the highest reward. The agent is trained to obtain the maximum reward during reinforcement learning. In this way, the optimal path from the start position to the goal position can be found. At the same time, the agent's actions at the positions on the path are determined.

When the path is unknown at the start position, it is impossible to create a space matrix as shown in Figure 8 (a). Here, a new method is proposed for generating the space matrix. This method records passed positions instead of the next positions. For example, when the agent moves from the start position (2, 1) to the position (2, 2), the passed position (2, 1) is recorded. This means that the dark-matter matrix can be created based on events that have occurred, rather than events that will occur in the future.

In the following, an example is used to illustrate the method in detail. In the example, the agent has a laser sensor.

The agent starts at the position (2, 1). It uses its laser sensor to scan the environment and detects that there are obstacles at the position (1, 1) and (3, 1). The agent then moves to the position (2, 2). It again uses its laser sensor to scan the environment and detects that there are obstacles at the position (1, 2) and (2, 4). Then, the agent moves to the position (3, 2). The agent continues to move in this way, recording the passed positions as it goes.

After a while, the agent has created a space matrix that shows all of the positions that it has passed. This matrix can be used to plan the agent's next move. For example, if the agent wants to move to the position (3, 4), it can use the space matrix to find the shortest path.

The new method for generating the space matrix is more efficient than the previous method. This is because the new method only records the passed positions, while the previous method had to record all of the possible positions. The new method is also more accurate, because it is based on events that have occurred, rather than events that will occur in the future.

The maze is represented by a matrix as shown in Figure 9(a), where the agent can be in any position marked with a "1" and cannot be in any position marked with a "0". The output of the sensor at each position is shown in Figure 9(b), and the index values of the sensor outputs are shown in Figure 9(c). For example, when the agent is at the position (2, 2), the output of the sensor is "1110", which indicates that the agent can move up, down, and left. The agent cannot be in the position where it is marked as "0" shown in Figure 9(a), so the outputs of the sensor at those positions are marked with an "X".

	1	2	3	4
1	1	1	1	0
2	1	1	0	1
3	0	1	0	1
4	0	1	1	1

(a) The maze matrix

	1	2	3	4
1	0101	0111	0111	X
2	1001	1110	X	0100
3	X	1100	X	1100
4	X	1001	0011	1010

(b) The output of the laser sensor

	1	2	3	4
1	0.5	0.7	0.7	X
2	0.9	1.4	X	0.4
3	X	1.2	X	1.2
4	X	0.9	0.3	1.0

(c) The index of the laser sensor

Figure 9: **The output of the laser sensor of the agent**
Source: own.

In Figure 9(a), the agent starts at position (2, 1) and can move to two different directions: up and right. The probability of moving up is 0.5 and the probability of moving right is 0.5. When the agent moves to position (2, 2), it can move up, down, or left. The probability of moving in each direction is 0.33. The target position is (2, 4). The agent receives a reward of 1 for each step it takes to reach the target position. The agent learns to move in the direction that leads to the highest reward. For example, if the agent moves to the right from position (2, 1), it will receive a reward of 1. This will increase the probability of the agent moving right in the future. After 2000 trials, the agent has learned to move from position (2, 1) to position (2, 4) in 7 steps. The probability of the agent moving in the direction that leads to the target position is 0.99.

To record sensor values, a working memory mechanism is used. The mechanism creates a vector with the same number of elements as the number of steps required for the agent to reach the goal position. For example, if the agent starts at position (2, 1) and moves through positions (2, 2), (3, 2), (4, 2), (4, 3), (4, 4), and (3, 4) to reach the goal position (2, 4), the vector will have 7 elements. The initial values of the elements are randomly assigned.

The current sensor value is then added to the vector, creating a new vector with 8 elements. The background of the element containing the current sensor value is white as shown in Figure 10(a), and the backgrounds of the elements containing the past sensor values are grey. For example, at the start position (2, 1), the current sensor value is 0.9, which is recorded in the first element of the vector. The background of the first element is white, and the backgrounds of the other elements are grey. The values of the elements from the second to the eighth are random because no sensor data has been recorded yet.

When the agent moves to position (2, 2), the index value of the sensor at that position, 1.4, is recorded in the first element of the vector. The previously recorded value, 0.9, is moved to the second element as shown in Figure 10(b). The backgrounds of the first and second elements are white, and the backgrounds of the other elements are grey.

This process continues as the agent moves through the maze as shown in Figure 10(c). When the agent reaches the goal position, all of the index values of the sensor outputs are recorded in the vector. The index value of the sensor output at the goal position is recorded in the first element, and the index value of the sensor output at the start position is recorded in the eighth element, as shown in Figure 10(d).

The working memory mechanism allows the agent to store a history of its sensor values. This information can be used to help the agent make decisions about where to move next. For example, if the agent has previously encountered a wall at a particular location, it is less likely to move in that direction in the future.

1	2	3	4	5	6	7	8
0.9	**94.0**	**84.0**	**12.0**	**78.0**	**58.0**	**84.0**	**23.0**

(a) At position (2,1)

1	2	3	4	5	6	7	8
1.4	**0.9**	**84.0**	**12.0**	**78.0**	**58.0**	**84.0**	**23.0**

(b) At position (2,2)

1	2	3	4	5	6	7	8
1.2	**1.4**	**0.9**	**12.0**	**78.0**	**58.0**	**84.0**	**23.0**

(c) At position (3,2)

1	2	3	4	5	6	7	8
0.4	**1.2**	**1.0**	**0.3**	**0.9**	**1.2**	**1.4**	**0.9**

(d) At position (2,4)

Figure 10: Working-memory and recorded index value of the sensor output
Source: own.

The agent's actions are recorded by an action index value. For example, if the agent moves to the right at the start position, the index value 0.1, which is the index value of moving to the right, is recorded. A vector **E** is used to record all the action index values from the start position to the end position, as shown in Figure 11(c).

As shown in Figure 11(a), the space **X** is a collection of the working memory of the agent moved from the start position to the goal position. Its inverse matrix $\mathbf{X}^{-1}$ is shown in Figure 11(b). By multiplying $\mathbf{X}^{-1}$ by **E**, a vector **C** is generated, as shown in Figure 11(d).

$$\begin{pmatrix} 0.3 & 0.9 & 1.2 & 1.4 & 0.9 & 58.0 & 84.0 & 23.0 \\ 0.4 & 1.2 & 1.0 & 0.3 & 0.9 & 1.2 & 1.4 & 0.9 \\ 0.9 & 94.0 & 84.0 & 12.0 & 78.0 & 58.0 & 84.0 & 23.0 \\ 0.9 & 1.2 & 1.4 & 0.9 & 78.0 & 58.0 & 84.0 & 23.0 \\ 1.0 & 0.3 & 0.9 & 1.2 & 1.4 & 0.9 & 84.0 & 23.0 \\ 1.2 & 1.4 & 0.9 & 12.0 & 78.0 & 58.0 & 84.0 & 23.0 \\ 1.2 & 1.0 & 0.3 & 0.9 & 1.2 & 1.4 & 0.9 & 23.0 \\ 1.4 & 0.9 & 84.0 & 12.0 & 78.0 & 58.0 & 84.0 & 23.0 \end{pmatrix}$$

(a) X

$$\begin{pmatrix} -0.03 & 2.77 & -0.03 & 0.03 & 0.01 & -0.03 & -0.06 & 0.00 \\ 0.00 & 0.01 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 & -0.01 \\ 0.00 & -0.01 & 0.00 & 0.00 & 0.00 & -0.01 & 0.00 & 0.01 \\ 0.00 & -0.08 & 0.00 & -0.09 & 0.00 & 0.09 & 0.00 & 0.00 \\ -0.01 & -0.02 & 0.00 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.02 & 0.03 & 0.00 & 0.00 & -0.02 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 & -0.01 & 0.00 \\ 0.00 & -0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.05 & 0.00 \end{pmatrix} \quad \begin{bmatrix} 0.1 \\ 0.0 \\ 0.1 \\ 0.1 \\ 0.8 \\ 0.4 \\ 0.8 \\ 0.4 \end{bmatrix} \quad \begin{bmatrix} -0.06 \\ 0.00 \\ 0.00 \\ 0.03 \\ 0.00 \\ -0.01 \\ 0.00 \\ 0.04 \end{bmatrix}$$

(b) $\mathbf{X}^{-1}$ **(c) E** **(d) C**

Figure 11: Working-memory and recorded index value of the sensor output
Source: own.

When the vector **C** is generated, the agent can calculate its actions at each relative position by multiplying the vector of the working-memory by **C**. This is expressed by equation (1) in Section 2.

The dark-matter matrix can be used to calculate a unique action index value even if the output values of the sensor are the same. For example, at the position (3, 2) and (3, 4), the index values of the sensor output value are both 1.2, as shown in Figure

12 (a) and (b). The index values are recorded at the first elements of the two vectors. In the working-memory, the values of the dark-matter values in the second to the eighth elements of the two vectors are different, as shown in Figure 12 (a) and (b).

1	2	3	4	5	6	7	8
1.2	1.4	0.9	12.0	78.0	58.0	84.0	23.0

(a) Values of the working-memory at position (3,2)

1	2	3	4	5	6	7	8
1.2	1.0	0.3	0.9	1.2	1.4	0.9	23.0

(b) Values of the working-memory at position (3,4)

Figure 12: Working-memory and recorded index value of the sensor output
Source: own.

The working-memory vectors are multiplied by the energy vector **E** to produce two action index values: 0.4 and 0.8, as shown in equation (3) and (4). The index value 0.4 indicates that the agent should move down to the position (3, 2), and the index value 0.8 indicates that the agent should move up to the position (3, 4).

$$[1.2, 1.4, 0.9, 12.0, 78.0, 58.0, 84.0, 23.0] \times \begin{bmatrix} -0.06 \\ 0.00 \\ 0.00 \\ 0.03 \\ 0.00 \\ -0.01 \\ 0.00 \\ 0.04 \end{bmatrix} = 0.4 \tag{4}$$

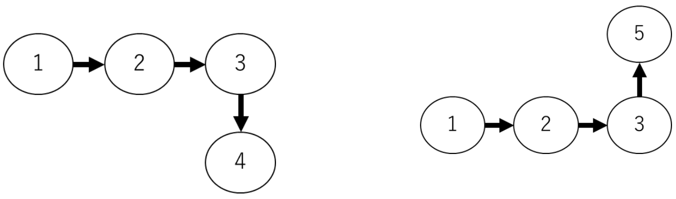
$$[1.2, 1.0, 0.3, 0.9, 1.2, 1.4, 0.9, 23.0] \times \begin{bmatrix} -0.06 \\ 0.00 \\ 0.00 \\ 0.03 \\ 0.00 \\ -0.01 \\ 0.00 \\ 0.04 \end{bmatrix} = 0.8 \tag{5}$$

The agent learns to take appropriate actions at different positions to move from the start position to the goal positions. This is done through experience, which is a form of knowledge. In conclusion, knowledge can be expressed in the dark-matter matrix of the space matrix. This is because knowledge is a form of information, and information can be encoded in the dark-matter matrix. The dark-matter matrix is possible to contains all of the knowledge that has ever been known.

4 The phenomenon of inapplicability of experience and the parallel spaces

The phenomenon of inapplicability of experiences often happened when we use our experiences to make judgements. When we made judgements, it often happened that wrong judgments and decisions are made based on experiences. For example, in the stock trading process, we will determine the current trade based on the stock's past ups and downs and our trading experience. However, it always happened that when we decided to buy, the stock fell, or when we decided to sell, the stock rose.

Figure 13 shows an example of the phenomenon of inapplicability of experience. In the figure, the circles are the states of an agent and arrows indicate state migration. As shown in Figure 13 (a), at the beginning, the agent is in state 1, then the it will move to the next state, state 2. Once it reaches state 3, it migrates to the next state, state 4. If an agent is trained and got the experienced as shown in Figure 13 (a), the agent will always move from the state 3 to state 4. However, if the next state of state 3 is state 5, the phenomenon of inapplicability of experiences happens as shown in Figure 13 (b).



$$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] & 3 \arrow[d] \\ & & 4 \end{tikzcd} \qquad \begin{tikzcd} & & 5 \\ 1 \arrow[r] & 2 \arrow[r] & 3 \arrow[u] \end{tikzcd}$$

(a) The next state of state 3 should be state based on the experience

(b) If the next state of state 3 is not state 4 but state 5, then the experience of (a) is inapplicable.

Figure 13: An example of the phenomenon of inapplicability of experience

When the phenomenon of inapplicability of the experience occurs, it is impossible to decide which state will be the next in the current state, as shown in Figure 14. In

Figure 14, both state 4 and state 5 are the next states after state 3. In this case, it's impossible to empirically decide which state will be the next state of state 3.

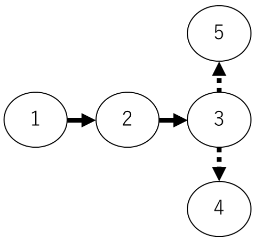
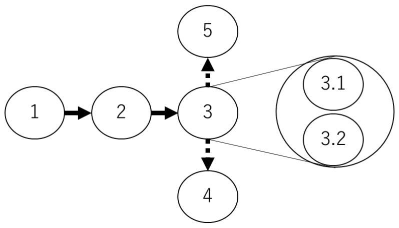
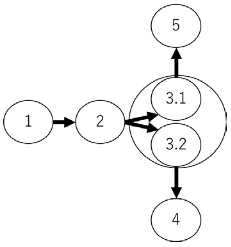


Figure 14: Both the state 4 and the state 5 will be the next state of state 3
Source: own.

A reason for the phenomenon of inapplicability of experiences is due to the sensor's lack of detection accuracy. When the sensor's detection accuracy is insufficient, it is impossible to detect whether the current state should be state 3.1 or state 3.2, as shown in Figure 15 (a). Since it could only detect the current state as state 3, it is impossible to determine whether the next state should be state 4 or state 5. When the sensor has enough detection accuracy to detect whether the current state is state 3.1 or state 3.2, it can be determined whether the next state should be state 4 or state 5, as shown in Figure 15 (b).



(a) Sensor's lack of detection accuracy



(b) The sensor has enough detection accuracy

Figure 15: Sensor's lack of detection accuracy causes the phenomenon of inapplicability of experiences.
Source: own.

The **parallel space model** proposed in this paper is a model to illustrate the phenomenon of inapplicability of experience. Figure 16 is an example. In figure 16, there are an observation space and two parallel spaces, Space1 and Space2. The observation space is a projection space of the parallel spaces. That is, the observation space is a space the sensor can detect. In Figure 16, the sensor can only detect *x* and *y* direction. It cannot detect *z* direction.

If two agents are trained in Space1 and Space2, respectively, they will move from state 3 to state 4 and state 5, respectively. The phenomenon of inapplicability of experience will not happen. But in the observation space, as shown in Figure 16, the phenomenon of inapplicability of experience will happen. If the agent moves in Space2 based on the experience obtained from Space1, from state 3, it will move to the wrong sate, state 4, which is not existed in Space2. If the agent moves in Space1 based on the experience obtained from Space2, from state 3, it will move to the wrong sate, state 5, which is not existed in Space1. From the point view in the observation space, from state 3, sometimes the agent moves to state 4 but sometime it moves state 5. In the observation space, we cannot predict what state the agent will move to from state 3 until it has moved.

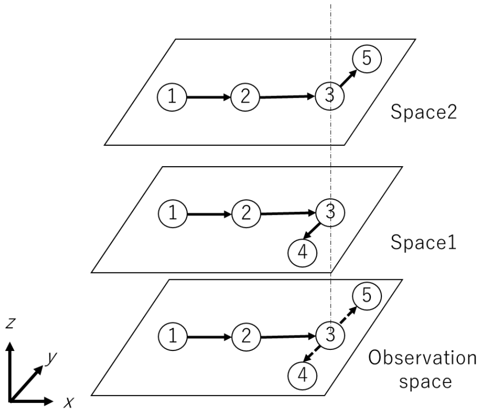


Figure 16: An example of parallel spaces
Source: own.

Suppose an agent is used to control an automated cleaning robot and it will clean two floors. The two floors are represented by Space1 and Space2, respectively. The robot's position on the floor plane is represented by x and y coordinates. The floor where the robot is located is represented by z-coordinates. The robot's sensors can detect positions of the robot in the x and y directions, but not the z direction. The positions of the robot are represented as 1, 2, 3 and 4 when the robot is on the Space1 floor plan. If the robot is on the Space2 floor plan, the positions of the robot are represented as 1, 2, 3 and 5. When the robot is on the floor plan Space1, as shown in Figure 17 (a), the matrix X is constructed with the elements of the first column are the position of the robot and the other columns are experiences and dark-matter. The inverse matrix of the matrix **X**, **X**$^{-1}$ is represented in Figure 17 (b). The elements of vector **E** are the next positions of the current positions. As shown in Fugure 17 (c), the next positions are 2, 3 and when the current positions are 1, 2 and 3, respectively. When the robot reached to the position 4, its next position is also 4. The rule vector **C**, shown in Figure 17 (d), is calculated multiplying **X**$^{-1}$ by **C**.

$$\begin{pmatrix} 1.0000 & 94.0000 & 84.0000 & 12.0000 \\ 2.0000 & 1.0000 & 84.0000 & 12.0000 \\ 3.0000 & 2.0000 & 1.0000 & 12.0000 \\ 4.0000 & 3.0000 & 2.0000 & 1.0000 \end{pmatrix} \quad \begin{pmatrix} -0.0081 & 0.0020 & -0.0158 & 0.2629 \\ 0.0107 & -0.0107 & -0.0002 & 0.0028 \\ 0.0000 & 0.0119 & -0.0122 & 0.0032 \\ 0.0002 & 0.0003 & 0.0883 & -0.0665 \end{pmatrix} \quad \begin{pmatrix} 2 \\ 3 \\ 4 \\ 4 \end{pmatrix} \quad \begin{pmatrix} 0.9781 \\ -0.0002 \\ -0.0003 \\ 0.0889 \end{pmatrix}$$

(a) X **(b) X^{-1}** **(c) E** **(d) C**

Figure 17: Space1
Source: own.

Figure 18 shows the matrices when the robot is on the floor plan Space2. Same as those of in Figure 17, the space **X**, its inverse matrix **X**$^{-1}$, the next position vector **E** and the rule vector **C** are represented in Figure 18 (a), (b), (c) and (d), respectively.

$$\begin{pmatrix} 1.0000 & 94.0000 & 84.0000 & 12.0000 \\ 2.0000 & 1.0000 & 84.0000 & 12.0000 \\ 3.0000 & 2.0000 & 1.0000 & 12.0000 \\ 5.0000 & 3.0000 & 2.0000 & 1.0000 \end{pmatrix} \quad \begin{pmatrix} -0.0064 & 0.0016 & -0.0125 & 0.2082 \\ 0.0107 & -0.0107 & -0.0001 & 0.0022 \\ 0.0001 & 0.0119 & -0.0122 & 0.0025 \\ -0.0002 & 0.0004 & 0.0875 & -0.0526 \end{pmatrix} \quad \begin{pmatrix} 2 \\ 3 \\ 5 \\ 5 \end{pmatrix} \quad \begin{pmatrix} 0.9701 \\ -0.0003 \\ -0.0124 \\ 0.1752 \end{pmatrix}$$

(a) X **(b) X^{-1}** **(c) E** **(d) C**

Figure 18: Space2
Source: own.

As shown in Figure 16, on both floor plans, Space1 and Space2, the agent moves in the same way from the position 1 to the position 3. Therefore, experiences of the agent at the position 3 are the same. The experience stored in the working memory is shown in Figure 19. It is a row vector. Its first element 3.0000 represents the current robot's position, the second element 2.0000 presents its previous position and the third element 1.0000 is the start position of the robot. The last element 12.0000 is the *dark-matter.*

$$\begin{pmatrix} 3.0000 & 2.0000 & 1.0000 & 12.0000 \end{pmatrix}$$

Figure 19: Experiences stored in the working memory
Source: own.

Although the same experience is stored the in working memory at position 3, different next position is obtained using different rule vectors. The rule vector **C** of Space1 and the rule vector **C** of Space2 are shown in Figure 17 and 18 (d), respectively. Multiplying the row vector stored in the working memory by the rule vector **C** respectively, the next position, 4, in the Space1, and the next position, 5, are calculated respectively, as shown in Figure 20 (a) and (b).

$$\begin{pmatrix} 3.0000 & 2.0000 & 1.0000 & 12.0000 \end{pmatrix} * \begin{pmatrix} 0.9781 \\ -0.0002 \\ -0.0003 \\ 0.0889 \end{pmatrix} = 4$$

(a) Calculating the next state at state 3 using the rule of Space1

$$\begin{pmatrix} 3.0000 & 2.0000 & 1.0000 & 12.0000 \end{pmatrix} * \begin{pmatrix} 0.9701 \\ -0.0003 \\ -0.0124 \\ 0.1752 \end{pmatrix} = 5$$

(b) Calculating the next state at state 3 using he rule of Space2

Figure 20: Calculating the next state at state
Source: own.

Dimensional expansion is necessary when the phenomenon of inapplicability of experience happens. In the example shown in Figure 16, if only the x and y dimensions are used, experience is not applicable to decide which state, state 4 or 5 will be the next state when the current state is state 3. Adding a new sensor is one of the methods for dimensional expansion. For example, we can add a color sensor to the cleaning robot to detect which floor it is on. Painting the floor of Space1 as yellow and the floor of Space2 as green, the output of the color sensor will be different on different floors. As a result, it can be found out which floor the robot is on and which will be the next position when the robot is at the position 3.

Another way for the dimensional expansion is to use sensors which can detect the next possible moving position. For example, if the sensors of the cleaning robot can detect whether the it can move to the position 4 or 5 at the position 3, it is possible to decide which will be the next position when the robot reached at the position 3.

Trial-and-error is also a dimensional expansion method. Suppose the robot is on the floor Space1. If the robot hits an obstacle as it moves from the position 3 to the next position, position 4, it can be found that the robot is not on the floor Space1 but on the floor Space2.

5 Conclusion and future work

In this paper, the concept of "***dark-matter***" and its relative concept on experience and knowledge are reviewed. The concept of "***space***" represented as a matrix created based on experience is also reviewed. In addition, the characteristic of the concept of space's "***rule***" is revealed, indicating that it is useful to transform experiences into the output of agents according to the inputs of the agents. Examples are used to illustrate experience and knowledge expression. The concept of the working memory mechanism is reviewed which is used to recorde the agent's experience and create the space matrix **X**. A new model "parallel spaces" is presented which is useful to illustrate the "phenomenon of inapplicability of experience." This model revealed why mistake decisions based on experiences. The most important contribution of this paper is that we revealed how the "phenomenon of inapplicability of experience" happened and proposed solutions. In the paper, it is also proposed that for each space, only one rule vector is required to calucate output results for the given input. Therefore, only a two-dimensional matrix, of which each column is a rule vector, is required for the parallel space model. Dimensional expansion is also required to decide which rule vector, or in other words, which space is used. Solutions for the dimensional expansion is presented. Adding new sensors and trial-and-error are the two methods for the dimensional expansion. As our future work, application systems based on the proposed methods and the mechanism will be developed.

References

[1] Chen, X. and Kiyoki, Y., "On Semantic Spatiotemporal Space and Knowledge with the Concept of "Dark-Matter"," Information Modelling and Knowledge Bases XXXIII, IOS Press, pp.110-128, 2021.

[2] Chen, X. and Kiyoki, Y., "A query-meaning recognition method with a learning mechanism for document information retrieval," Information Modelling and Knowledge Bases XV, IOS Press, Vol. 105, pp.37-54, 2004.

[3] Chen, X. and Kiyoki, Y., "A dynamic retrieval space creation method for semantic information retrieval," Information Modelling and Knowledge Bases XVI, IOS Press, Vol. 121, pp.46-63, 2005.

[4] Kiyoki, Y. and Kitagawa, T., "A semantic associative search method for knowledge acquisition," Information Modelling and Knowledge Bases, IOS Press, Vol. VI, pp.121-130, 1995.

[5] Kitagawa, T. and Kiyoki, Y., "A mathematical model of meaning and its application to multidatabase systems," Proc. 3rd IEEE International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems, pp.130-135, April 1993.

[6] Chen, X., Kiyoki, Y. and Kitagawa, T., "A multi-language oriented intelligent information retrieval system utilizing a semantic associative search method," Proceedings of the 17th IASTED International Conference on Applied Informatics, pp.135-140, 1999.

[7] Chen, X., Kiyoki, Y. and Kitagawa, T., "A semantic metadata-translation method for multilingual cross-language information retrieval," Information Modelling and Knowledge Bases XII, IOS Press, Vol. 67, pp.299-315, 2001.

[8] Kiyoki, Y., Kitagawa, T. and Hitomi, Y., "A fundamental framework for realizing semantic interoperability in a multidatabase environment, " International Journal of Integrated Computer-Aided Engineering, Vol.2, No.1(Special Issue on Multidatabase and Interoperable Systems), pp.3-20, John Wiley & Sons, Jan. 1995.

[9] Kiyoki, Y., Kitagawa, T. and Hayama, T., "A metadatabase system for semantic image search by a mathematical model of meaning, " ACM SIGMOD Record, Vol.23, No. 4, pp.34-41, Dec. 1994.

[10] Kiyoki, Y, Chen, X. and Kitagawa, T., "A WWW Intelligent Information Retrieval System Utilizing a Semantic Associative Search Method," APWeb'98, 1st Asia Pacific Web Conference on Web Technologies and Applications, pp. 93-102, 1998.

[11] Ijichi, A. and Kiyoki, Y.: "A Kansei metadata generation method for music data dealing with dramatic interpretation," Information Modelling and Knowledge Bases, Vol.XVI, IOS Press, pp. 170-182, May, 2005.

[12] Kiyoki, Y., Chen, X. and Ohashi, H.: "A semantic spectrum analyzer for realizing semantic learning in a semantic associative search space," Information Modelling and Knowledge Bases, Vol.XVII, IOS Press, pp.50-67, May 2006.

[13] Takano, K. and Kiyoki, Y.: "A causality computation retrieval method with context dependent dynamics and causal-route search functions," Information Modelling and Knowledge Bases, ISO Press, Vol.XVIII, pp.186-205, May 2007.

[14] Chen, X. and Kiyoki, Y.: "A visual and semantic image retrieval method based on similarity computing with query-context recognition," Information Modelling and Knowledge Bases, IOS Press, Vol.XVIII, pp.245-252, May 2007.
[15] Nitta T, "Resolution of singularities introduced by hierarchical structure in deep neural networks," IEEE Trans Neural Netw Learn Syst., Vol.28, No.10, pp.2282-2293Oct. 2017.
[16] Wiatowski, T. and Bölcskei, H., "A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction," IEEE Transactions on Information Theory, PP(99) · Dec. 2015.
[17] Hochreiter, S., Bengio, Y., Frasconi, P. and Schmidhuber, J. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," In Kremer, S. C. and Kolen, J. F. (eds.), A Field Guide to Dynamical Recurrent Neural Networks, IEEE Press, 2001.
[18] Hochreiter, S. and Schmidhuber, J., "Long short-term memory,". Neural computation, Vol.9, No.8, pp.1735-1780, 1997.
[19] Kalchbrenner, N., Danihelka, I. and Graves, "A. Grid long short-term memory," CoRR, abs/1507.01526, 2015.
[20] Chen, X. and Kiyoki, Y., "On Logic Calculation with Semantic Space and Machine Learning," Information Modelling and Knowledge Bases XXXI, IOS Press, Vol. 321, pp.324-343, 2019.
[21] Chen, X. Prayongrat, M. and Kiyoki, Y., "A Concept for Control and Program Based on the Semantic Space Model," Information Modelling and Knowledge Bases XXXII, IOS Press, Vol. 333, pp. 26-44, 2020.
[22] Chen, X., "An Exploratory Research on the Expression of Knowledge and Its Generation Process based on the Concept of "Dark-matter", Information Modelling and Knowledge Bases XXXIV, IOS Press, Vol. 364, pp. 110-124, 2023.