Introduction to Inventory Space

Joseph Juma

Independent

joseph@josephjuma.com

April 2026

ABSTRACT

We introduce *Inventory Space*, a heterogenous metric product space for representing discrete inventory systems, which combines Hamming-type categorical axes and integer-valued Euclidean axes to model inventory systems, such as those found in video games.

1 Introduction

When designing certain systems, it's useful to have a way to represent discrete inventory systems, like those found in roleplaying games (*RPGs*), in a geometric manner. For example, I'm currently working on a prototype for the creation of a super-position collapsed (*SPC*) based artificial intelligence system [1] for agentic AI in first-person shooter environments which requires a metric space on inventory systems.

2 Background

2.1 Discrete Inventory Systems

A **discrete inventory system**(*DIS*) is any system that stores a list of items, either with each specific item constituting a new entry into the list, or with an associated integer value indicating the quantity of a given item. For example, one might think of the inventory in a traditional Japanese roleplaying game (*JRPG*) such as those found in either the *Final Fantasy* or *Golden Sun* series. In these games, a character can possess a number of different items, each with a known quantity of it.

Some DIS have a trait whereby some items are noted as being either *unique* (i.e. a user cannot possess more than one of them) or, alternatively, *non-stacking* (i.e. each entry of the item has a maximum quantity of 1). In such systems, a given item can be represented by a boolean value (i.e. either a 1 or 0).

For items that can have more than one instance per-entry (*stacking*) one can instead use an integer value representation.

For an inventory of n-many items, an ordered n-tuple may be used where each axes is either an integer-valued axis representing a stackable item entry, or a boolean-valued axis representing a non-stackable or unique item entry. This **inventory n-tuple** is a representation of an inventory space.

Additional operations over inventory tuples, including the addition of a distance metric further allow us to transform the arbitrary inventory-tuple into an inventory vector in a wider inventory space. To understand

inventory space, we must first address the geometrical spaces of each axes type.

2.2 Geometrical Metric Spaces

If unfamiliar, a *geometrical metric space* is the mathematical term for spaces like those found in common geometry - which is technically referred to as *Euclidean geometry*.

Very quickly: a *set* is any collection of points (i.e. arbitrary values). An *algebra* is a set with the operations on the points in that set. For example, the Booelan set is $\{0, 1\}$, and the Boolean algebra is the Boolean set along with operations on it like the common logic operations (e.g. AND, OR, $NOT \ldots$).

A *space* is a set or algebra with added structure (i.e. mathematical relationships between points). A *metric space* is a set, or algebra, with a *distance metric* function over the points in that space that calculates a value similar to how we measure distance in classical (i.e. Euclidean) geometry. Technically speaking, a distance metric itself defines a type of structure, so as long as a distance metric exists on a set - it's a space.

Loosely speaking, *geometrical space* is a space that has behavior we can vaguely attribute as forming shapes.

So, together a *geometrical metric space* is a set of values (i.e. points), operations on those values, a structure and a distance metric (which itself can be the structure) that can produce something resembling geometry.

For example, because the xor operation has the following truth table:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Table 1: XOR truth table

It can be used as the distance on the boolean set because $\Delta(1, 0) = 1, \Delta(1, 1) = 0, \Delta(0, 0) = 0$. So we can mathematically describe a one-dimensional Boolean metric space as:

$$(\{0, 1\}, \Delta(A, B) = A \oplus B) \tag{1}$$

Given this, we can now proceed to discussing specific geometrical metric spaces.

2.3 Hamming Space

Hamming Space is an n-dimensional geometrical metric space defined over the Boolean Algebra (i.e. $\{0, 1\}^n$). It uses the **Hamming Metric** as its distance metric with the formula:

$$\Delta(A, B) = \sum_{i=1}^{n} a_i \oplus b_i \tag{2}$$

Where...

1. Δ is the distance between two given points.

2. A and B are points in n-dimensional boolean space (i.e. n-tuples of boolean values).
3. a_i is the i-th element in A.
4. b_i is the i-th element in B.
5. n is the dimensions in the boolean space.
6. $\oplus$ is the logical xor operation.

For example, if in the $n = 3$ Hamming space, we might have two points: $A = (0, 1, 1)$ and $B = (1, 0, 1)$. The distance between these using the Hamming metric is then:

$$\Delta(A, B) = (0 \oplus 1) + (1 \oplus 0) + (1 \oplus 1) = 1 + 1 + 0 = 2 \tag{3}$$

This allows us to represent a collection of n-many boolean values as a metric space, which can result in geometrical behaviors.

2.4 Euclidean Space over Integers

An n-dimensional **Euclidean Space over Integers** is an n-dimensional space over the set of integers, rather than real numbers, so each vector can only have whole number values (e.g. 1, 3, 131) rather than real numbers value (e.g. 3.1415, 0.5, 35.50). The Euclidean integer space uses the Euclidean distance metric just like standard Euclidean space:

$$\Delta(A, B) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2} \tag{4}$$

Where...

1. Δ is the distance between two given points.
2. A is an n-dimensional point in Euclidean integer space.
3. a_i is the i-th integer value in A.
4. B is an n-dimensional point in Euclidean integer space.
5. b_i is the i-th integer value in B.
6. n is the number of dimensions in the space.

For example, the two points $A = (1, 2, 3)$ and $B = (0, 4, 7)$ the distance between them is:

$$\Delta(A, B) = \sqrt{(0-1)^2 + (4-2)^2 + (7-3)^2} = \sqrt{1 + 4 + 16} = \sqrt{21} = 4.5825\ldots \tag{5}$$

2.5 Product Space

A **product space** is a space comprised of the combination of one or more other spaces. More formally, a product space is the *cartesian product of two or more topological spaces*.

The **Cartesian product** is merely the set resulting from combining every element in two sets. For example, given two sets $A = \{0, 1\}$ and $B = \{a, b, c\}$ the Carteisan product would be $A \odot B = \{(a, 0), (a, 1), (b, 0), (b, 1), (c, 0), (c, 1)\}$.

A **topological space** is any space that has a defined notion of "closenes" between two or more points, even if not necessarily able to be measured as a distance. So, most (if not all) metric spaces with a distance metric are topological spaces. As such, we can combine the previously mentioend geometrical spaces into product spaces.

3 Inventory Space

An n-dimensional **Inventory Space** is a product metric space with heterogeneous axis, whereby each element (i.e. dimension or axes) is either a boolean value (i.e. Hamming dimension) or integer value (i.e. Euclidean-integer dimension). The distance metric Δ over n-dimensional inventory space is a complex function, which applies a different function depending on the category of dimension, with the following formula:

$$\Delta(A, B) = \sqrt{\sum_{i=1}^{n} \begin{cases} integer : (b_i - a_i)^2 \\ boolean : b_i \oplus a_i \end{cases}} = \sqrt{\sum_{i=1}^{n} \begin{cases} (a_i, b_i) \in \mathbb{Z} : (b_i - a_i)^2 \\ (a_i, b_i) \in \{0, 1\} : b_i \oplus a_i \end{cases}} \tag{6}$$

Where...

1. $\Delta(A, B)$ is the distance between points A and B.
2. a_i is the i-th element in the point A.
3. b_i is the i-th element in the point B.
4. $x \oplus y$ is the logical xor operation between boolean values x and y.

This distance metric is a hybrid whereby each of the axes where the value is from the boolean set employ the xor-operator for distance, while any integer value uses the Euclidean distance method, with them being combined using the square-root method to encode distance back to one that behaves similar to common Euclidean geometry (i.e. Euclidean geometry over real numbers).

This space allows us to model an inventory. However, to better grasp the intuition on this it's useful to discuss *item space* - the combined space of a singular item.

3.1 Item Space

Suppose we have an item, such as a sci-fi assault rifle. This rifle has two firing modes: one that shoots electromagnetic rounds rapidly like normal bullets, and another which emits a high-energy projectile that vaporizes anything it touches. Each of these attacks consume different ammunition. The **item space** for this item looks like:

1. A boolean value of 1 for possessing the item.

2. An integer value indicating the standard fire (i.e. *primary-fire*) ammunition.
3. An integer value indicating the alternative energy projectile (i.e. *secondary-fire*) ammunition.

Suppose we hadn't yet acquired the weapon, but had 15 of the primary ammunition and 5 of the second. The resulting point would be $A = (0, 15, 5)$. Suppose we pick up the scifi rifle, a lot of primary ammo (i.e. we now have 80) and expend all the secondary-fire. The second point would now be $B = (1, 80, 0)$.

The **item distance** (i.e. inventory distance for a singular item) is given by the inventory distance metric on these axes:

$$\Delta(A, B) = \sqrt{(1 \oplus 0) + (80 - 15)^2 + (0 - 5)^2} = \sqrt{1 + 4225 + 25} = 65.19969\ldots \tag{7}$$

An additional interesting trick with item-spaces is they can be condensed down encoding wise by projecting the boolean value onto the ammunition values. For example:

$$(a_0 * a_1, a_0 * a_2) = (1 * 80, 1 * 0) = (80, 0) \tag{8}$$

Because if you don't have a given item, then you won't be able to use it. If you have 0 ammunition for the item, you also won't be able to use it. Therefore, 0 ammunition and a 0 ownership-bit can be considered the same. On the other hand, any non-zero values combined with an owned item mean it can be used. So this short-hand can be used to extract the bit information, though in many systems it's good to still have a distinct bit for item ownership agnostic of ammunition or charge values.

3.2 Weighted Item-Space

Consider we have an inventory with three items, one unique (a weapon) and two not:

$$I_0 = (\text{sword} : (1), \text{potion} : (1, 5), \text{ether} : (1, 5)) \tag{9}$$

If we use one potion, this becomes:

$$I_1 = (\text{sword} : (1), \text{potion} : (1, 4), \text{ether} : (1, 5)) \tag{10}$$

The distance between these two inventory points is then:

$$\Delta(I_0, I_1) = \sqrt{0 + 0 + 1 + 0} = \sqrt{1} = 1 \tag{11}$$

However, if we remove the sword instead, we get:

$$I_2 = (\text{sword} : 0, \text{potion} : (1, 5), \text{ether} : (1, 5)) \tag{12}$$

Which has a distance of:

$$\Delta(I_2, I_0) = \sqrt{1 + 0 + 0 + 0} = 1 \tag{13}$$

In many cases this is fine, where the absence of an item is weighted equally with the consumption of another common item. However, there are times when one wishes to treat the absence of a given item, perhaps based on item type, with a higher weight than the loss of a single charge of a given item. This results in using **Weighted Inventory-Spaces** (WIS).

3.2.1 Weighted Item Space Distance Metric

A WIS differs from a common Inventory-Space by defining coefficients for the two components of the distance function. This updates the distance metric into:

$$\Delta(A,B) = \sqrt{\sum_{i=1}^{n} \begin{cases} integer : w_I(b_i - a_i)^2 \\ boolean : w_B(b_i \oplus a_i) \end{cases}} = \sqrt{\sum_{i=1}^{n} \begin{cases} (a_i, b_i) \in \mathbb{Z} : w_I(b_i - a_i)^2 \\ (a_i, b_i) \in \{0,1\} : w_B(b_i \oplus a_i) \end{cases}} \tag{14}$$

Where...

1. w_I is the weight coefficient for integer values.
2. w_B is the weight coefficient for boolean values.

This allows for a designer to selectively weight the two distinct behaviors (i.e. possessing an item, vs change in quantity) differently which can effect the resulting geometries and distance-derived mechanisms.

Alternatively, rather than using a set weight for each category of axes, each axes may itself be weighted by an established coefficient. One particularly useful variant of this, when a given item has a maximum quantity it might become, multiplies the ownership boolean value by the maximum quantity value in order to weight a given item's ownership proportional to its maximum capabilities. This produces a distance metric such as:

$$\Delta(A,B) = \sqrt{\sum_{i=1}^{n} \begin{cases} integer : (b_i - a_i)^2 \\ boolean : w_{B,i}(b_i \oplus a_i) \end{cases}} = \sqrt{\sum_{i=1}^{n} \begin{cases} (a_i, b_i) \in \mathbb{Z} : (b_i - a_i)^2 \\ (a_i, b_i) \in \{0,1\} : w_{B,i}(b_i \oplus a_i) \end{cases}} \tag{15}$$

Where, $w_{B,i}$ is the weight for boolean value i based on the corresponding maximum value of the corresponding integer value.

4 Conclusion

With the given formulation herein, I've now covered the basics of inventory space. Thank you for reading, and have a great day.

References

[1] Joseph Juma. *WO2026035737 - System and Methods of Training and Operation for a General Artificial Intelligence System for Data Generation.* 2026. International Patent Application (WIPO).