Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 133–143
DOI: 10.15439/2025F6266
ISSN 2300-5963 ACSIS, Vol. 43

A New Perspective of Associative Memories for Partial Patterns

Soma Dutta
0000-0002-7670-3154
University of Warmia and Mazury in Olsztyn
ul. Słoneczna 54, 10-710 Olsztyn, Poland
Email: soma.dutta@matman.uwm.edu.pl

Jaewon Shin
0009-0005-5284-295X
University of Warmia and Mazury in Olsztyn
ul. Słoneczna 54, 10-710 Olsztyn, Poland
Email: jaewon.shin@student.uwm.edu.pl

***Abstract*—This study aims to present a new perspective for creating associative memories using incomplete data and project them into artificial neural networks to retrieve complete data from incomplete data. In previous studies, various approaches to construct associative memories and retrieval of stored data have been proposed. This paper attempts to pin point some of the limitations observed in the existing approaches and propose a way to get rid of that. In particular, a different perspective of comparing two incomplete patterns is proposed and based on that a flexible way of constructing associative patterns of a given input (partial) pattern is developed. Finally, the respective neural network architecture is proposed following similar construction reported in the existing research.**

I. Introduction

PATTERN recognition is a natural cognitive process that attempts to match an encountered instance of a pattern with the already stored patterns in the memory and thereby retrieve (plausibly) relevant information about the new pattern. Learning from patterns, extracting association among patterns and then applying that knowledge in recognizing useful patterns in different contexts of decision making characterize different important features of human intelligence [1]. The emergence of artificial neural network (ANN) [2] as a prototypical model of human brain, or more specifically biological neural network, is already well known. From its inception ANN has been considered as an effective tool in pattern recognition, signal processing, symbol processing etc [3], [4], [5]. The task of pattern association generally aims at storing different patterns similar to a given input pattern so that a concept related to the given input pattern can be recalled when a new pattern appears to be similar or a member of the stored patterns. Variations of a given input pattern is a common phenomenon, often caused by noise incorporated while collecting data or partial specification of data due to incomplete information. Content-Addressable Memory (CAM) or Associative memory (AM) [5], [6], [7] is the research domain of ANN which concerns about associating similar patterns of a given input data to create a memory and recalling the concepts associated to the input data for those stored patterns and implement the idea in the context of ANN.

In [3] the authors described that "The term associative memory (AM) or content-addressed memory refers to a memory system where recall of a stored pattern is accomplished by providing a noisy or partially specified input pattern.". A mathematical representation of the problem is as follows.

Given a data containing a set of input-output pairs $(\vec{u_1}, \vec{v_1})$, $(\vec{u_2}, \vec{v_2}), \ldots, (\vec{u_k}, \vec{v_k})$ where each input vector $\vec{u_i}$ of dimension n is connected to an output vector $\vec{v_i}$ of dimension m, the task of learning the relation between input-output patterns is realized through standard perceptron model with different activation functions. However, the additional point of associating any pattern from the associative memory of a given input pattern $\vec{u_i}$ with its output $\vec{v_i}$ is done by creating the notion of *associative memory* of each input vector $\vec{u_i}$, denoted as $AM(\vec{u_i})$, and then projecting the association of $(AM(\vec{u_i}), \vec{v_i})$ through a binary mapping module. The $AM(\vec{u_i})$ for a given input pattern $\vec{u_i}$ is created based on the standard notion of Hamming distance by counting number of bits that differ between two binary vectors. Further setting a desired precision level ρ_i, a limit for allowing number of mismatch of bits with $\vec{u_i}$, the $AM_{\rho_i}(\vec{u_i})$ is created by accepting vectors differing from $\vec{u_i}$ within the range of the pre-fixed precision level ρ_i.

On the other hand, a binary mapping is simply a function $f_I : U \longrightarrow V$ from the domain $U = \{\vec{u_i} : 1 \leq i \leq k\}$ to the range set $V = \{\vec{v_i} : 1 \leq i \leq k\}$ such that $f_I(\vec{u_i}) = \vec{v_i}$. In order to make f_I be able to recall $\vec{v_i}$ for any pattern from $AM_{\rho_i}(\vec{u_i})$ it is extended to f_A where the domain $U_A = \cup_{i=1}^{k} AM_{\rho_i}(\vec{u_i})$. So, once through the binary mapping module f_I is extended to f_A by defining $f_A(\vec{u}) = \vec{v_i}$ if $\vec{u} \in AM_{\rho_i}(\vec{u_i})$ the next task leads towards tuning the perceptron model recognizing f_I in a way that it can recognize the extended function f_A. Here, the precision level ρ_i chosen for each input vector $\vec{u_i}$ in creating $AM_{\rho_i}(\vec{u_i})$ plays a crucial role as in order to ensure unambiguous recall of a particular output $\vec{v_i}$ for a noisy input pattern $\vec{u}$ the condition $AM_{\rho_i}(\vec{u_i}) \cap AM_{\rho_j}(\vec{u_j}) = \emptyset$ for any $i \neq j$, $1 \leq i, j \leq k$ is needed [3], [8].

Here a few points to be noticed. One is, by allowing a mismatch in the bits of $\vec{u_i}$ within a specified precision level ρ_i the point of noisy data, as mentioned in the above quote, is addressed. However, the aspect of recalling associative memory in the context of incomplete patterns of a given pattern is not paid a due attention to as according to their proposal two partial vectors only can be comparable using

Topical area: Advanced Artificial Intelligence in Applications

standard Hamming distance when the positions of missing data of both the vectors are the same. More specifically, two partial binary vectors $\langle 1, ?, 1, 0, 1\rangle$ and $\langle 1, 1, ?, 0, 1\rangle$, where ? denotes missing value, cannot be compared following [3]. However, as on the known bits positions $\{1, 4, 5\}$ both the vectors match, potentially they could represent the same pattern. The second issue is related to the condition $AM_{\rho_i}(\vec{u_i}) \cap AM_{\rho_j}(\vec{u_j}) = \emptyset$ for any $i \neq j$, $1 \leq i, j \leq k$ which is imposed in [3] in order to avoid multiple associative recall possibility for a noisy vector. For example following [3], both the above vectors are considered to be partial patterns of $\langle 1, 1, 1, 0, 1\rangle$ which means $\langle 1, ?, 1, 0, 1\rangle \odot \langle 1, 1, 1, 0, 1\rangle$ and $\langle 1, 1, ?, 0, 1\rangle \odot \langle 1, 1, 1, 0, 1\rangle$ hold. So, both can belong to $AM(\langle 1, 1, 1, 0, 1\rangle)$ with precision level 1 although they are not comparable with each other following standard Hamming distance. For reference the readers are referred to Figure 1 considered in [3] where the Hamming distance, denoted as HD, could not be computed for half-pattern. This seems a bit counter-intuitive as if we consider two partial patterns are equally potential to represent some fragments of a full image they might have some common fragments matching with each other. Thus, without having a possibility of comparing how two such partial patterns match to each other may lead to significant information gap. Moreover, the same partial pattern $\langle 1, ?, 1, 0, 1\rangle$ could be considered as a partial pattern of a full image $\langle 1, 0, 1, 0, 1\rangle$ and thus considering the lowest precision level 1, we would not manage to avoid the conflict $\langle 1, ?, 1, 0, 1\rangle \in AM(\langle 1, 1, 1, 0, 1\rangle) \cap AM(\langle 1, 0, 1, 0, 1\rangle)$. In [3], the non-overlap of associative memories of two input vectors is achieved by imposing specific precision levels ρ_i and ρ_j of two input vectors $\vec{u_i}$ and $\vec{u_j}$ in a way that $d_H(\vec{u_i}, \vec{u_j}) > \rho_i + \rho_j$. However, as d_H itself has some limitation in calculating distance between two partial vectors we opt for departure from the idea presented in [3]. The third point is concerning the method of preprocessing incomplete patterns before projecting them to a perceptron model. In [3] the authors did not account for incomplete patterns in the training sample. However, while extending the relation f_I from the input-output pairs $(\vec{u_1}, \vec{v_1})$ …, $(\vec{u_k}, \vec{v_k})$ of the training sample over the sets of associative patterns for the inputs $\vec{u_1}, \ldots, \vec{u_k}$, associative patterns are allowed to be incomplete. But before creating the perceptron model corresponding to f_A, the extension of f_I on incomplete patterns, the missing parts are imputed just by setting them to 0's. As mentioned in several papers [9], [10] setting missing parts of a partial pattern by 0's indicates ignoring the missing part while analyzing the data and thus leads to flaws in classification accuracy.

So, in this paper our aim is to address all these above mentioned conditions by some more general perspectives. Specifically, our first proposal is to define a distance function which can compare any two partial (possibly full) binary vectors resolving the first problem. The second consideration is to generalize the notion of 'partial pattern of a vector', denoted by $\odot$ by a more general notion and thus allowing the precision level for the associative memory of an input vector $\vec{u_i}$ to be selected keeping an eye on the whole data of input vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$. In regard to the third point, in our approach, firstly we allow incomplete patterns in the input training sample. Moreover, considering that data collected in a particular context might have a reconciliation possibility within themselves, we propose a way of matching partial input patterns within themselves in order to visualize if they reflect any possibility of reconciling full pattern(s) (see Section VI). The idea comes from the way of reconciling content of a full image from several of its parts as is done in jigsaw puzzles. Projection of the extended function f_A, in our context, in a perceptron model can be realized in the same way as in [3], [11].

The content of the paper is organized as follows. Section II reviews the basic details and mathematical model of associative memory proposed in [3], [8] and discusses about some limitations of the considered approach. Section III introduces a new perspective of measuring distance between two partial vectors following Hamming distance as well as presents a new notion of potential extension of partial Patterns. Moreover, it points out the departure from the similar notions proposed in [3], [8]. Section IV presents the essential conditions and results that a dataset needs to be satisfied for creating associative memory. Section V describes the data processing conditions, methods and iterations for creating associative memories of a given input set of partial vectors keeping the standard condition of avoiding multiple recall [3] in mind. Section VI sketches an idea for retrieving associative memories in the context of perceptron model. Section VII presents some concluding remarks.

II. Review of Related Research

To make the paper self-contained in this section let us present a summary of basic relevant details. As mentioned in Introduction the starting point is a data set containing a set of input-output pairs such as $(\vec{u_1}, \vec{v_1})$, $(\vec{u_2}, \vec{v_2})$, …, $(\vec{u_k}, \vec{v_k})$. Neural memories are generally of two kinds, namely *auto-associative memories* and *hetero-associative memories*. The aim of the auto-associative memories is to reconstruct a pattern from a noisy or partially specified pattern; that is here output vector corresponding to each input vector $\vec{u_i}$ is the same and target is to reconstruct $\vec{u_i}$ from $AM(\vec{u_i})$. On the other hand, hetero-associative memory aims to retrieve information concerning an input vector $\vec{u_i}$ given any instance from its $AM(\vec{u_i})$. The relation between k pairs of input-output vectors of dimension n and m respectively is realized through a 2-layer perceptron having the input layer comprising of n neurons, each of which connected to each of the k neurons lying in the hidden layer, and each of these k neurons are then connected to m output neurons. Clearly, n neurons at the input layer represent n-components of an input vector, say $\vec{u_i}$, m neurons at the output layer represent m components of an output vector, and each of the k neurons at the hidden layer represents k different input-output pairs. For more details the readers are referred to [3], [11].

A. Associative Memory

Now as described in Introduction to extend f_I from U to the sets of associative memories for each input vector of U, first $AM(\vec{u_i})$ is defined in the following manner.

For any $\vec{u_i} \in U$ and respective precision level ρ_i, $AM_{\rho_i}(\vec{u_i})$ $=\{\vec{u}|\vec{u} \in \mathbf{B}^n \& d_H(\vec{u}, \vec{u_i}) \leq \rho_i\}$[1] where $\mathbf{B}^n$ represents the set of all n-dimensional binary vectors and d_H represents the Hamming distance. Then f_I is extended over $\cup_{i=1}^{k} AM_{\rho_i}(\vec{u_i})$ in the following manner.

$f_A : \cup_{i=1}^{k} AM_{\rho_i}(\vec{u_i}) \rightarrow V$, $f_A(\vec{u}) = \vec{v_i}$; if $\vec{u} \in AM_{\rho_i}(\vec{u_i})$, $1 \leq i \leq k$. As mentioned before, the precision level ρ_i for each $\vec{u_i}$ is chosen such that $AM_{\rho_i}(\vec{u_i}) \cap AM_{\rho_j}(\vec{u_j}) = \emptyset$ for any $i \neq j$, $1 \leq i, j \leq k$.

As clearly visible that the above context of associative memory does not represent the possibility for including partial vectors with missing values at some of the components. In [3], [8] the idea of constructing associative memories for partially specified patterns is developed in the context of bipolar vectors where components of a vector may assume values from $\{-1, 0, 1\}$. However, 0 is used only as the marker for missing positions and any partial vector is supposed to assume values from only $\{1, -1\}$. Given any partial input vector $\vec{u_i}$ the missing positions of the vector are first padded with 0 so that the Hamming distance between two vectors can be computed. Then for two $\vec{u}$ and $\vec{v}$ the notion, viz., *$\vec{u}$ is a partial pattern of $\vec{v}$* is defined in the following way.

Definition II.1 ([3], [8])**.** *Given a partial bipolar vector $\vec{u}$, $\vec{u}$ is said to be a partial pattern of $\vec{v}$, denoted as $\vec{u} \odot \vec{v}$, if the available components of $\vec{u}$ are the same as that of the corresponding components of $\vec{v}$.*

The Hamming distance between two partial bipolar patterns is computed using the standard definition as $d_H(\vec{u}, \vec{v})$ is defined with the constraint that the unknown positions of $\vec{u}$ and $\vec{v}$ are the same. As a consequence after padding with 0's only the available positions with 1 and -1 would matter in counting mismatches between $pad0(\vec{u})$ and $pad0(\vec{v})$. Then the notion of associative memory in the context of partial vectors is defined in the following way.

Definition II.2 ([3], [8])**.** *(i) Given any input pattern $\vec{u_i}$, the set of all partial patterns of $\vec{u_i}$ with j-bits is given by* $U_i^j = \{\vec{u} : bits(\vec{u}) = j \ \& \ \vec{u} \odot \vec{u_i}\}$.
(ii) Further given a specified precision level ρ_i for $\vec{u_i}$, the associative memory for $\vec{u_i}$ with precision level ρ_i is defined as $AM_{\rho_i}^j = \{pad0(\vec{u'}) : \exists \vec{u} \in U_i^j \ \& \ d_H(\vec{u'}, \vec{u}) \leq \lfloor j/n \rfloor \times \rho_i\}$.

However, here a few points to be noted. The restrictions imposed above regarding only counting 1, -1 for the bit positions indicates that the purpose could have been served by binary vectors as well. Moreover, two vectors $\langle 1, -1, ?, -1, 1\rangle$ and $\langle 1, ?, 1, -1, 1\rangle$ intuitively may represent the same content as on the known positions they have the same values. As presented in [3], it seems that to claim $\vec{u} \odot \vec{u_i}$ the vector $\vec{u_i}$ must be a full vector. Thus, firstly for an input pattern $\vec{u_i}$ with missing values we cannot create $AM_{\rho_i}^j(\vec{u_i})$. Secondly, considering $\vec{u_i} = \langle 1, -1, 1, -1, 1\rangle$ we see both $\langle 1, -1, ?, -1, 1\rangle \odot \langle 1, -1, 1, -1, 1\rangle$ and $\langle 1, ?, 1, -1, 1\rangle \odot \langle 1, -1, 1, -1, 1\rangle$ hold. Thus, $\langle 1, -1, ?, -1, 1\rangle, \langle 1, ?, 1, -1, 1\rangle \in U_i^4$; but as, $d_H(\langle 1, ?, 1, -1, 1\rangle, \langle 1, -1, ?, -1, 1\rangle)$ is not defined both together cannot be considered in $AM_{\rho_i}^4(\langle 1, -1, 1, -1, 1\rangle)$, the set of associative patterns of $\langle 1, -1, 1, -1, 1\rangle$.

[1]Here to be noted that in [3] for this notion a different notation, namely $U_i^n(\rho_i)$ is used.

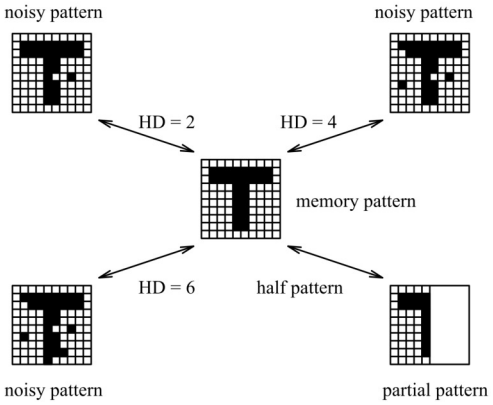

Figure 1. Computing Hamming distance(HD) for noisy patterns [3]

Keeping these limitations in mind our proposal, presented in the following sections, departs from a few aspects.

III. Hamming Distance and Potential Extensions of Partial Patterns

The aim of this section is to develop step-by-step the basic notions, departing from the respective notions proposed in [3], [8], so that the limitations discussed in Section II can be overcome. Our first target is to modify the definition for computing distance between two partial patterns in a way that instead of ignoring it would account for different unknown parts of the input patterns. In this regard, let us first present some notational details below.

Let $\vec{u}$ be any n-dimensional partial binary vector with some unknown components, denoted as $Unp(\vec{u})$; the set of known positions of $\vec{u}$ is denoted by $Bits(\vec{u})$, and $|Bits(\vec{u})|$ determines the number of bits in the vector, alternatively denoted by $bits(\vec{u})$. For example, for $\vec{u} = \langle ?, 1, 0, ?, 1, 0\rangle$ the $bits(\vec{u}) = 4$, $Bits(\vec{u}) = \{2, 3, 5, 6\}$, and $Unp(\vec{u}) = \{1, 4\}$.

A. Hamming distance for partial patterns

Let $\vec{u}$ and $\vec{u'}$ be two n-dimensional partial vectors having (possibly) some unknown components, denoted by $Unp(\vec{u})$ and $Unp(\vec{u'})$ respectively.

Definition III.1. *Given two partial binary vectors $\vec{u}$ and $\vec{u'}$ the Hamming distance between the vectors is defined as $D_H(\vec{u}, \vec{u'}) = d_H(\vec{u^t}, \vec{u^{t'}}) + l$, where $\vec{u^t}$ and $\vec{u^{t'}}$ are obtained by dropping all positions of $Unp(\vec{u}) \cup Unp(\vec{u'})$ from $\vec{u}$ and $\vec{u'}$ respectively, and $l = |Unp(\vec{u}) \cup Unp(\vec{u'})|$.*

It is clear from Definition III.1 that the modified notion of Hamming distance between two partial patterns considers a

pessimistic perspective by admitting the possibility that the missing positions of the vectors might have different content. The results below present in what sense it could be considered as a metric.

Proposition III.2. $D_H(\vec{u},\vec{v}) \leq D_H(\vec{u},\vec{w}) + D_H(\vec{w},\vec{v})$ *for any* $\vec{u},\vec{v},\vec{w}$

Based on Definition III.1, the proof relies on meticulously checking all different possibilities of unknown positions and their interrelations for three vectors $\vec{u},\vec{v},\vec{w}$.

Proposition III.3. D_H *satisfies the following properties of a metric.*

1) $D_H(\vec{u},\vec{v}) \geq 0$ *for any* $\vec{u},\vec{v}$.
2) $D_H(\vec{u},\vec{v}) = k$ *if and only if* $\vec{u^t} = \vec{v^t}$, $|Unp(\vec{u}) \cup Unp(\vec{v})| = k$.
3) $D_H(\vec{u},\vec{v}) = D_H(\vec{v},\vec{u})$ *for any* $\vec{u},\vec{v}$.

The proof of Proposition III.3 is straightforward from Definition III.1. From the property (2) of Proposition III.3, we can notice that $D_H(\vec{u},\vec{u}) = k$ given $|Unp(\vec{u})| = k > 0$. So, in contrary to standard metrics D_H is not reflexive. However, in the context of partial patterns it is quite intuitive that two partial patterns even may look the same on the known parts but there might be some mismatch in content in the undisclosed parts. Moreover, if we stick to the condition that $Unp(\vec{u}) = Unp(\vec{u'})$, then d_H can be considered as a lower estimation of D_H, and thus provides a more general perspective of calculating distance between two partial patterns.

Proposition III.4. *Given* $Unp(\vec{u}) = Unp(\vec{u'})$, $D_H(\vec{u},\vec{u'}) \geq d_H(\vec{u},\vec{u'}) + l$ *where* $|Unp(\vec{u})| = l$.

From Proposition III.4 it can be noticed that as the number of bits of $\vec{u}$ and $\vec{u'}$ increase D_H will be more close to the estimation proposed by d_H.

B. Potential extension for partial patterns

In this section we present a new perspective for the notion associative patterns by considering a modification of the same notion proposed in [3] from two aspects. Firstly, unlike $\odot$ we would introduce a new predicate $\otimes$ which incorporate the possibility of $\vec{u} \otimes \vec{v}$ to hold for a partial vector $\vec{v}$. Secondly, the notion of associative patterns developed based on $\otimes$ would allow two partial patterns $\vec{u}$ and $\vec{u'}$ as associative patterns of a given input pattern $\vec{u_i}$ even they do not share the same unknown parts.

Definition III.5. *Given a partial binary vectors* $\vec{u}$ *and any (partial) binary vector* $\vec{v}$, $\vec{u}$ *is said to be potentially extended to* $\vec{v}$, *denoted by* $\vec{u} \otimes \vec{v}$, *if* $Unp(\vec{u}) \supseteq Unp(\vec{v})$ *and values of all the known components of* $\vec{u}$ *match to that of the respective known components of* $\vec{v}$.

The Theorem below shows that $\otimes$ generalizes the notion of $\odot$.

Theorem III.6. *For a full vector* $\vec{v}$, $\vec{u} \odot \vec{v}$ *is a special case of* $\vec{u} \otimes \vec{v}$.

Proof. Let consider n-dimensional vectors $\vec{u},\vec{v},$. Let $\vec{u} \otimes \vec{v}$ where $Unp(\vec{v}) = \emptyset$ (The condition $Unp(\vec{u}) \supseteq Unp(\vec{v})$ for having $\vec{u} \otimes \vec{v}$ is trivially satisfied.) Now as $\vec{u} \otimes \vec{v}$, $d_H(\vec{u^t},\vec{v^t}) = 0$, where $\vec{u^t},\vec{v^t}$ are the vectors obtained by truncating $Unp(\vec{u})$ from both the vectors. Thus $\vec{v}$ is a full vector and $\vec{u}$ is a partial vector, s.t. $\forall l \in Bits(\vec{u})$, the l-th components of $\vec{u}$ and $\vec{v}$ are the same. $\therefore \vec{u} \odot \vec{v}$ □

Now, analogous to $U^j_{\rho_i}$ presented in Section II in our context for an input vector $\vec{u_i}$ we construct the following set.
$U_i^{bits(\vec{u_i})} = \{\vec{u} \in \mathbf{B}^n | 0 < bits(\vec{u}), \vec{u} \otimes \vec{u_i}\}, 1 \leq i \leq k$.
Considering $|Unp(\vec{u_i})| = p_i$ we can write $U_i^{bits(\vec{u_i})} = U_i^{n-p_i}$.

It is to be worth noticing that in $U_i^{n-p_i}$ we do not allow vectors with full unknown parts or in other words a vector with empty Bits.

IV. Data restriction in choosing precision level for associative memory

In [3], while developing binary mapping and respectively perceptron model for a set of input vectors the authors maintained the constraint of mutually disjoint associative memories for a set of input vectors by tuning the precision level ρ_i for the associative memory constructed for each input vector $\vec{u_i}$. In our work, as the notion of $U^j_{\rho_i}$ is replaced by the notion of $U_i^{n-p_i}$ where $|Unp(\vec{u_i})| = p_i$ to ensure mutually non-overlapping associative patterns for a given input set of vectors we propose to consider non-overlapping unknown components for each pair of vectors $\vec{u_i}$ and $\vec{u_j}$ in the input set of vectors.

Apart from the technical point of view of imposing $Unp(\vec{u_i}) \cap Unp(\vec{u_j}) = \emptyset$ for any $i \neq j$ and $1 \leq i,j \leq k$ there is an intuitive perspective of imposing this constraint as well. Often while solving a puzzle related to creating a known image, nowadays prevalently known as jigsaw puzzle[2], we are given a number of pieces each having different known fragments of the image visible. Usually, we aim to fit the unknown and known fragments of two pair of pieces in a way that some part of the whole image can be retrieved. For instance, given two partial input vectors $\vec{u_i} = \langle 1,0,?,0,1\rangle$ and $\vec{u_j} = \langle 1,?,1,?,1\rangle$, in one hand, merging them by replacing one's unknown components with others respective known components may help to visualize a complete pattern $\langle 1,0,1,0,1\rangle$. On the other hand, following the notion of $\otimes$ neither $\vec{u_i} \otimes \vec{u_j}$ nor $\vec{u_j} \otimes \vec{u_i}$ holds. That is, all the input vectors are considered to be independently indispensable to recover the full content of the actual image as neither of them can be potentially extended to the other following the notion of $\otimes$. Moreover, it would be clear in the latter sequels that the impact of this condition is not limited to the above discussed perspectives only.

A. Imposition of Restriction on Data

We start with imposing a few constraint on a data set of n-dimensional input vectors $\vec{u_1},\vec{u_2},\ldots,\vec{u_k}$.

[2]https://en.wikipedia.org/wiki/Jigsaw_puzzle

Definition IV.1 (Dset)**.** *A set of n-dimensional input vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ satisfying the following conditions is called a Dset.*

1) $0 \leq |Unp(\vec{u_i})| \leq \lfloor n/2 \rfloor$
2) $Unp(\vec{u_i}) \cup Bits(\vec{u_i}) = \{1, 2, \ldots, n\}$
3) $Unp(\vec{u_i}) \cap Unp(\vec{u_j}) = \emptyset$ *for* $1 \leq i, j \leq k, i \neq j$

The condition (1) imposes a restriction, may be called as Maximum Half Uncovered Pattern (MHUP). The condition (2) imposes that unknown and known part of each vector must exhaust the full pattern, may be called Exhaustive Pattern (EP). The condition (3) may be named as Mutually Disjoint Unknown Parts (MDUP).

A series of results below would clarify how the condition (MDUP) may determine the choice of precision level for a given input vector in the context of creating associative memory.

Proposition IV.2. *If $\vec{u} \in U_i^{n-p_i} \cap U_j^{n-p_j}$ then $Unp(\vec{u}) \supseteq Unp(\vec{u_i}) \cup Unp(\vec{u_j})$ and $d_H(\vec{u_i^t}, \vec{u_j^t}) = 0$ where $\vec{u_i^t}$ and $\vec{u_j^t}$ are obtained by dropping all positions of $Unp(\vec{u})$ from $\vec{u_i}$ and $\vec{u_j}$ respectively.*

Proof. $\vec{u} \in U_i^{n-p_i}$ and $\vec{u} \in U_j^{n-p_j}$; that is, $\vec{u} \otimes \vec{u_i}$ and $\vec{u} \otimes \vec{u_j}$. Let $\vec{u_i^t}$ and $\vec{u^t}$ be the vectors obtained by dropping the positions of $Unp(\vec{u})$ from both $\vec{u_i}$ and $\vec{u}$. So $Unp(\vec{u_i}) \subseteq Unp(\vec{u})$ and $d_H(\vec{u_i^t}, \vec{u^t}) = 0$ as well as $Unp(\vec{u_j}) \subseteq Unp(\vec{u})$ and $d_H(\vec{u_j^t}, \vec{u^t}) = 0$. □

The above proposition leads to the following corollary.

Corollary IV.3. *If $\vec{u} \in U_i^{n-p_i} \cap U_j^{n-p_j}$, then $p_i + p_j \leq p$.*

Proof. As $Unp(\vec{u_i}) \cap Unp(\vec{u_j}) = \emptyset$, then by Proposition IV.2, $Unp(\vec{u_i}) \cup Unp(\vec{u_j}) \subseteq Unp(\vec{u})$ which implies $p_i + p_j \leq p$. □

Combining Proposition IV.2 and Corollary IV.3 using contraposition we can say that if for a vector $\vec{u}$ with $|Unp(\vec{u})| = p$, $p < p_i + p_j$ or $d_H(\vec{u_i^t}, \vec{u_j^t}) \neq 0$, then $\vec{u} \notin U_i^{n-p_i} \cap U_j^{n-p_j}$, where $\vec{u_i}^t$, $\vec{u_j}^t$ are obtained from $\vec{u_i}$, $\vec{u_j}$ respectively by truncating respective positions of $Unp(\vec{u})$.

So, from Corollary IV.3 it is clear that if two input vectors are not having common unknown parts then that suffices to choose a precision level for a vector $\vec{u}$ and ensure that $\vec{u}$ cannot be regarded as a common associative pattern for both the input vectors. However, this is not a necessary condition ensuring $U_i^{n-p_i} \cap U_j^{n-p_j} = \emptyset$. The sequel below will throw some light in this regard.

Proposition IV.4. *If $Unp(\vec{u_i}) \cap Unp(\vec{u_j}) = \emptyset$, then $U_i^{n-p_i} \neq U_j^{n-p_j}$ given $p_i \geq 2$ or $p_j \geq 2$.*

Proof. Let $Unp(\vec{u_i}) = \{r_1, r_2, \ldots, r_{p_i}\}$, $Unp(\vec{u_j}) = \{t_1, t_2, \ldots, t_{p_j}\}$ where $Unp(\vec{u_i}) \cap Unp(\vec{u_j}) = \emptyset$ and $p_j \geq 2$. Now, consider $\vec{u}$ with $Unp(\vec{u}) = \{r_1, r_2, \ldots, r_{p_i}\} \cup \{t_l\}$ where $t_l \in Unp(\vec{u_j})$ and $D_H(\vec{u_i}^t, \vec{u}^t) = 0$ where $\vec{u_i}^t$, $\vec{u}^t$ are obtained by truncating all the positions of $Unp(\vec{u})$ from both the vectors. Clearly, $\vec{u} \otimes \vec{u_i}$ and thus $\vec{u} \in U_i^{n-p_i}$. But as $Unp(\vec{u}) \not\supseteq Unp(\vec{u_j})$, $\vec{u} \notin U_j^{n-p_j}$.

In the similar fashion, for $p_i \geq 2$ we can create a vector $\vec{u'}$ such that $\vec{u'} \in U_j^{n-p_j}$ but $\vec{u'} \notin U_i^{n-p_i}$. Thus, we can claim that $U_i^{n-p_i} \neq U_j^{n-p_j}$. □

Proposition IV.4 indicates that by considering non-overlapping unknown parts in the input data and putting the restriction that at most one input vector in the input data may have exactly one unknown component, we will be able to ensure that the set of associative patterns for each input vector will be distinct. However, the condition does not ensure that $U_i^{n-p_i} \cap U_j^{n-p_j} \neq \emptyset$. The example below illustrates the claim.

Example IV.1. *Let us consider $Unp(\vec{u_i})$ and $Unp(\vec{u_j})$ as the same as in Proposition IV.4 with an additional condition that $Unp(\vec{u_i}) \cup Unp(\vec{u_j}) \neq \{1, 2, \ldots, n\}$. Now consider a vector $\vec{u}$ with $Unp(\vec{u}) \supseteq Unp(\vec{u_i}) \cup Unp(\vec{u_j})$ and $D_H(\vec{u}^t, \vec{u_i}^t) = D_H(\vec{u}^t, \vec{u_j}^t) = 0$ where $\vec{u}^t, \vec{u_i}^t, \vec{u_j}^t$ are obtained respectively from the original vectors after removing the components of $Unp(\vec{u})$. This clearly indicates that $\vec{u} \otimes \vec{u_i}$ and $\vec{u} \otimes \vec{u_j}$ and thus $\vec{u} \in U_i^{n-p_i} \cap U_j^{n-p_j}$.*

In [3], the associative patterns for the set of input vectors are imposed to be non-overlapping. The reason behind such consideration is already explained in Section II. The above results indicate that, in our context, in order to avoid overlap in the respective associative patterns for the input vectors we may need a further pre-processing phase for data.

Based on Corollary IV.3, we can notice that given any two input vectors $\vec{u_i}$ and $\vec{u_j}$ if we choose a vector $\vec{u}$ with $Unp(\vec{u}) = p$ such that $p < p_i + p_j$ then $\vec{u} \notin U_i^{n-p_i} \cap U_j^{n-p_j}$. That is, to avoid overlap in the respective associative patterns of any two input vectors of a data set $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ we can impose the following condition.

Condition IV.5. *Given a data set of input vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$, for any $\vec{u_i}$, $1 \leq i \leq k$, the precision level ρ_i for the set of associative patterns of $\vec{u_i}$, denoted by $AM(\vec{u_i})$, must satisfy $p_i < \rho_i < p_i + p_j$ for any $j \neq i$.*

Here a natural question arises that whether given an input data we will always have a situation to choose a precision level for an arbitrary input vector following Condition IV.5. Following proposition would clarify the issue further.

Proposition IV.6. *Given the input data $\vec{u_1}, \vec{u_2}, ..., \vec{u_k}$ if for a vector $\vec{u_i}$, $1 \leq i \leq k$, there is no $\rho_i \in \mathbb{Z}^+$ s.t. $p_i < \rho_i < p_i + p_j$ for any $i \neq j$, then there is at least one $j \neq i$ for which $p_j = 1$.*

Proof. From the assumption that there is no $\rho_i \in \mathbb{Z}^+$, s.t. $p_i < \rho_i < p_i + p_j$ for all $i \neq j$ it is clear that $p_i + p_j$ must be equal to $p_i + 1$ for some $j \therefore p_j = 1$ for some j. □

Proposition IV.7. *If $p_j = 1$ for some $j \neq i$, then there is no $\rho_i \in \mathbb{Z}^+$ satisfying $p_i < \rho_i < p_i + p_j$ for $i \neq j$.*

Proof. Let $p_j = 1$ for some $j \neq i$, and there exist a $\rho_i \in \mathbb{Z}^+$ satisfying $p_i < \rho_i < p_i + p_j$ for $i \neq j$. As $p_j = 1$ for some $j \neq i$, we can write $p_i < \rho_i < p_i + 1$ and $0 < \rho_i - p_i < 1$. This is a contradiction as $p_i, \rho_i \in \mathbb{Z}^+$. Hence there is no $\rho_i \in \mathbb{Z}^+$ satisfying $p_i < \rho_i < p_i + p_j$ for $i \neq j$. □

Clearly, if we have a data set of input vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ such that $|Unp(\vec{u_i})| = p_i > 1$ for each $i \in \{1, 2, \ldots, k\}$, then following Condition IV.5 finding a precision level $\rho_i \in \mathbb{Z}^+$ for each input vector $\vec{u_i}$ is possible. This result also goes in accordance with our observation from Proposition IV.4. So, allowing only input vectors with more than one unknown components may help to create a unique precision level for the set of associative patterns for each input vectors following Condition IV.5. However, we need to dig further to check how such a condition would impact on the availability of sufficient amount of input data.

B. Impact of data restriction on availability of data

Given the restriction on data imposed in Definition IV.1 and the additional restriction mentioned in Condition IV.5 it is now important to check how much data we may have to train a required perceptron model. In this regard, let us first consider the context of a data set satisfying the conditions (MHUP) $bits(\vec{u_i}) \geq \lceil n/2 \rceil$ and (EP) ($Bits(\vec{u_i})$, $Unp(\vec{u_i})$) forms a partition of $\{1, 2, \ldots n\}$, mentioned in Definition IV.1. Before presenting the result let us define some useful notations.

Notation IV.8. (i) $S_n(bits) = \{X \subseteq \{1, 2, ..., n\} : |X| \geq \lceil n/2 \rceil\}$.
(ii) $S^i(Y) = \{X \subseteq Y : 0 \leq |X| < i\}$

Given the above notations we can notice that $S_n(bits) = P(\{1, 2, ..., n\}) - S^{\lceil n/2 \rceil}(\{1, 2, ..., n\})$.

Proposition IV.9. *$|S_n(bits)| = 2^n - \sum_{i=0}^{m-1} {}^nC_i$, where $n = 2m$ or $n = 2m - 1, m \geq 1$.*[3]

Proof. We know $|P(\{1, 2, ..., n\})| = 2^n$. Now for $S^{\lceil n/2 \rceil}(\{1, 2, ..., n\})$, we have two cases (1) $n = 2m$ and (2) $n = 2m - 1$
(1) If $n = 2m, \lceil n/2 \rceil = m$, then $S^m(\{1, 2, ..., n\}) = \sum_{i=0}^{m-1} {}^{2m}C_i$
(2) If $n = 2m - 1, \lceil n/2 \rceil = m$, then $S^m(\{1, 2, ..., n\}) = \sum_{i=0}^{m-1} {}^{2m-1}C_i$
Therefore, clearly from (1) and (2), $|P(\{1, 2, ..., n\})| - |S^{\lceil n/2 \rceil}(\{1, 2, ..., n\})| = 2^n - \sum_{i=0}^{m-1} {}^nC_i$, where $n = 2m$ or $n = 2m - 1$, $m \geq 1$. □

Proposition IV.10. *$2^n - \sum_{i=0}^{m-1} {}^nC_i = \sum_{i=0}^{\lfloor n/2 \rfloor} {}^nC_i$, where $n = 2m$ or $n = 2m - 1$, $m \geq 1$*

Proof. It is well known $2^n = \sum_{i=0}^{n} {}^nC_i$. Now we need to show $\sum_{i=0}^{n} {}^nC_i = \sum_{i=0}^{m-1} {}^nC_i + \sum_{i=0}^{\lfloor n/2 \rfloor} {}^nC_i$
(1) when $n = 2m, \lceil n/2 \rceil = m$ and $\lfloor n/2 \rfloor = m$
LHS = ${}^{2m}C_0 + {}^{2m}C_1 + {}^{2m}C_2 + ... + {}^{2m}C_{2m}$
RHS[4] = ${}^{2m}C_0 + {}^{2m}C_1 + ... + {}^{2m}C_{m-2} + {}^{2m}C_{m-1} + {}^{2m}C_0 + {}^{2m}C_1 + ... + {}^{2m}C_{m-1} + {}^{2m}C_m$
$= {}^{2m}C_0 + {}^{2m}C_1 + ... + {}^{2m}C_{m-2} + {}^{2m}C_{m-1} + {}^{2m}C_m + {}^{2m}C_{m+1} + ... + {}^{2m}C_{2m-1} + {}^{2m}C_{2m}$
$\therefore$ LHS=RHS

(2) when $n = 2m - 1, \lceil n/2 \rceil = m$ and $\lfloor n/2 \rfloor = m - 1$
LHS = ${}^{2m-1}C_0 + {}^{2m-1}C_1 + {}^{2m-1}C_2 + ... + {}^{2m-1}C_{2m-1}$
RHS = ${}^{2m-1}C_0 + {}^{2m-1}C_1 + ... + {}^{2m-1}C_{m-2} + {}^{2m-1}C_{m-1} + {}^{2m-1}C_0 + {}^{2m-1}C_1 + ... + {}^{2m-1}C_{m-1} + {}^{2m-1}C_m$
$= {}^{2m-1}C_0 + {}^{2m-1}C_1 + ... + {}^{2m-1}C_{m-2} + {}^{2m-1}C_{m-1} + {}^{2m-1}C_m + {}^{2m-1}C_{m+1} + ... + {}^{2m-1}C_{2m-2} + {}^{2m-1}C_{2m-1}$
$\therefore$ LHS=RHS

□

So, combining Propositions IV.9, IV.10 we can say that $|S_n(bits)| = \sum_{i=0}^{\lceil n/2 \rfloor} {}^nC_i$. That is, for 10-dimensional vectors there can be maximum 638 vectors satisfying (MHUP) and (EP) mentioned in Definition IV.1. So, the next question arises that how many vectors out of $|S_n(bits)|$ number of possible vectors can have non-overlapping unknown parts. In this regard, let us consider any input vector $\vec{u_i}$ with $bits(\vec{u_i}) = r$ where $\lceil n/2 \rceil \leq r \leq n$.

Proposition IV.11. *Given an input vector $\vec{u_i}$ with $bits(\vec{u_i}) = r$, where $\lceil n/2 \rceil \leq r \leq n$, the maximum number of vectors in $\boldsymbol{B}^n$ having disjoint unknown parts with $Unp(\vec{u_i})$ is ${}^rC_{n-r}$.*

Proof. Let $\vec{u_i}$ be a vector with $Bits(\vec{u_i}) = \{s_1, s_2, \ldots, s_r\}$ where $\lceil n/2 \rceil \leq r \leq n$ and $|Unp(\vec{u_i})| = n - r$. Following our condition $r \geq n - r$. Then in order to have other r-bits vectors with non-overlapping unknown components we can consider a vector $\vec{u_i}'$ whose $Unp(\vec{u_i}')$ is obtained by exchanging all $n-r$ positions of $Unp(\vec{u_i})$ with positions from $\{s_1, s_2, \ldots, s_r\}$ of $\vec{u_i}$. This allows us to create ${}^rC_{n-r}$ number of different vectors with r-bits having disjoint unknown parts with $\vec{u_i}$. □

Example IV.2. *Following Proposition IV.11, for $n = 10$, we can allow input vectors with number of bits ranging from 5 to 10. Given an input vector $\vec{u_1}$ with $|Unp(\vec{u_1})| = 5$ there can be exactly 1 possible 5-bit vector, say $\vec{u_{11}}$, having non-overlapping unknown parts with $\vec{u_1}$. For an input vector $\vec{u_2}$ with $|Unp(\vec{u_2})| = 4$ there can be exactly 15 possible 6-bit vectors having non-overlapping unknown parts with $\vec{u_2}$. However, these 15 vectors may have common unknown parts within themselves. Thus, at each turn we can select exactly one vector, say $\vec{u_{21}}$, from these 15 possible cases satisfying the condition of non-overlapping unknown components with $\vec{u_2}$. For an input vector $\vec{u_3}$ with $|Unp(\vec{u_3})| = 3$ there can be exactly 35 possible 7-bit vectors having non-overlapping unknown parts with $\vec{u_3}$. So, in one round of selection out of 35 possibilities we can select a pair of vectors, say $\vec{u_{31}}, \vec{u_{32}}$, such that $\vec{u_3}$ and these two vectors satisfy mutually disjoint unknown parts. For an input vector $\vec{u_4}$ with $|Unp(\vec{u_4})| = 2$ there can be exactly 28 possible 8-bit vectors having non-overlapping unknown parts with $\vec{u_4}$. In this case, out of 28 possibilities we can select 4 vectors, say $\vec{u_{41}}, \vec{u_{42}}, \vec{u_{43}}, \vec{u_{44}}$ such that along with $\vec{u_4}$ they satisfy mutually disjoint unknown parts. For an input vector $\vec{u_5}$ with $|Unp(\vec{u_5})| = 1$ there can be exactly 9 possible 9-bit vectors having non-overlapping unknown parts with $\vec{u_5}$ and all of them can be considered together in one round of selection as they satisfy mutually disjoint unknown parts.*

[3] $n = 2m$ or $n = 2m + 1$, is also possible.
[4] LHS and RHS are respectively the abbreviations for Left Hand Side and Right Hand Side of a relational equation.

From the above clarification, two important aspects are surfacing out. One is the chosen number of unknown components for the vectors of a particular dimension. If half of the components are unknown the number of different vectors with non-overlapping unknown parts is very less, and when the number of unknown components is decreasing possible number of different vectors satisfying (MHUP) and (EP) is increasing. However, as it is visible from the example that there seems to be a threshold value for number of unknown components, which is here 3, after which the number of possible vectors with a specified number of unknown components is decreasing. On the other hand, even for a vector $\vec{u_i}$ with $n - r$ ($\leq \lfloor n/2 \rfloor$) number of unknown components not all of the ${}^{r}C_{n-r}$ number of possible vectors having non-overlapping unknown parts with $\vec{u_i}$ can be considered together as they would not satisfy (MDUP). So, in order to train a perceptron model if we wish to use the available data as much as possible satisfying the conditions of a Dset, we may need to split the data over several iterations and learn separately the perceptron model generated in each iteration.

V. Constructing Associative Memory and Splitting Data Over Iterations

Given the above analysis about the number of possible input vectors of dimension n satisfying the conditions of Definition IV.1 and the condition concerning precision level of each input vector (see Proposition IV.6) let us introduce the following method ensuring non-overlapping sets of associative patterns for a given Dset.

Definition V.1 (M-AM)**.** *Let a Dset containing k number of n-dimensional input vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ be given. The method, called as M-AM, of creating the associative memory for each input vector is described as follows.*

1) *For each i, if the precision level ρ_i is s.t. $p_i < \rho_i < p_i + p_j$ for any $j \neq i$ holds, then consider $U_{\rho_i} = \{\vec{u} \in U_i^{n-p_i} : |Unp(\vec{u})| \leq \rho_i\}$ as $AM(\vec{u_i})$.*
2) *If for a vector $\vec{u_i}$ there is no ρ_i satisfying Condition IV.5, then follow the following process of modifying U_{ρ_i}.*
   a) *Find $O_i = \{l \mid p_i + p_l = min_j\ p_i + p_j,\ j \neq i\}$*
   b) *Consider $\rho_i = p_i + p_l$*
   c) *$U_{\rho_i}^{mod} = \{\vec{u} \in U_{\rho_i} : Unp(\vec{u}) \not\supseteq Unp(\vec{u_l}),\ l \in O_i\}$ and consider $U_{\rho_i}^{mod}$ as $AM(\vec{u_i})$.*

The method M-AM describes an algorithm for determining a precision level for each input vector keeping an eye on the whole input data and accordingly creating an associative memory for each input vector of a given data set $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$. However, at this stage our next concern is to check what amount of associative patterns can be generated for each input vector following the method M-AM.

Proposition V.2. *Given input vectors $\vec{u_1}, \vec{u_2}, ..., \vec{u_k}$, for an arbitrary $\vec{u_i}$, if the precision level ρ_i is s.t. $p_i < \rho_i < p_i + p_j$ for any $j \neq i$ holds, then $|U_{\rho_i}| = {}^{n-p_i}C_{\rho_i - p_i}$*

Proof. According to M-AM for any $\vec{u} \in U_{\rho_i}$, $Unp(\vec{u}) \supseteq Unp(\vec{u_i})$, $|Unp(\vec{u})| \leq \rho_i$ where $p_i < \rho_i < p_i + p_j$, any $i \neq j$. So, for any $\vec{u} \in U_{\rho_i}$ $Unp(\vec{u})$ contains $Unp(\vec{u_i})$ and any $|Unp(\vec{u}) - Unp(\vec{u_i})|$ = ρ_i - p_i positions out of $|\{1, 2, \ldots, n\} - Unp(\vec{u_i})|$ = n-p_i positions of $Bits(\vec{u_i})$. Thus, $|U_{\rho_i}| = {}^{n-p_i}C_{\rho_i - p_i}$. □

Proposition V.3. *Given input vectors $\vec{u_1}, \vec{u_2}, ..., \vec{u_k}$, if for an arbitrary $\vec{u_i}$, there is no ρ_i s.t. $p_i < \rho_i < p_i + p_j$ for any $j \neq i$, then $|U_{\rho_i}^{mod}| = {}^{n-(p_i+|O_i|)}C_1$ where $O_i = \{L : p_i + p_L = min_j\ p_i + p_j, j \neq i\}$*

Proof. Clearly, $U_{\rho_i}^{mod} \subseteq U_{\rho_i}$ and as there is some $j \neq i$ for which p_j = 1, $\rho_i = p_i + 1$. So, following Proposition V.2 $\rho_i - p_i = 1$. Moreover, while creating $U_{\rho_i}^{mod}$ as we are considering that for any $u \in U_{\rho_i}^{mod}$ if $l \in O_i$ then $l \notin Unp(\vec{u})$, we also need to remove from $\{1, 2, \ldots, n\}$ all unknown positions of the vectors $\vec{u_l}$ such that $l \in O_i$. So, in contrary to Proposition V.2 here we have $n - (p_i + |O_i|)$ remaining positions from where $\rho_i - p_i$ that is 1 position can be chosen to be included in $Unp(\vec{u})$. So, we have ${}^{n-(p_i+|O_i|)}C_1$ many ways to create $Unp(\vec{u})$ by adding one additional position to $Unp(\vec{u_i})$. Hence $|U_{\rho_i}^{mod}| = {}^{n-(p_i+|O_i|)}C_1$. □

Following Proposition V.3, we can see that the more input vectors with unknown components equal to 1, i.e., $|O_i|$ increases, the number of vectors belonging to $U_{\rho_i}^{mod}$ decreases. For this reason, we should avoid having a large number of input data consisting of just one unknown component vector.

A. Stepwise Unknown Full Cover Input Data

Following Proposition V.3, we should avoid a set of input vectors such that each has only one unknown component. The sequel below will make the point more precise.

Proposition V.4. *Following M-AM for any n number of n-dimensional vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_n}$, if $AM(\vec{u_i}) \supsetneq \{\vec{u_i}\}$ then it can not be the case that $p_1 = p_2 = \cdots = p_n = 1$.*

Proof. Contrapositively, if for a data set satisfying conditions of M-AM we assume $p_1 = p_2 = \cdots = p_n = 1$, then for any $\vec{u_i}$ by Proposition IV.7 there is no ρ_i such that $p_i < \rho_i < p_i + p_j$ for any $j \neq i$, $i, j \in \{1, 2, \ldots, n\}$. Thus,$AM(\vec{u_i}) = U_{\rho_i}^{mod}$. But $O_i = \emptyset$. Hence, $AM(\vec{u_i}) = \{\vec{u_i}\}$. □

Definition V.5 (SUFC)**.** *Given a set of n-dimensional input vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ with $|Unp(\vec{u_i})| = p_i$ for $1 \leq i \leq k$, and $p_1 = p_2 = \cdots = p_k = 1$ if $\cup_{i=1}^{k} Unp(\vec{u_i}) = \{1, 2, \ldots, n\}$ and $Unp(\vec{u_i}) \cap Unp(\vec{u_j}) = \emptyset$ for any $i \neq j$, then we call such data set as Stepwise Unknown Full Cover (SUFC).*

Thus, along with conditions mentioned in Definition IV.1 let us add another additional condition on the Dset.

Condition V.6. *A Dset of input vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ should not satisfy SUFC.*

Based on the above discussion it is clear that apart from the conditions mentioned in Definition IV.1 we need some other conditions to be imposed in order to follow method M-AM in the process of creating associative memory for a Dset. Thus, below we present a modified notion of Dset.

Definition V.7 (Dset$_{AM}$)**.** *Given a set of input vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ we say it as a Dset$_{AM}$, if it is a Dset and satisfies Condition V.6.*

B. Processing Input Data Through Iterations

Based on the results and discussion presented in the above sections, we can notice a few important aspects for effectively using a given data in the process of creating associative patterns and then learning perceptron models based on them. The first is that, given a Dset of n-dimensional input vectors to apply method M-AM for creating associative memories for input vectors we may need to split the Dset over different iterations. Secondly, in order to avoid overlapping sets of associative patterns for all the vectors considered in a particular iteration the input vectors considered in each iteration should form a Dset$_{AM}$. Thirdly, based on Proposition IV.11 and Example IV.2 we recognize a need for selecting a Dset$_{AM}$ for each iteration satisfying a *top down decomposition ordering*, described below, by allowing gradually less unknown parts (or gradually more known parts) in the selection of input vectors.

C. Top-down Decomposition Ordering

We propose a method for ordering the input data to be selected in a $Dset_{AM}$ in such a way that it considers vectors with gradually increasing order of unknown parts while selecting for a $Dset_{AM}$ from a given data.

Definition V.8 (TDDO)**.** *Let us consider a set of input vectors, $In = \{\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}\}$ of dimension n, where $n > 3$.* [5] *A Top-Down Decomposition Ordering of input vectors is composed of the following steps.*

1) *Select a vector $\vec{u_j}$ ($1 \leq j \leq k$) such that $|Unp(\vec{u_j})|$ is minimum. Call the vector $\vec{u_{j1}}$.*
2) *Select the next vector from $In \setminus \{\vec{u_j}\}$ such that the number of unknown components of the selected vector is $|Unp(\vec{u_j})|$+1 and the unknown components of the new vector does not have common overlap with $Unp(\vec{u_j})$. Call the new vector $\vec{u_{j2}}$.*
3) *Continue the process as long as there is no vector from the remaining set of In satisfying condition (2).*

Example V.1. *Given a input data $In = \{\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}\}$ for $n = 10$. Following Definition V.8 consider input vectors from In such that $\vec{u_{j1}}$ with $|Unp(\vec{u_{j1}})| = 1$, $\vec{u_{j2}}$ with $|Unp(\vec{u_{j2}})| = 2$, $\vec{u_{j3}}$ with $|Unp(\vec{u_{j3}})| = 3$ and $\vec{u_{j4}}$ with $|Unp(\vec{u_{j4}})| = 4$, and they satisfy (MDUP). Clearly, for $n = 10$, there are no more than 4 vectors that have no common overlapping unknown components covering $\{1, 2, \ldots, 10\}$. Thus, an iteration consist of this set of vectors $\{\vec{u_1}, \vec{u_2}, \vec{u_3}, \vec{u_4}\}$.*

From Example V.1 we can see that following TDDO if we select input vectors, starting from lowest number of unknown components, in a strict increasing manner depending on the dimension of the vector there seems to be a threshold value for the number of unknown components. For instance, in case of Example V.1 for 10 dimensional vectors this threshold value is 4. The sequel below throws light on this.

Definition V.9 (Threshold and cardinality value for TDDO)**.** *Let a set of input vectors of n-dimension ($n > 3$) be given. To create a Dset$_{AM}$ for one iteration we first select a vector with exactly one unknown component and continue selecting a sequence of vectors following TDDO. When the process reaches to stage 3 of TDDO (see Definition V.8) the maximum number of unknown components of the last selected vector in Dset$_{AM}$ is considered to be the threshold value for TDDO for dimension n, denoted by T_n, and the cardinality of the Dset thus constructed is called the cardinality value for Dset$_{AM}$, denoted as $|Dset_{AM}|$.*

Following Definition V.9 it is clear that for an n-dimensional input vectors once we reach to the stage (3) of TDDO and get hold of a vector reaching threshold T_n we cannot include any vector in the respective Dset$_{AM}$ having more than T_n number of unknown components. So, one important question is how to determine the threshold T_n for any arbitrary dimension n.

Theorem V.10. *Given a set of input vectors of dimension n let T_n and $|Dset_{AM}|$ be respectively the threshold value and cardinality value of the Dset$_{AM}$ obtained following TDDO corresponding to an iteration. Then considering $|Dset_{AM}| = i$ the following relations hold between T_n and i.*

1) *If $n = 2, 3$, $T_n = 1$.*
2) *For any $n > 3$ and $i \geq 2$, $\frac{i(i+1)}{2} \leq n \leq \frac{i(i+1)}{2} + i$ holds and for all such n, $T_n = i$.*

Proof. For $n = 2, 3$ the case is straightforward.

For any n ($n > 3$) the increasing order of unknown components of the vectors selected in $Dset$ are $1, 2, \ldots T_n$. So, $|Dset_n| = T_n = i$. Following TDDO, $n - (1 + 2 + \cdots + i) \leq i$.
$\therefore n \leq \frac{i(i+1)}{2} + i$ and clearly for $n > 3$, the value i must satisfy $i \geq 2$.
Moreover, total number of positions covered by the unknown parts of the vectors of $Dset_n$ cannot exceed n. $\therefore \frac{i(i+1)}{2} \leq n$
$\therefore \frac{i(i+1)}{2} \leq n \leq \frac{i(i+1)}{2} + i$, for all $n > 3$ and $i \geq 2$.
Moreover, as for any $n \in [\frac{i(i+1)}{2}, \frac{i(i+1)}{2} + i]$ after covering gradually $1, 2, \ldots, i$ number of unknown positions, i.e., total $\frac{i(i+1)}{2}$ positions out of n dimension, there can be at most i positions remaining to be considered as unknown part of a vector. Thus, the threshold value T_n for all such n would be the same as i. □

Example V.2. *Following Theorem V.10, let us consider that $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ be a set of input vectors of dimension $n > 3$. For $i = 2$ we see that 4, $5 \in [\frac{i(i+1)}{2}, \frac{i(i+1)}{2} + i]$ and $T_n = 2$; that is following TDDO there cannot be any vector of more than 2 unknown parts in a $Dset_{AM}$. For $i = 3$, $n \in [6, 9]$ and for any n lying in this interval $T_n = 3$ as the gradual increasing order of the number of unknown components of any vector entitled to enter in a $Dset_{AM}$ of an iteration are 1, 2, 3. Similarly, for $i = 4$, $[\frac{i(i+1)}{2}, \frac{i(i+1)}{2} + i] = [10, 14]$ and*

[5] Following the condition that for a n-dimensional vector maximum number unknown components can be $\lfloor n/2 \rfloor$, for a vector of dimension one would not have any unknown component, and for a vector of dimension 2 or dimension 3 can have exactly one unknown component. So, TDDO does not apply to the vectors with dimension $n \leq 3$.

for any $n \in [10, 14]$ the threshold value for maximum unknown part following TDDO is $T_n = 4$.

D. Splitting data over iterations

As it is clear from previous sections, for n-dimensional vectors satisfying first two conditions mentioned in Definition IV.1 we may have $\sum_{i=0}^{\lfloor n/2 \rfloor} {}^nC_i$ of vectors in the input data In. However, following method M-AM in one iteration we can use only one small fragment of of data from In using TDDO. So, below we propose a complete procedure for using the data of In over some iterations.

Definition V.11 (SDOI)**.** *For a input data In containing n-dimensional vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$ by Splitting Data Over Iterations (SDOI) we mean the following procedure. In the first iteration, say I_0, we choose the vectors for $DSet^0_{AM}$ as follows.*

(i) *Select the first appeared vector $\vec{u_j} \in In$ with minimum $|Unp(\vec{u_j})|$ ($\leq T_n$) and rename it $\vec{u_{j1}}$*

(ii) *Following TDDO select vectors from $In \setminus \{\vec{u_j}\}$ with gradually increasing order of unknown components (considering first such appeared in input data) and continue till it reaches the threshold value T_n*

(iii) *Step (ii) ends when at most T_n vectors, say $\vec{u_{j1}}, \vec{u_{j2}} \ldots, \vec{u_{jT_n}} \in Dset^0_{AM}$. For the remaining $n - \frac{T_n(T_{n+1})}{2}$ positions select vectors from $In \setminus Dset^0_{AM}$ such that the gradual selection of the vectors satisfy monotonically decreasing order of unknown parts (satisfying Definition IV.1) as long as $\cup_{\vec{u} \in Dset^0_{AM}} Unp(\vec{u})$ = $\{1, 2, \ldots, n\}$.*

(iv) *Once Step (iii) ends, a new iteration of creating next $Dset_{AM}$ starts from step (i) over the remaining data $In \setminus Dset^0_{AM}$.*

(v) *Step (iv) continues as long as it is possible to create a new $Dset_{AM}$ following steps (i)-(iii).*

Example V.3. *For $n = 14$, in the iteration I_0 we may have $Dset_{AM} = \{\vec{u_1}, \vec{u_2}, \vec{u_3}, \vec{u_4}, \vec{u_5}, \vec{u_6}\}$ where $|Unp(\vec{u_i})| = i$ for $1 \leq i \leq 4$, and after reaching $T_{14} = 4$, $|Unp(\vec{u_5})| = 3$ and $|Unp(\vec{u_6})| = 1$. There can be several possibilities of constructing such a $Dset_{AM}$. For instance, one such is $Unp(\vec{u_1}) = \{1\}$, $Unp(\vec{u_2}) = \{2, 3\}$, $Unp(\vec{u_3}) = \{4, 5, 6\}$, $Unp(\vec{u_4}) = \{7, 8, 9, 10\}$, $Unp(\vec{u_5}) = \{11, 12, 13\}$, and $Unp(\vec{u_6}) = \{14\}$.*

Now let us present another example in order to understand how the associative memories for a given set of input vectors of a specific dimension n can be constructed following M-AM method and the data selection method described in Definition V.11.

Example V.4. *Given a input data In containing 8-dimensional vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_k}$. Following Definitions V.1, V.8, V.9 V.11 consider the input vectors from In such that $\vec{u_1}$ with $Unp(\vec{u_1}) = \{1\}$, $\vec{u_2}$ with $Unp(\vec{u_2}) = \{2, 3\}$, $\vec{u_3}$ with $Unp(\vec{u_3}) = \{4, 5, 6\}$, [6] $\vec{u_4}$ with $Unp(\vec{u_4}) = \{7, 8\}$, that is, $I_0 = \{\vec{u_1}, \vec{u_2}, \vec{u_3}, \vec{u_4}\}$. We can check that the precision level for $\vec{u_1}$ is obtained following the first condition of Definition V.1. Since $\rho_1 = 2$ satisfying Condition IV.5, we can consider $\vec{u} \in U_{\rho_1}$, s.t. $Unp(\vec{u}) = \{1, i\}$ where $i \neq 1$, $i \leq 8$. That is $U_{\rho_1} = \{\langle ?, u_{12}, u_{13}, \ldots, u_{18}\rangle$: exactly one of $u_{1j} = ?$ for $j \neq 1$ and $u_{1k} = \vec{u_{ik}}$, $k \neq j, 1\}$. To illustrate more precisely, let us assume $\vec{u_1} = \langle ?, 0, 1, 0, 1, 0, 1, 0\rangle$. Then $U_{\rho_1} = \{\langle ?, ?, 1, 0, 1, 0, 1, 0\rangle, \langle ?, 0, ?, 0, 1, 0, 1, 0\rangle, \langle ?, 0, 1, ?, 1, 0, 1, 0\rangle, \ldots, \langle ?, 0, 1, 0, 1, 0, 1, ?\rangle\}$. For the other vectors $\vec{u_i}$, $i = 2, 3, 4$, the precision levels cannot be obtained following the first condition of Definition V.1 (see Proposition IV.7). Therefore, we follow the process of modifying U_{ρ_i}. According to Proposition IV.7, $O_i = \{1\}$, $\rho_i = p_i + p_1$ for any $i \neq 1$, $1 < i \leq 4$. [7] Therefore we should consider $\vec{u} \in U^{mod}_{\rho_i}$, s.t. $Unp(\vec{u}) = Unp(\vec{u_i}) \cup \{j\}$ where $1 < i \leq 4$, $j \neq 1$, $j \leq 8$, $Unp(\vec{u_i}) \cap \{j\} = \emptyset$. That is $U^{mod}_{\rho_2} = \{\langle u_{21}, ?, ?, u_{24}, \ldots, u_{28}\rangle$: exactly one of $u_{2j} = ?$ for $j \neq 1, 2, 3$ and $u_{2k} = \vec{u_{ik}}, k \neq j, 2, 3\}$. To illustrate this, let us say a given vector $\vec{u_2} = \langle 1, ?, ?, 0, 1, 0, 1, 0\rangle$. Then $U^{mod}_{\rho_2} = \{\langle 1, ?, ?, ?, 1, 0, 1, 0\rangle, \langle 1, ?, ?, 0, ?, 0, 1, 0\rangle, \langle 1, ?, ?, 0, 1, ?, 1, 0\rangle, \langle 1, ?, ?, 0, 1, 0, ?, 0\rangle \langle 1, ?, ?, 0, 1, 0, 1, ?\rangle\}$. Similarly, $U^{mod}_{\rho_3} = \{\langle u_{31}, u_{32}, u_{33}, ?, ?, ?, u_{37}, u_{38}\rangle$: exactly one of $u_{3j} = ?$ for $j \neq 1, 4, 5, 6$ and $u_{3k} = \vec{u_{ik}}, k \neq j, 4, 5, 6\}$ and $U^{mod}_{\rho_4} = \{\langle u_{41}, \ldots, u_{46}, ?, ?\rangle$: exactly one of $u_{4j} = ?$ for $j \neq 1, 7, 8$ and $u_{4k} = \vec{u_{ik}}, k \neq j, 7, 8\}$.*

VI. Information Retrieval from Associative Memory Using Perceptron

In this section, we attempt to sketch the idea of projecting a 2-layer perceptron from a $Dset_{AM}$ generated by an iteration following SDOI. In contrary to [3], in our case a $Dset_{AM}$ contains input data with unknown components. Following [9], [10], if the perceptron is trained on samples with incomplete feature vectors, i.e., partial input data, significant propagation errors may occur during the inference process. In addition, if important features are frequently missing, the learned decision boundaries may become biased or suboptimal, which can degrade the model's prediction performance. To get rid of this problem, it is essential to implement a process to replace missing data before training. This allows the perceptron to learn from partially known data, thereby outputting incomplete input data as complete data. So, below we propose a way to replace the partial input vectors with a potential full vector.

A. Replacing Partial Vectors By Full Vectors

The $Dset_{AM}$ generated by an iteration of SDOI contains vectors satisfying (MDUP). That is, the unknown parts of any pair of vectors do not have common overlap with each others. This feature can be utilized to reconstruct full vectors before training the perceptron model. We replace each vector from a $Dset_{AM}$ as follows.

1) Given $\vec{u_i}, \vec{u_j} \in Dset_{AM}$ for any $i \neq j$, $1 \leq i, j \leq T_n$ the unknown component of the vector $\vec{u_i}$ is replaced with the

[6] Following Theorem V.10, the threshold value $T_n = 3$ for $n = 8$.

[7] We only consider the given vectors $\vec{u_1}, \vec{u_2}, \vec{u_3}, \vec{u_4}$.

known components of the vector $\vec{u_j}$. For vector $\vec{u_i}$ only the positions from $Unp(\vec{u_i}) \cap Bits(\vec{u_j})$ are replaced with that of $\vec{u_j}$. The new vector created in this way is called $\vec{u_{ij}}$. Following similar process $\vec{u_{ji}}$ is also obtained.
2) Check Hamming distance between $\vec{u_{ij}}$ and $\vec{u_{ji}}$, for any $i \neq j$
3) Among the vectors $\vec{u_{ij}}$ $j \neq i$ obtained by exchanging parts with $\vec{u_i}$ the one with minimum Hamming distance, would determine the full vector that can be selected to replace $\vec{u_i}$ considering as the most plausible ideal representation of $\vec{u_i}$.

Following the above method, we can expect that the reconstructed input vector will match more closely the ideal input data set without noise and unknown parts. Using the Hamming distance as a similarity measure, we can systematically select the most reasonable candidate from the pool of potential complete vectors, obtaining an input data set that better matches the intended representation of the original set of partial input vectors.

Example VI.1. *For $n = 10$, in an iteration I_0 we may have $Dset^0_{AM} = \{\vec{u_1}, \vec{u_2}, \vec{u_3}, \vec{u_4}\}$, where $|Unp(\vec{u_i})| = i$, for $1 \leq i \leq 4$, $T_n = 4$.*
Let the vectors be $\vec{u_1} = \langle 0,1,0,0,?,1,1,1,0,0\rangle$, $\vec{u_2} = \langle 1,1,0,0,0,1,0,?,0,?\rangle$, $\vec{u_3} = \langle 1,0,0,0,0,?,?,1,?,0\rangle$, $\vec{u_4} = \langle ?,?,?,?,0,1,1,1,0,0\rangle$, where ? denotes missing value in a vector.
Replacing $Unp(\vec{u_1})$ with $Bits(\vec{u_2})$, $Bits(\vec{u_3})$ and $Bits(\vec{u_4})$, we respectively have $\vec{u_{12}} = \langle 0,1,0,0,0,1,1,1,0,0\rangle$, $\vec{u_{13}} = \langle 0,1,0,0,0,1,1,1,0,0\rangle, \vec{u_{14}} = \langle 0,1,0,0,0,1,1,1,0,0\rangle$.
Similarly, replacing the unknown components $\vec{u_2}$ with the known components of other vectors we have, $\vec{u_{21}} = \langle 1,1,0,0,0,1,0,1,0,1\rangle$, $\vec{u_{23}} = \langle 1,1,0,0,0,1,0,1,0,1\rangle$, $\vec{u_{24}} = \langle 1,1,0,0,0,1,0,1,0,0\rangle$. For $\vec{u_3}$ we have $\vec{u_{31}} = \langle 1,0,0,0,0,1,1,1,0,0\rangle$, $\vec{u_{32}} = \langle 1,0,0,0,0,1,0,1,0,0\rangle$, $\vec{u_{34}} = \langle 1,0,0,0,0,1,1,1,0,0\rangle$, and for $\vec{u_4}$ we have $\vec{u_{41}} = \langle 0,1,0,0,0,1,1,1,0,0\rangle$, $\vec{u_{42}} = \langle 1,1,0,0,0,1,1,1,0,0\rangle$, $\vec{u_{43}} = \langle 1,0,0,0,0,1,1,1,0,0\rangle$.
Next, we check the Hamming distance between each pair of exchanged vectors. $D_H(\vec{u_{12}}, \vec{u_{21}}) = 3$, $D_H(\vec{u_{13}}, \vec{u_{31}}) = 2$, $D_H(\vec{u_{14}}, \vec{u_{41}}) = 0$, $D_H(\vec{u_{23}}, \vec{u_{32}}) = 2$, $D_H(\vec{u_{24}}, \vec{u_{42}}) = 1$, $D_H(\vec{u_{34}}, \vec{u_{43}}) = 0$. [8]
From the above calculation we see, the minimum Hamming distance of $D_H(\vec{u_{1j}}, \vec{u_{j1}})$ where $j \in \{2,3,4\}$, is $D_H(\vec{u_{14}}, \vec{u_{41}})$. So partial vectors $\vec{u_1}$ and $\vec{u_4}$ are exchanged the parts within themselves the newly obtained full vectors $\vec{u_{14}}$ appears to be most closely fitting. Thus, $\vec{u_1}$ can be substituted with full vector $\vec{u_{14}}$. Similarly, partial vector $\vec{u_2}$ can be substituted with full vector $\vec{u_{24}}$, partial vector $\vec{u_3}$ can be substituted with full vector $\vec{u_{34}}$ and partial vector $\vec{u_4}$ can be substituted with full vector $\vec{u_{41}}$ or $\vec{u_{43}}$.

[8]Since Hamming metric, i.e., $D_H(\vec{u_{12}}, \vec{u_{21}}) = D_H(\vec{u_{21}}, \vec{u_{12}})$, there are only six results left in this case excluding duplicate results.

B. Perceptron model

As described in Section II, creating perceptron model following [3] solely depends on two facts; one is choosing precision levels for the input patterns in a way that their associative memories do not overlap and the second is to convert partial associative patterns of a given input (full) pattern to full patterns. The first point is important in establishing the claim that the set of input-outputs pairs in the extended function f_A are linearly separable and it is required to develop the perceptron model for f_A. In our case, we opt to depart from several aspects considered in [3], and these are mentioned in Introduction as well as in Section II. However, from the development and results presented in the sections above, starting from Section III, it is clear that both the contexts responsible for smooth projection of the function f_A in a perceptron model are guaranteed. Thus, in our case the same strategy as in [3] of creating perceptron model can be followed.

VII. Conclusion

The paper presents some limitations of the approach to creating associative memory in the context of retrieving information from partial patterns using perceptron model presented in [3] and proposes a few more general perspectives of different notions used in [3], [8]. First of all, unlike [3] we propose incomplete information in the input data by allowing partial vectors in the input data. Secondly, the paper proposes a new perspective of measuring distance between two partial vectors using the notion of Hamming distance. According to [3] two partial vectors with different unknown fragments cannot be comparable. Following our approach this limitation is overcome. Thirdly, our proposal for creating associative patterns for an input (partial) vector $\vec{u_i}$ depends on a notion of '$\vec{u}$ is a potential extension of $\vec{u_i}$', denoted by $\vec{u} \otimes \vec{u_i}$. The idea is developed generalizing the notion of $\odot$ presented in [3]. Moreover, in the process of converting incomplete or partial patterns by some plausible full patterns in [3] a random process of replacing unknown parts by 0's is followed. Whereas in our context, we consider an approach of finding plausible full pattern of an input partial pattern by exchanging content of the given pattern with other input vectors of the training data. The idea is similar to solving a puzzle related to constructing a full image of a desired object from several available pieces where different fragments of the full image are visible.

From theoretical perspective, though we manage to resolve the limitations mentioned before, following our proposal a given data set of n-dimensional input vectors cannot be used together in one iteration to train a perceptron model. However, there seems to be a possibility of creating perceptron model from aggregating data from different iterations. In [12], the authors proposed different models for generalizing Hamming associative memory. In particular, they introduced a notion of *local Hamming memory* which is based on splitting the input vectors (input data) into smaller clusters and learning the dependence of all the input vectors in that cluster with their respective outputs. Modifying the idea of local

Hamming memory the authors [12] also defined a notion of *decoupled Hamming associative memory* which considers non-overlapping windows (clusters) of input vectors. Though there is a similarity of their considered idea of non-overlapping windows of input vectors [12] with our idea of splitting the input vectors over iterations, there is a difference in basic consideration. In our case, the input vectors can be partial and in [12] the input vectors are full binary vectors. Following [3] non-overlapping associative memories of the input vectors allows a direct projection of the extended function f_A into a multilayered perceptron model. In our context, though we have departed from [3] on many aspects, we stick to the technical advantage of non-overlapping associative memories of the input vectors. However, following the notion of Distributed Associative Memory (DAM), in [13], the authors envisaged a possibility of representing multiple association of an input data with different sets of patterns. This idea can be worth investigating further in our context by allowing associative patterns of two input vectors to have common intersection. But this needs additional research and experiments in order to optimize the performance of the proposed method and check the changes needed to develop the set-up based on DAM.

REFERENCES

[1] P. M. Churchland, *Plato's camera: How the physical brain captures a landscape of abstract universals*. MIT press, 2012. DOI: 10.1080/10848770.2016.1139356. [Online]. Available: https://doi.org/10.7551/mitpress/9116.001.0001.

[2] T. M. Mitchell, *Machine Learning*, 1st ed. USA: McGraw-Hill, Inc., 1997, ISBN: 0070428077. [Online]. Available: https://dl.acm.org/doi/10.5555/541177.

[3] C.-H. Chen and V. Honavar, "A neural architecture for content as well as address-based storage and recall: Theory and applications," *Connection Science*, vol. 7, no. 3, pp. 281–300, 1995. DOI: 10.1080/09540099509696194.

[4] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982. DOI: 10.1073/pnas.79.8.2554.

[5] B. Prasad, K. Prasad, S. Yeruva, P. Murty, and S. Rama, "A study on associative neural memories," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 1, no. 6, pp. 124–133, 2010. DOI: 10.14569/IJACSA.2010.010619.

[6] G. R. Barker and E. C. Warburton, "Multi-level analyses of associative recognition memory: The whole is greater than the sum of its parts," *Current opinion in behavioral sciences*, vol. 32, pp. 80–87, 2020. DOI: 10.1016/j.cobeha.2020.02.004.

[7] J. Steinberg and H. Sompolinsky, "Associative memory of structured knowledge," *Scientific Reports*, vol. 12, no. 1, p. 21 808, 2022. DOI: 10.1038/s41598-022-25708-y.

[8] C. H. Chen and V. Honavar, "Neural network architecture for high-speed database query processing," *Microcomputer Applications*, vol. 15, no. 1, pp. 7–13, 1996, ISSN: 0820-0750. [Online]. Available: https://dr.lib.iastate.edu/handle/20.500.12876/20228.

[9] C. F. Caiafa, Z. Wang, J. Solé-Casals, and Q. Zhao, "Learning from incomplete data by simultaneous training of neural networks and sparse coding," *arXiv preprint arXiv:2011.14047*, 2020. DOI: 10.48550/arXiv.2011.14047.

[10] W.-D. Chang, J. Shin, M.-Y. Song, and T.-S. Chon, "Enhanced back propagation algorithm for estimating ecological data with missing values," *WSEAS Transactions on Computers*, vol. 5, no. 9, pp. 2043–2048, Sep. 2006.

[11] C. Chen and V. Honavar, "Neural network automata," in *Proc. of World Congress on Neural Networks*, vol. 4, 1994, pp. 470–477.

[12] P. Watta and M. H. Hassoun, "Generalizations of the hamming associative memory," *Neural processing letters*, vol. 13, no. 2, pp. 183–194, 2001. DOI: 10.1023/A:1011384407294.

[13] T. Park, I. Choi, and M. Lee, "Distributed associative memory network with memory refreshing loss," *Neural Networks*, vol. 144, pp. 33–48, 2021. DOI: 10.1016/j.neunet.2021.07.030.