

Proto Logic and Neural Sub-Symbolic Reasoning

Andreas Wichert
Department of Informatics
INESC-ID / IST - Technical University of Lisboa
Portugal
andreas.wichert@ist.utl.pt

April 11, 2012

Abstract

The sub-symbolic representation of the world often corresponds to a pattern that mirrors the world as described by the biological sense organs. Sparse binary vectors can describe sub-symbolic representations, which can be efficiently stored in associative memories. According to the production system theory, a geometrically based problem-solving model can be defined as a production system operating on sub-symbols. Our goal is to form a sequence of associations, which lead to a desired state represented by sub-symbols, from an initial state represented by sub-symbols. A simple and universal heuristic function can be defined, which takes into account the relationship between the vector and the corresponding similarity of the represented object or state in the real world. The manipulation of the sub-symbols is described by a simple proto logic, which verifies if a subset of sub-symbols is present in a set of sub-symbols.

1 Introduction

One form of distributed representation corresponds to a pattern that mirrors the way the biological sense organs describe the world. Sense organs sense the world by receptors. The order of the receptors defines the reality as a simple Euclidian geometry. It is the basis of the distributed representation. Changes in the world correspond to the changes in the distributed representation. Prediction of these changes by the nervous system is an example of a simple geometrical reasoning process. Mental imagery problem solving is an example of a complex geometrical problem solving. It is described by a sequence of associations, which progressively change the mental imagery until a desired problem solution is formed. For example, do the skis fit in the boot of my car? Mental representations of images retain the depictive properties of the image itself as perceived by the eye Kosslyn [1994]. The imagery is formed without perception through the construction of the represented object from memory.

Symbols on the other hand are not present in the world; they are the constructs of a human mind and simplify the process of representation used in communication and problem solving. Symbols are used to denote or refer to other things in the world (according to the pioneering work of Tarski Tarski [1956]). They are defined by their occurrence in a structure and by a formal language, which manipulates these structures Newell [1990], Simon [1991]. In this context, symbols do not by themselves represent any utilizable knowledge. They cannot be used for a definition of similarity criteria between themselves. The use of symbols in algorithms

which imitate intelligent human behaviour led to the famous physical symbol system hypothesis by Newell and Simon Newell and Simon [1976]: “The necessary and sufficient condition for a physical system to exhibit intelligence is that it be a physical symbol system.”

The author does not agree with the physical symbol system hypothesis. Instead the author states: the actual perception of the world and manipulation in the world by living organisms lead to the invention or recreation of an experience. The recreation resembles at least in some respects the experience of actually perceiving and manipulating objects, however in the absence of direct sensory stimulation. This kind of representation is called sub-symbolic.

Sub-symbolic representation suggests a heuristic function based on similarity between sub-symbols. Symbols liberate people from the reality of the world although they are embodied in geometrical problem solving through the usage of additional heuristic functions. Without the use of heuristic functions, real world problems become intractable.

In this paper the basis of the manipulation of the sub-symbols is described by simple proto logic, which verifies if a subset of sub-symbols is present in a set of sub-symbols in contrast to other powerful logics like predicate or temporal logics.

The paper is organized as follows: the representation principles of objects by features as used in cognitive science is reviewed. In the next step, the paper indicates how the perception-oriented representation is built on this approach. The optimal sparse sub-symbolic representation is defined. Finally, the sub-symbolic problem solving, which relies on a sensorial representation of reality, is introduced.

2 Sub-symbols

Perception-oriented representation is an example of sub-symbolic representation, such as the representation of numbers by the Oksapmin tribe of Papua New Guinea. The Oksapmin tribe of Papua New Guinea counts by associating a number with the position of the body Lancy [1983]. The sub-symbolic representation often corresponds to a pattern that mirrors the way the biological senses organs describe the world. Vectors represent patterns. A vector is only a sub-symbol if there is a relationship between the vector and the represented object or state in the real world through sensors or biological senses. Feature based representation is an example of sub-symbolic representation.

2.1 Feature Approach

Objects can be described by a set of discrete features, such as red, round and sweet McClelland and Rumelhart [1985], Tversky [1977]. The similarity between them can be defined as a function of the features they have in common Gilovich [1999], Goldstone [1999], Osherson [1995], Sun [1995]. The contrast model of Tversky Tversky [1977] is one well-known model in cognitive psychology Opwis and Plötzner [1996], Smith [1995], which describes the similarity between two objects, which are described by their features. An object is judged to belong to a verbal category to the extent that its features are predicted by the verbal category Osherson [1987]. The similarity of a category represented by a feature set C and of a feature set F is given by the following formula, which is inspired by the contrast model of Tversky Opwis and Plötzner [1996], Smith [1995], Tversky [1977],

$$Sim(C, F) = \frac{|C \cap F|}{|C|} \in [0, 1] \quad (1)$$

$|C|$ is the number of the prototypical features that define the category a . For example, the category *bird* is defined by the following features: flies, sings, lays eggs, nests in trees, eats insects. The category *bat* is defined by the following features: flies, gives milk, eat insects. The following features are present: flies and gives milk.

$$\begin{aligned} Sim(\mathbf{bird}, present\ features) &= \frac{1}{5} \\ Sim(\mathbf{bat}, present\ features) &= \frac{2}{3} \end{aligned}$$

The present features are counted and normalized so that the value can be compared. The similarity value can be interpreted as a probability that the object belongs to the category. This is a very simple and efficient form of representing sub-symbols. A binary vector in which the positions represent different features can represent the set of features. For each category a binary vector can be defined. Overlaps between stored patterns correspond to overlaps between categories.

2.2 Sub-symbolic representation by associative memory

Associative memory models human memory Churchland and Sejnowski [1994], Fuster [1995], Palm [1990], Squire and Kandel [1999]. The associative memory and sub-symbolic distributed representation incorporate the following abilities in a natural way Anderson [1995b], Hertz et al. [1991], Kohonen [1989], Palm [1982]:

- The ability to correct faults if false information is given.
- The ability to complete information if some parts are missing.
- The ability to interpolate information. In other words, if a sub-symbol is not currently stored the most similar stored sub-symbol is determined.

The Lernmatrix, also simply called “associative memory”, was developed by Steinbuch in 1958 as a biologically inspired model from the effort to explain the psychological phenomenon of conditioning Steinbuch [1961, 1971]. Later this model was studied under biological and mathematical aspects by Willshaw Willshaw et al. [1969] and Palm Palm [1982, 1990].

Associative memory is composed of a cluster of units. Each unit represents a simple model of a real biological neuron. The Lernmatrix was invented by Steinbuch, whose goal was to produce a network that could use a binary version of Hebbian learning to form associations between pairs of binary vectors, for example each one representing a cognitive entity. Each unit is composed of binary weights, which correspond to the synapses and dendrites in a real neuron. They are described by $w_{ij} \in \{0, 1\}$ in Figure 1. T is the threshold of the unit. The Lernmatrix is simply called *associative memory* if no confusion with other models is possible Anderson [1995a], Ballard [1997].

The patterns, which are stored in the Lernmatrix, are represented by binary vectors. The presence of a feature is indicated by a ‘one’ component of the vector, its absence through a ‘zero’ component of the vector. A pair of these vectors is associated and this process of association is called learning. The first of the two vectors is called the *question vector* and the second, the *answer vector*. After learning, the question vector is presented to the associative memory and the answer vector is determined by the retrieval rule.

Learning Initially, no information is stored in the associative memory. Because the information is represented in weights, all unit weights are initially set to zero.

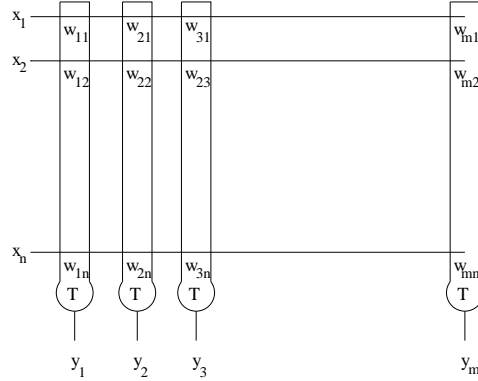


Figure 1: The Lernmatrix is composed of a set of units which represent a simple model of a real biological neuron. The unit is composed of weights, which correspond to the synapses and dendrites in the real neuron. In this figure they are described by $w_{ij} \in \{0, 1\}$ where $1 \leq i \leq m$ and $1 \leq j \leq n$. T is the threshold of the unit.

In the learning phase, pairs of binary vector are associated. Let \vec{x} be the question vector and \vec{y} the answer vector, the learning rule is:

$$w_{ij}^{new} = \begin{cases} 1 & \text{if } y_i \cdot x_j = 1 \\ w_{ij}^{old} & \text{otherwise.} \end{cases} \quad (2)$$

This rule is called the binary Hebbian rule Palm [1982]. Every time a pair of binary vectors is stored, this rule is used.

Retrieval In the *one-step* retrieval phase of the associative memory, a fault tolerant answering mechanism recalls the appropriate answer vector for a question vector \vec{x} . The retrieval rule for the determination of the answer vector \vec{y} is:

$$y_i = \begin{cases} 1 & \sum_{j=1}^n w_{ij} x_j = T \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

where T is the threshold of the unit. The threshold T is set to the number of “one” components in the question vector \vec{x} , $T := |\vec{x}|$. It is quite possible that no answer vector is determined (zero answer vector). This happens when the question vector has a subset of components that was not correlated with the answer vector.

Storage capacity For an estimation of the asymptotic number L of vector pairs (\vec{x}, \vec{y}) that can be stored in an associative memory before it begins to make mistakes in the retrieval phase, it is assumed that both vectors have the same dimension n . It is also assumed that both vectors are composed of k ones, which are equally likely to be in any coordinate of the vector. In this case it was shown Hecht-Nielsen [1989], Palm [1982], Sommer [1993] that the optimum value for k is approximately

$$k \doteq \log_2(n/4). \quad (4)$$

For example for a vector of the dimension $n=1000000$ only $k = 18$ ones should be used to code a pattern according to the Equation 4. For an optimal value for k according to the Equation 5 with ones equally distributed over the coordinates of the vectors, approximately L vector pairs can be stored in the associative memory Hecht-Nielsen [1989], Palm [1982]. L is approximately

$$L \doteq (\ln 2)(n^2/k^2). \quad (5)$$

This value is much greater than n . The estimate of L is very rough because Equation 6 is only valid for very large networks. Equation 6 does not apply for networks of reasonable size, however the capacity increase is still considerable. For realistic values please consult Table 2 in Knoblauch et al. [2010]. Small deviation from the logarithmic sparseness reduces the network capacity. It is very difficult to find coding schemas that represent the information by logarithmic sparse codes Knoblauch et al. [2010].

It should be noted that the Lernmatrix system allows high capacity and fast access when working in parallel, each unit represents a neuron that performs calculations. On a conventional Von Neumann architecture, compressed look-up tables are more efficient Knoblauch et al. [2010]. However a Von Neuman architecture is not biologically plausible.

3 Sparse Code for Sub-Symbols

Usually suboptimal sparse codes are used. An example of a suboptimal sparse code is the representation of words by context-sensitive letter units Bentz et al. [1989], Rumelhart and McClelland [1986], Wickelgren [1969, 1977]. The ideas for the used robust mechanism come from psychology and biology Bentz et al. [1989], Rumelhart and McClelland [1986], Wickelgren [1969, 1977]. Each letter in a word is represented as a triple, which consists of the letter itself, its predecessor, and its successor. For example, six context-sensitive letters encode the word *desert*, namely: *_de, des, ese, ser, ert, rt_*. The character “*_*” marks the word beginning and ending. Because the alphabet is composed of 26+1 characters, 27^3 different context-sensitive letters exist. In the 27^3 dimensional binary vector each position corresponds to a possible context-sensitive letter, and a word is represented by indication of the actually present context-sensitive letters.

A set of features can be represented by a binary vector and represent a category. A position in the corresponding vector corresponds to a feature. To be sparse, the set of features that describes a category compared to the dimension of the vector has to be sufficiently small. This is because, of all possible features, only some should define categories. This can be achieved by sparsification based on unary sub-vector representation.

3.1 Sparsification based on unary sub-vectors

A binary representation of a number h would require a vector of length $d = \lfloor \log_2 h \rfloor + 1$. However if we represent the number h in unary, we require h positions. One unary representation of $h \neq 0$ is a string of $h - 1$ zeros with a one at h -th position. A binary number of length d is represented by a unary number of 2^d positions, which is exponential in the size of input. A binary vector \vec{x} of dimension t is split into f distinct sub vectors of dimension $p = \dim(t/f)$. The binary sub vectors $u_i(\vec{x})$ of dimension $p = \dim(t/f)$ are represented as unary vectors of dimension 2^p :

$$\vec{x} = \underbrace{x_1, x_2, \dots, x_p}_{u_1(\vec{x})}, \dots, \underbrace{x_{m-p+1}, \dots, x_m}_{u_f(\vec{x})} \quad (6)$$

The resulting binary vector is composed out of the unary vectors and has the dimension $f \cdot 2^p$. In the following example a binary vector of dimension 6 is split into 2 distinct sub vectors of dimension 3. The binary sub vectors $u_i(\vec{x})$ of dimension 3 are represented as unary vectors of dimension 2^3 :

$$\vec{x} = \underbrace{1, 0, 1}_{u_1(1,0,1)}, \underbrace{0, 0, 1}_{u_2(0,0,1)} \quad (7)$$

$$u_1(1, 0, 1) = (0, 0, 0, 0, 1, 0, 0, 0); \quad (h = 5)$$

$$u_2(0, 0, 1) = (1, 0, 0, 0, 0, 0, 0, 0); \quad (h = 1)$$

$$u(\vec{x}) = (0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0) \quad (8)$$

Resulting in a new vector of dimension $16 = 2 * 2^3$ with 2 ones.

3.2 Sensors at different positions

Such a binary sparse vector could correspond to set of c sensors at different positions. At a position one and only one sensor is activated. For f positions and c sensors a state would be represented by a binary vector of the dimension $f * c$ with f ones. In the preceding example $f=2$ and $c=8$. An example of such unary coding is the Map Transformation Cascade model of the visual system Cardoso and Wichert [2010]. Several models of the visual system [Cardoso and Wichert, 2010, Fukushima, 1980, 1989, Riesenhuber and Poggio, 1999] were motivated by the work on the visual systems of Hubel and Wiesel. The neural units have local receptive fields and are ordered in layers. The layers form a hierarchy in the sense that features at one stage are built from features at earlier stages. An image passes through layers of units with progressively more complex features. The hierarchical network gradually reduces the information from the input layer through the output layer until classification can be performed. The proposed Map Transformation Cascade Cardoso and Wichert [2010] is a less complex description of the pattern recognition capabilities of the Neocognitron. Each layer represents a set of features. A binary vector in which the positions represent different features at different positions on the image describes it. The input image is tiled with a squared mask M of size $j \times j$ in which a corresponding category of a feature is determined, see Figure 2. Each feature is determined through the use of the elements in each squared mask. Each of the corresponding f sub-patterns \vec{x}_t , with $t \in \{1, 2, \dots, f\}$, is mapped into one corresponding category represented by a number h . The categories can be learned by a simple clustering algorithm such as K-Means Cardoso and Wichert [2010]. The number of categories is represented by the number c . A category h is represented by a unary vector of dimension c with $c - 1$ zeros and a one at the position h . The whole image state is represented by a binary vector of dimension $n = c \times f$ with f ones. This vector is formed by the concatenation of the unary vectors that represent the categories at different positions.

3.3 Logarithmic sparsification

The ideal c value for a logarithmic sparse code is related to the number of ones $k \doteq \log_2(n/4)$.

$$\begin{aligned} k &= \log_2(f \cdot c/4) \\ 2^k &= f \cdot c/4 \\ c &= \frac{4 \cdot 2^k}{f} \end{aligned} \quad (9)$$

The ideal value for c grows exponentially in relation to f with the assumption that the number of ones is $k = f$. Usually the value for c is much lower than the ideal

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

Figure 2: The input image is tiled with a squared mask M of size $j \times j$ in which a corresponding category of a feature is determined. Each feature is determined through the use of the elements in each squared mask. In this example, a simple image is covered with $f = 36$ masks, a category would correspond to a line or an edge at a certain orientation.

value resulting in a suboptimal sparse code. The representation of images by masks results in a suboptimal code. The optimal code is approached with the size of masks, the larger the mask, the smaller the value of f . The number of pixels inside a mask grows quadratically in the size of the edge. A larger masks implies the ability to represent more distinct categories, which implies a bigger c . An ideal value for c is possible only if the value for $f \ll 100$.

3.4 Logarithmic sparsification based on cognitive entities

Often no category is present at a certain location. The non presence of a category could be represented by a vector with c zeros ($h=0$). Suppose that the actual number of present categories is much smaller than the number of positions, in this case $k \ll f$. A pointer representation of objects in a scene leads to an even sparser representation. In this case the number of ones k is not related to the number of categories at f positions, but to the number of objects at f possible positions. Objects and their positions in the visual field can represent a visual scene. A sub-vector of the vector representing the visual scene represents each object.

3.4.1 Cognitive entities

It was suggested Gross and Mishkin [1977] that the brain includes two mechanisms for visual categorization Posner and Raichle [1994]: one for the representation of the object and the other for the representation of the localization Kosslyn [1994]. According to this division, the identity of a visual object can be coded apart from its location. A visual scene can be either represented by an image or by objects and their position in the visual field. Objects are represented by patterns together with their corresponding position in the image. Cognitive entities Anderson [1995a] represent objects and their position in the image. Each cognitive entity represents the identity of the object and its position is given by Cartesian coordinates (see Figures 3, 4 and 5). The advantage of such cognitive entity representation is that the manipulation of objects is simplified and it is a basis for a binary sparse code

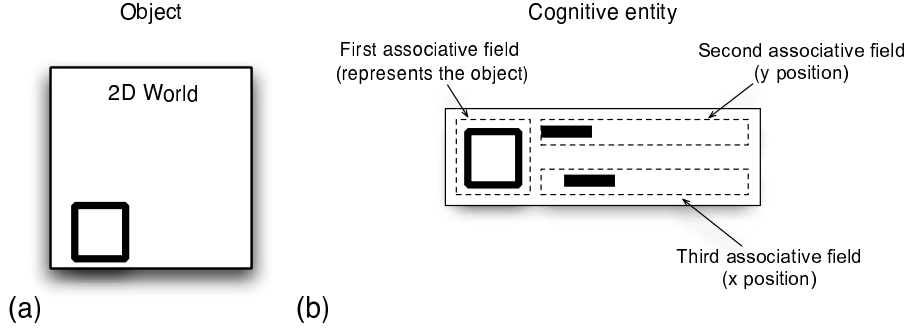


Figure 3: Representation of an object in a 2D world (a) by a cognitive entity (b). The identity of an object is represented in the first associative field by a binary pattern that is normalized for size and orientation. Its location corresponding to the abscissa is represented by a binary vector in the second associative field. The location corresponding to the ordinate is likewise represented by a binary vector in the third associative field of the size of the ordinate of the pictogram representing the state. A binary bar of the size and position of the object in the pictogram of the state represents the location.

that can be stored efficiently in an associative memory.

The identity of an object is represented by a binary pattern which is normalized for size and orientation. Its location on the x -axis is represented by a binary vector of the size of the abscissa of the pattern representing the object. The location on the y -axis is likewise represented by a binary vector of the size of the coordinate of the pattern representing the object. A binary bar of the size and position of the object in the pictogram of the state represents the location and size (see Figure 3) in each of those vectors. The three vectors that compose the cognitive entity are called associative fields. Each associative field is represented by a binary vector of a fixed dimension; each cognitive entity is formed by the concatenation of the associative fields.

3.4.2 Representation of a cognitive entity by a unary vector

A cognitive entity can be represented alternatively by a unary vector. A simple code for an object would indicate if it is present or not. One indicates present, zero not present. Four categories of objects are represented in this example by the first associative field; cube, cube clear, pyramid and pyramid clear ($c = 4$, see Figure 6, Figure 4 and 5).

The presence of a category is indicated by a unary vector of dimension four. There are $x \times y$ possible positions of the object. In our example there are 10×10 possible positions, see Figure 7.

A cognitive entity represents an object at a certain position. The corresponding category is represented by a unary vector of dimension four on the corresponding position. For each remaining position, a zero vector of dimension four is repeated, see Figure 8. The principle of forming a unary vector representing a cognitive entity is based on the tensor product between the vector representing the category that is present or not and the vector representing the position.

As a result, a cognitive entity is represented by a unary vector of the dimension $c \times x \times y$ or $c \times f$, see Figure 8.

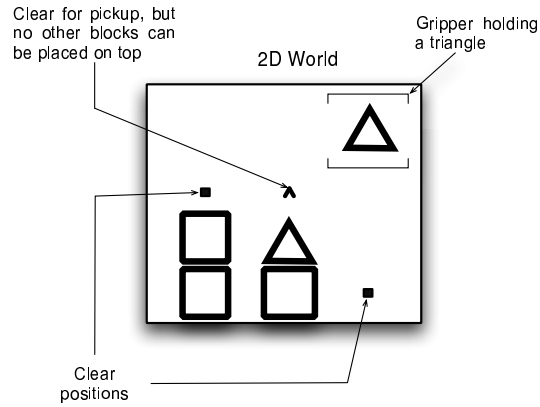


Figure 4: A state in the geometric block world. Blocks can be placed in three different positions and picked up and set down. There are two different categories of blocks: cubes and pyramids. No other block may be placed on top of a pyramid, while either type of block may be placed on top of a cube. The gripper is represented in the upper right corner. Five objects are present: three cubes and two pyramids. The “clear” positions are represented by a dot.

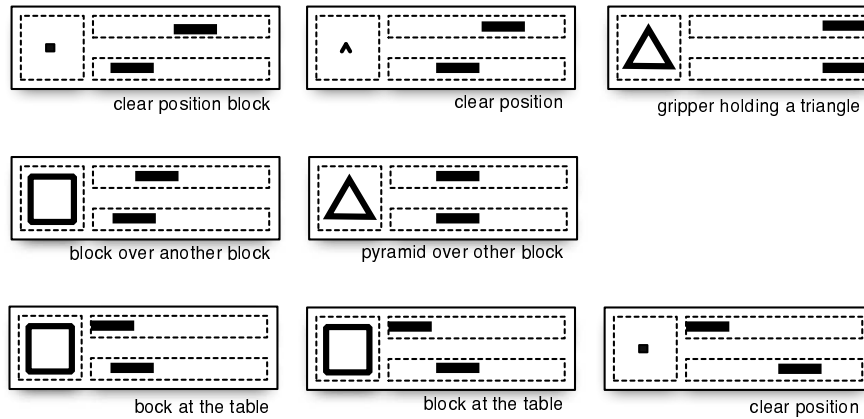


Figure 5: Alternative representation of the geometric block world state (see Figure 4) represented by a set of eight cognitive entities.



Figure 6: Four categories of objects are represented in our example by the first associative field; cube, clear, pyramid and pyramid clear.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Figure 7: There are $f = 10 \times 10$ possible positions of 4 categories of objects. A cognitive entity is represented by a unary vector of the dimension 4×100 . Seven objects are represented, two cubes, three clears, one pyramid and one clear pyramid. The visual scene is represented by seven unary vectors. Each cognitive entity corresponds to a unary sub-vector.

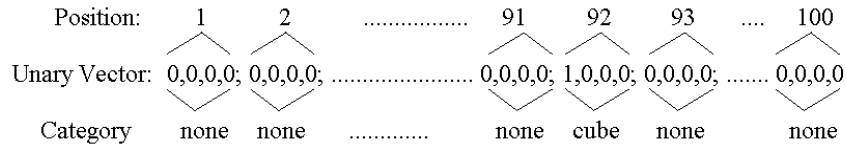


Figure 8: A cognitive entity represents an object at a certain position. The corresponding category is represented by a unary vector of dimension four on the corresponding position, in our case a cube on the position 92. For each remaining position, a zero vector of dimension four is repeated indicating that no category is present.

3.4.3 Sparse scene representation based unary sub-vectors

A scene is represented by a set of these unary vectors resulting in a sparse binary vector. Each cognitive entity corresponds to a unary sub-vector, which represents an object. A set of those sub-vectors represents a visual scene.

For an image of size $x \times y$, an object covers the image f times. In our example there are $f = 10 \times 10$ possible positions of 4 categories of objects. A cognitive entity is represented by a unary vector of the dimension 4×100 , m objects are represented by m cognitive entities. In a vector representing a visual scene, the objects correspond to unary sub-vectors of fixed length. This form of representation is called the set pointer representation and it corresponds to the cognitive entity representation (see Figure 5). Set pointer representation allows the manipulation of objects by means of proto logic. Proto logic allows the access of objects of a set as shown in section 3. A binary vector representing a visual scene like the state in the block world is formed by the concatenation of the unary vectors. This representation is called a “set”, because the order of the unary vectors representing the cognitive entities is not defined. This representation is logarithmic sparse, in our example for $m = 7$ objects, $c = 4$ and $f = 100$ resulting in a vector of dimension 2800 with only seven ones. The $k = m$ and the maximal number of represented objects of four different categories ($c=4$) at $f = 100$ positions is constrained by $m < 10$ for a logarithmic sparse code:

$$\begin{aligned} k &\doteq \log_2(n/4) \\ k &\leq \log_2(m \cdot 4 \cdot 100/4) \\ k &\leq \log_2(m) + \log_2(100) \end{aligned}$$

for $k=m$,

$$m - \log_2(m) \leq 6.64$$

To compute a distance between two visual scenes represented by two “sets” of m cognitive entities, resulting in two $c \times f \times m$ binary sparse vector with m ones, one computes a bitwise “OR” between the n sub-vectors representing the m cognitive entities of a set, resulting in a $c \times f$ dimensional vector with m ones (compressed ordered representation). The similarity is measured by Equation 1. The similarity between visual scenes is defined as a function of the objects and their locations they have in common Gilovich [1999], Goldstone [1999], Osherson [1995], Sun [1995].

4 Sub-symbolic Production System based on Proto Logic

Human problem solving can be described by a problem-behaviour graph constructed from a protocol of the person talking aloud, mentioning considered moves and aspects of the situation. According to the resulting theory, searching a state includes the initial situation and the desired situation in a problem space that solves problems Anderson [1995b], Newell [1990]. This process can be described by the production system theory. The production system in the context of classical Artificial Intelligence and Cognitive Psychology is one of the most successful computer models of human problem solving. The production system theory describes how to form a sequence of actions, which lead to a goal, and offers a computational theory of how humans solve problems Anderson [1995b]. Production systems are composed of if-then rules that are also called productions. A rule contains several “if” patterns and one or more “then” patterns. A pattern in the context of rules is an individual predicate, which can be negated together with arguments. A rule can establish a new

assertion by the “then” part (its conclusion) whenever the if part (its precondition) is true. One of the best-known cognitive models, based on the production system, is Soar. The Soar state, operator and result model was developed to explain human problem-solving behaviour Newell [1990]. It is a hierarchical production system in which the conflict-resolution strategy is treated as another problem to be solved.

According to the production system theory, a geometrically based problem-solving model can be defined as a production system operating on vectors of fixed dimension. Instead of rules, associations are used and vectors represent states. Instead of predicates and facts, sub-vectors and proto logic are used. The goal is to form a sequence of associations, which lead to a desired state represented by a vector, from an initial state represented by a vector. Each association changes some parts of the vector. In each state, several possible associations can be executed, but only one has to be chosen, otherwise, conflicts in the representation of the state would occur. To perform these operations, a vector representing a state is divided into sub-vectors. An association recognizes some sub-vectors of the vector and exchanges them for different sub-vectors. The association is composed of a precondition of fixed arranged β sub-vectors and a conclusion of β sub-vectors. Associations are learned by the associative memory (see Figure 9). Each cognitive entity is represented by a unary vector. A precondition and the conclusion are represented by a $4 \times 100 \times 3 = 1200$ dimensional binary vector with three ones.

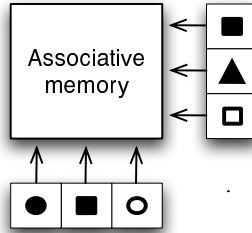


Figure 9: The learning phase of an association represented by 3 sub-vectors. In our example the precondition and the conclusion are represented by a $4 \times 100 \times 3 = 1200$ dimensional binary vector with three ones.

4.1 Proto Logic

Suppose a vector is divided into α sub-vectors with $\alpha > \beta$. An association recognizes β different sub-vectors and exchanges them for β different sub-vectors.

Let $\alpha = 7$ objects that were recognized in the visual scene. The seven visual objects are indicated at a certain position of the scene by symbols A, B, C, D, E, F and G . The task of proto logic is to identify a precondition formed by visual objects represented by the set B, C, G , $\beta = 3$. *Proto logic operates on sets. It verifies whether a subset is present in a certain set.* The task of proto logic is trivial when working with sets. Each of the symbols B, C, G is checked for presence in the set that represents the scene. It is verified if a set representing a precondition is a subset of the set representing a scene.

However if the precondition (set of objects) is stored in an associative memory the task of proto logic is non trivial Wichert [2011]. In an associative memory direct access to the stored information is not present. An associative memory operates on vectors of fixed dimensions.

A set of objects (a precondition) is represented by a vector by concatenating the sub-vectors that represent the objects. For m sub-vectors there are $m!$ possible orderings of the corresponding sub-vectors. Each sub-vector corresponds to a cognitive entity.

To verify if a set of β sub-vectors representing a precondition is a subset of the set of n sub-vectors representing a scene, there are $\frac{\beta!}{(\alpha-\beta)!}$ orderings. Then it is verified if each permutation corresponds to a valid precondition of an association. For example, if there is a total of seven elements and a sequence of three elements from this set is selected, then the first selection is one from seven elements, the next one from the remaining six, and finally one from the remaining five, resulting in $7 * 6 * 5 = 210$, see Figure 10. In our example, all possible three-permutation sub-vectors of seven sub-vectors are formed to test if the precondition of an association is valid. An association is valid, if the answer vector representing the conclusion is not equal to a zero vector.

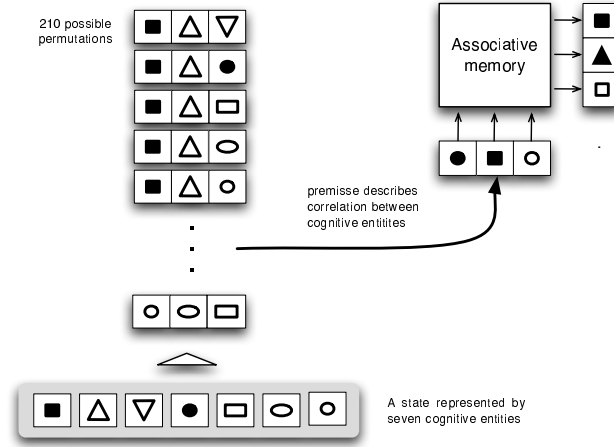


Figure 10: To recognize one learned association, permutations are formed. For example, if there is a total of seven elements and a sequence of three elements from this set is selected, then the first selection is one from seven elements, the next one from the remaining six, and finally from the remaining five, resulting in $7 * 6 * 5 = 210$. In our example, all possible three-permutation sub-vectors of seven sub-vectors are formed to test if the precondition of an association is valid.

4.2 Sub-symbolic Problem solving

Sub-symbolic problem solving forms a sequence of associations that change the initial scene in the desired scene. The input to the problem is the initial and the desired scene. The solution is the sequence of associations. The sub-symbolic problem solving is based on the following principles:

- A scene is represented by α different sub-vectors. Each sub-vector corresponds to a cognitive entity.
- The association is composed of a precondition of fixed arranged β sub-vectors and a conclusion of β sub-vectors with $\alpha > \beta$.
- Proto logic verifies whether a subset, β sub-vectors of is present in a set of α different sub-vectors representing a scene.

Algorithm for sub-symbolic problem solving

1. For a scene (starting with the initial scene), represented by α different sub-vectors;
2. Valid associations are determined, their number is indicated by ω .
3. ω identical copies of the set of α different sub-vectors representing a scene are formed.
4. For each of the ω valid associations and each copy of the set representing the scene;
 - (a) The subset of β sub-vectors of the precondition is replaced by the subset β sub-vectors of the conclusion of the association resulting in a temporal answer set.
5. ω temporal answer sets of α sub-vector are mapped into ω answer vectors by performing a bitwise “OR” between the α sub-vectors representing the α cognitive entities of each temporal answer set.
6. The similarity between the ω answer vectors and the desired state is measured by Equation 1.
7. If the similarity corresponds to equality, the problem is solved and the computation is terminated.
8. Otherwise the most similar answer vector to the desired state according to by Equation 1 is chosen. The corresponding temporal answer set represents the scene and a new cycle of computation is repeated. The computation is repeated in cycles until a solution is found. This search strategy corresponds to hill climbing Winston [1992].

Out of several possible associations, the one is chosen that modifies the state in such a way that it becomes more similar to the desired state according to Equation 1. The desired state corresponds to the category of Equation 1, each feature represents a possible state. The states are represented by sparse features. With the aid of this, heuristic hill climbing is performed.

4.3 Simple and universal heuristic function

The computation can be improved by this simple and universal heuristic function, which takes into account the relationship between the vector and the corresponding similarity of the represented states (see Figure 11 and Figure 12). The heuristics function makes a simple assumption that the distance between the states in the problem space is related to the similarity of the vectors representing the states. The similarity between the corresponding vectors can indicate the distance between the sub-symbols representing the state. Empirical experiments in popular problem-solving domains of Artificial Intelligence, like a robot in a maze, block world or 8-puzzle indicate that the distance between the states in the problem space is actually related to the similarity between the images representing the states Wichert [2001, 2009], Wichert et al. [2008].

The hill climbing search resulted from the fact that distance between states in the problem space is related to the similarity between the sub-symbols. This heuristic is fairly simple and cannot be applied to problems where the similarity of the representation is not related to the distance in the problem space, such as for example the missionaries and cannibals problem. This also happens due to the fact

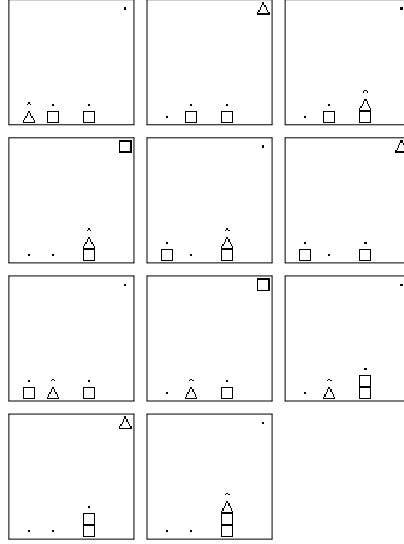


Figure 11: The simplest method corresponds to a random choice, and does not offer any advantage over simple symbolical representation. An example of visual planning of the tower building task of three blocks using random choice is shown. The upper left pattern represents the initial state; the bottom right pattern, the desired state.

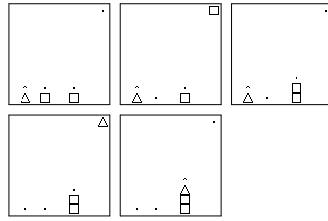


Figure 12: The computation can be improved by a simple and universal heuristic function, which takes into account the relationship between the vector and the corresponding similarity of the represented object or states in the real world as expressed by Equation 1 for binary vectors. The heuristics function makes a simple assumption that the distance between the states in the problem space is related to the distance between the sub-symbols representing the visual states. The distance between the states in the problem space is related to the distance between the visual state. An example of visual planning of the tower building task of three blocks using hill climbing using the similarity function, see Equation 1. The upper left pattern represents the initial state; the bottom right pattern, the desired state.

that we do not represent the problem space and our system gets caught in loops and fails to deliver a solution. Of course humans have difficulties with problems like the missionaries and cannibals problem, in which one cannot perform the first necessary action without undoing them at a later stage. In case the problems become too complex, the sub-symbolic problem is often transferred to a symbolic representation and solved using external memory in the real world, like paper and pencil Wichert et al. [2008].

5 Conclusion

Living organisms experience the world as a simple Euclidian geometrical world. The actual perception of the world and manipulation in the world by living organisms lead to the invention or recreation of an experience that, at least in some respects, resembles the experience of actually perceiving and manipulating objects in the absence of direct sensory stimulation. This kind of representation is called sub-symbolic. The manipulation of the sub-symbols is described by simple proto logic, which verifies if a subset of sub-symbols is present in a certain set of sub-symbols. Sub-symbolic representation implies heuristic functions. The assumption that the distance between states in the problem space is related to the similarity between the sub-symbols representing the states is only valid in simple cases. However, simple cases represent the majority of problems in any real world domain. Sense organs sense the world by receptors that are part of the sensory system and the nervous system. Optimal sparse binary vectors can describe sub-symbolic representation, which can be efficiently stored in biologically motivated associative memories.

Acknowledgments

This paper is an extended version of the presentation during the NeSy'11 Workshop at IJCAI-11. The author would thank those present for the valuable discussion during the presentation and two anonymous reviewers for their valuable suggestions. This work was supported by Fundação para a Ciência e Tecnologia (FCT) (INESC-ID multiannual funding) through the PIDDAC Program funds.

References

- Anderson, J. A. (1995a). *An Introduction to Neural Networks*. The MIT Press.
- Anderson, J. R. (1995b). *Cognitive Psychology and its Implications*. W. H. Freeman and Company, fourth edition.
- Ballard, D. H. (1997). *An Introduction to Natural Computation*. The MIT Press.
- Bentz, H. J., Hagstroem, M., and Palm, G. (1989). Information storage and effective data retrieval in sparse matrices. *Neural Networks*, 2(4):289–293.
- Cardoso, A. and Wichert, A. (2010). Neocognitron and the map transformation cascade. *Neural Networks*, 23(1):74–88.
- Churchland, P. S. and Sejnowski, T. J. (1994). *The Computational Brain*. The MIT Press.
- Fukushima, K. (1980). Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern*, 36(4):193–202.

- Fukushima, K. (1989). Analysis of the process of visual pattern recognition by the neocognitron. *Neural Networks*, 2:413–420.
- Fuster, J. (1995). *Memory in the Cerebral Cortex*. The MIT Press.
- Gilovich, T. (1999). Tversky. In *The MIT Encyclopedia of the Cognitive Sciences*, pages 849–850. The MIT Press.
- Goldstone, R. (1999). Similarity. In *The MIT Encyclopedia of the Cognitive Sciences*, pages 763–765. The MIT Press.
- Gross, C. and Mishkin (1977). The neural basis of stimulus equivalence across retinal translation. In Harnad, S., Dorty, R., Jaynes, J., Goldstein, L., and Krauthamer, editors, *Lateralization in the nervous system*. Academic Press, New York.
- Hecht-Nielsen, R. (1989). *Neurocomputing*. Addison-Wesley.
- Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley.
- Knoblauch, A., Palm, G., and Sommer, F. (2010). Memory capacities for synaptic and structural plasticity. *Neural Computation*, 22:289–341.
- Kohonen, T. (1989). *Self-Organization and Associative Memory*. Springer-Verlag, 3 edition.
- Kosslyn, S. M. (1994). *Image and Brain, The Resolution of the Imagery Debate*. The MIT Press.
- Lancy, D. (1983). *Cross-Cultural Studies in Cognition and Mathematics*. Academic Press, New York.
- McClelland, J. and Rumelhart, D. (1985). Distributed memory and the representation of general and specific memory. *Journal of Experimental Psychology: General*, 114:159–188.
- Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press.
- Newell, A. and Simon, H. (1976). Computer science as empirical inquiry: symbols and search. *Communication of the ACM*, 19(3):113–126.
- Opwis, K. and Plötzner, R. (1996). *Kognitive Psychologie mit dem Computer*. Spektrum Akademischer Verlag, Heidelberg Berlin Oxford.
- Osherson, D. N. (1987). New axioms for the contrast model of similarity. *Journal of Mathematical Psychology*, 31:93–103.
- Osherson, D. N. (1995). Probability judgment. In Smith, E. E. and Osherson, D. N., editors, *Thinking*, volume 3, chapter two, pages 35–75. MIT Press, second edition.
- Palm, G. (1982). *Neural Assemblies, an Alternative Approach to Artificial Intelligence*. Springer-Verlag.
- Palm, G. (1990). Assoziatives Gedächtnis und Gehirntheorie. In *Gehirn und Kognition*, pages 164–174. Spektrum der Wissenschaft.
- Posner, M. I. and Raichle, M. E. (1994). *Images of Mind*. Scientific American Library, New York.

- Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025.
- Rumelhart, D. and McClelland (1986). On learning the past tense of english verbs. In McClelland, J. and Rumelhart, D., editors, *Parallel Distributed Processing*, pages 216–271. MIT Press.
- Simon, H. A. (1991). *Models of my Life*. Basic Books, New York.
- Smith, E. E. (1995). Concepts and categorization. In Smith, E. E. and Osherson, D. N., editors, *Thinking*, volume 3, chapter one, pages 3–33. MIT Press, second edition.
- Sommer, F. T. (1993). *Theorie neuronaler Assoziativspeicher*. PhD thesis, Heinrich-Heine-Universität Düsseldorf, Düsseldorf.
- Squire, L. R. and Kandel, E. R. (1999). *Memory. From Mind to Moleculus*. Scientific American Library.
- Steinbuch, K. (1961). Die Lernmatrix. *Kybernetik*, 1:36–45.
- Steinbuch, K. (1971). *Automat und Mensch*. Springer-Verlag, fourth edition.
- Sun, R. (1995). A two-level hybrid architecture for structuring knowledge for commonsense reasoning. In Sun, R. and Bookman, L. A., editors, *Computational Architectures Integrating Neural and Symbolic Processing*, chapter 8, pages 247–182. Kluwer Academic Publishers.
- Tarski, A. (1956). *Logic, Semantics, Metamathematics*. Oxford University Press, London.
- Tversky, A. (1977). Feature of similarity. *Psychological Review*, 84:327–352.
- Wichert, A. (2001). Pictorial reasoning with cell assemblies. *Connection Science*, 13(1).
- Wichert, A. (2009). Sub-symbols and icons. *Cognitive Computation*, 1(4):342–347.
- Wichert, A. (2011). The role of attention in the context of associative memory. *Cognitive Computation*, 3(1).
- Wichert, A., Pereira, J. D., and Carreira, P. (2008). Visual search light model for mental problem solving. *Neurocomputing*, 71(13-15):2806–2822.
- Wickelgren, W. A. (1969). Context-sensitive coding, associative memory, and serial order in (speech)behavior. *Psychological Review*, 76:1–15.
- Wickelgren, W. A. (1977). *Cognitive Psychology*. Prentice-Hall.
- Willshaw, D., Buneman, O., and Longuet-Higgins, H. (1969). Nonholgraphic associative memory. *Nature*, 222:960–962.
- Winston, P. H. (1992). *Artificial Intelligence*. Addison-Wesley, third edition.