

**A Hubel Wiesel Model of Early Concept
Generalization Based on Local Correlation of
Input Features**

SEPIDEH SADEGHI
(B. Sc., Iran University of Science & Technology)

**A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE**

2011

Acknowledgements

I would like to express my genuine gratitude to Dr. Kiruthika Ramanathan, from Data Storage Institute (D.S.I), for her support and encouragement in the research and the preparation of this thesis. Through her leadership, insightful advice and excellent judgment, I was able to increase my basic knowledge of analysis and commit to research in the area of my interest.

I would like to express my gratitude to Professor Chong Tow Chong - my supervisor from National University of Singapore (N.U.S) -, and Dr. Shi Luping - my supervisor from D.S.I - for reviewing the progress of my project. I am also thankful to Singapore International Graduate Award (S.I.N.G.A) and D.S.I for providing me such wonderful project opportunity and the financial support throughout the course of the project. Appreciation is also extended to Electrical and Computer Engineering department at National University of Singapore.

I also thank all my friends from N.U.S and D.S.I for the excellent company they gave during the course of the project. I would also like to thank all my friends in Singapore who made my stay a wonderful experience.

Last, but not least, I am grateful to my parents, and sisters, whose devotion, support, and encouragement have inspired me and been my source of motivation for graduate school.

Table of Contents

| | |
|-------------------------------------------------------|-----------|
| 1. Introduction | 1 |
| 1.1 On Concepts and Generalization | 1 |
| 1.2 Background and Related Studies | 2 |
| 1.2.1 Concept acquisition and generalization | 2 |
| 1.2.2 Hubel Wiesel models of memory | 4 |
| 1.2.3 Hubel Wiesel models of concept representation | 6 |
| 1.3 Objective of the Thesis | 7 |
| 1.4 Summary of the Model | 8 |
| 1.5 Organization of the Thesis | 11 |
| 2. Methodology | 14 |
| 2.1 System Architecture | 15 |
| 2.1.1 Architecture | 15 |
| 2.1.2 Bottom up hierarchical learning | 19 |
| 2.2 Hypothesis | 22 |
| 2.3 Local Correlation Algorithm | 27 |
| 2.3.1 Marking features/modules as general or specific | 31 |

| | |
|-------------------------------------------------------------------------------------|-----------|
| 2.3.2 Generalization | 33 |
| 2.3.2.1 Input management | 33 |
| 2.3.2.2 Prioritization | 35 |
| 2.3.3 The effect of local correlation model on the categorization of single modules | 36 |
| 3. Results and Discussions | 39 |
| 3.1 Two Types of Input Data | 39 |
| 3.2 Generalization | 46 |
| 3.3 Local Correlation Operations and Computational Parameters | 49 |
| 3.4 Building Hierarchical Structures of Data | 55 |
| 4. Conclusion | 61 |
| 4.2 Concluding Remarks | 61 |
| 4.3 Future Works | 63 |
| Bibliography | 66 |
| Appendix A-1: Dataset A - List of Entities | 73 |
| Appendix A-2: Dataset B - List of Entities | 75 |
| Appendix A-3: Dataset C - List of Entities | 77 |

A Hubel Wiesel Model of Early Concept Generalization Based on Local Correlation of Input Features

Sepideh Sadeghi

Submitted on JAN 21, 2011

In Partial Fulfillment of the Requirements for the
Degree of Master of Engineering in Electrical and Computer Engineering

Abstract

Hubel Wiesel models, successful in visual processing algorithms, have only recently been used in conceptual representation. Despite the biological plausibility of a Hubel-Wiesel like architecture for conceptual memory and encouraging preliminary results, there is no implementation of how inputs at each layer of the hierarchy should be integrated for processing by a given module, based on the correlation of the features. If we assume that the brain uses a unique Hubel Wiesel like architecture to represent the input information of any modality, it is important to account for the local correlation of conceptual inputs as an equivalent to the existing local correlation of visual inputs in the visual counterpart models. However, there is no intuitive local correlation among the conceptual inputs. The key contribution of this thesis is the proposal of an input integration framework that accounts for the local correlation of the conceptual inputs in a Hubel Wiesel like architecture to facilitate the achievement of broad and coherent concept categories at the top of the hierarchy. The building blocks of our model are two algorithms: 1) Bottom-up hierarchical learning algorithm, and 2) Input integration framework. The first

algorithm handles the process of categorization in a modular and hierarchical manner that benefits from competitive unsupervised learning in its modules. The second algorithm consists of a set of operations over the input features or modules to weigh them as general or specific to specify how they should be locally correlated within the modules of the hierarchy. Furthermore, the input integration framework interferes with the process of similarity measurement applied by the first algorithm such that, high-weighted features would count more than the low-weighted features towards the similarity of conceptual patterns. Simulation results on benchmark data admit that implementing the proposed input integration framework facilitates the achievement of the broadest coherent distinctions of conceptual patterns. Achieving such categorizations is a quality that our model shares with the process of early concept generalization. Finally, we applied our proposed model of early concept generalization iteratively over two sets of data, which resulted in the generation of finer grained categorizations, similar to progressive differentiation. Based on our results, we conclude that the model can be used to explain how humans intuitively fit a hierarchical representation for any kind of data.

Keywords: Early Concept Generalization, Hubel Wiesel Model, Local Correlation of Inputs, Categorization, General Features, Specific Features.

Thesis Supervisors:

1. Prof. Chong Tow Chong, National University of Singapore, and Singapore University of Technology and Design.
2. Dr. Shi Luping, Senior Scientist, Data Storage Institute.

List of Tables

| | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Table 3.1 Features and their f_w values sorted in a decreasing order..... | 43 |
| Table 3.2 Features and their weights..... | 44 |
| Table 3.3 Datasets used in the simulations..... | 46 |
| Table 3.4 The effect of growth threshold on the quality of categorization biasing, using <i>max-weight</i> operation over dataset B (7 modules at the bottom layer)..... | 52 |
| Table 3.5 The effect of growth threshold on the quality of categorization biasing, using <i>sum-weights</i> operation over dataset B (7 modules at the bottom layer)..... | 52 |
| Table 3.6 Summary of the experiments..... | 54 |
| Table 4.1 The effect of decreasing growth threshold on the categorization of the local correlation model | 63 |

List of Figures

| | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Figure 1.1 The flow chart of the bottom-up algorithm - Hubel Wiesel model of early concept generalization. The highlighted rectangles demonstrate local correlations operations..... | 10 |
| Figure 1.2 The flow chart of the top-down algorithm – to model progressive differentiation..... | 11 |
| Figure 2.1 Hierarchical structure of the learning algorithm when the data includes 12 features.. | 18 |
| Figure 2.2 (a) Inputs and outputs to a single module $m_{k,i}$, (b) the concatenation of information from the child modules of the hierarchy to generate inputs for the parent module..... | 21 |
| Figure 2.3 General features versus specific features..... | 23 |
| Figure 2.4 Bug-like patterns used in [15], and the corresponding labeling rules for the categorization task..... | 25 |
| Figure 2.5 Inputs and outputs of the child modules. The outputs of child modules are the inputs to the parent module..... | 30 |
| Figure 2.6 (a) A set of patterns and their corresponding features, (b) features sorted in non-increasing order on the basis of their N_j values, (c) features are marked according to the value of τ | 32 |
| Figure 2.7 (a) The use of general, specific and an intermediate features (low weighted general features) in each module when the number of features per module is odd, (b) the use of general and specific features when the number of features per module is even..... | 35 |
| Figure 2.8 (a) ‘canary’ as an animal is mistakenly grouped with ‘pine’ as a plant when prioritization and input management are not included, (b) substituting the specific feature ‘walk’ with the general feature ‘root’ fixes the categorization due to inclusion of input management, (c) | |

canary' as animal is mistakenly grouped with plants when prioritization and input management are not included, (d) applying prioritization, fixes categorization to be coherent.....37

Figure 3.1 single features divide the pattern space into two groups.....40

Figure 3.2 Unique structured data. (1) Categorization of the patterns on the basis of f_j must be similar to the categorization on the basis of f_i , or (2) only one of the previous categories built on the basis of f_i is divided on the basis of f_j41

Figure 3.3 Input patterns.....43

Figure 3.4 The hierarchical structure of data in Figure 3.3 when features: 'Is blue' and 'Is orange' are disregarded44

Figure 3.5 The hierarchical structure of the right branch in Figure 3.4 when the categorization is biased on the basis of shape.....45

Figure 3.6 The hierarchical structure of the right branch in Figure 3.4 when the categorization is biased on the basis of color.....45

Figure 3.7 (a) the most frequent/common outcome categorization of dataset A by local correlation model – successful categorization, (b) Illustrating the probability of successful categorization over set A, being obtained in a set of trials using *sum-weights*, *max-weight* and no correlation model under different hierarchies of learning. Each probability demonstrates the ratio of the number successful categorizations obtained over 10 trials carried out using a specific correlation operation and under specific hierarchy of learning.....48

Figure 3.8 (a) The most frequent categorization of dataset C by local correlation model – successful categorization, (b) Illustrating the probability of successful categorization over set C, being obtained in a set of trials using *sum-weights* and *max-weight* operations under different hierarchies of learning. Each probability is computed in the same way as explained in (3.7 b)...49

Figure 3.9 The probability of categorization in Figure 3.7(a) over dataset A. A comparison of *sum-weights* and *max-weight* under different growth thresholds (8 learning modules at the bottom layer).....51

Figure 3.10 The probability of categorization in Figure 3.8(a) over dataset C. using *max-weight* operation under different growth thresholds in different hierarchical structures.....51

Figure 3.11 Hierarchical structure of dataset A.....57

Figure 3.12 Hierarchical structure of dataset C.....58

Figure 3.13 Temporal (cycle) and spatial (hierarchy) relationships of seasons and months.....58

Figure 3.14 (a) Less abstraction in the categorization, (b) higher levels of abstraction in the categorization due to the use of the non-leaf concept ‘~ mammals’.....59

List of Abbreviations

| | |
|------------|-----------------------------|
| SOM | Self Organizing Map |
| GSOM | Growing Self Organizing Map |

List of Symbols

| | |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \mathbf{P}_i | i th input pattern in the input data set |
| f_j | j th feature in an input vector or in a neuron weight vector |
| $f_{i,j}$ | j th feature in the i th input pattern |
| $activation_{i,j,k}$ |Activation of the i th neuron in the j th module in response to the k th input pattern |
| input Matrix $_i$ |Training matrix for the module number i |
| $m_{k,i}$ | i th module in the k th level of the hierarchy |
| N_j |Presence number of feature f_j |
| w_j |Weight of the j th input feature or module |
| τ |Ratio of the number general features to the number of specific features input to each module at the bottom most layer of the hierarchy |
| μ |Ratio of the number of general modules to the number of specific modules input to each module at the intermediate or top most layers of the hierarchy |
| \mathbf{Q} |Queue of inputs (features or modules) for level k of the hierarchy |
| \mathbf{G} |Queue of general inputs (features or modules) for level k of the hierarchy, used for marking |
| \mathbf{S} |Queue of specific inputs (features or modules) for level k of the hierarchy, used for marking |
| $nFeature$ |Capacity of a module in terms of the number of features it may receive |
| $nChild$ |Capacity of a module in terms of the number of child modules it may receive |
| $nModule(i)$ |Number of modules at level i |

nLevel.....Level number. It equals to one at the bottom of the hierarchy and increases moving upwards in the hierarchy

nSpecific.....Number of specific features available in **S**

gNum.....Number of general features in a module

sNum.....Number of specific features in a module

Max-weight.....One implementation of the local correlation algorithm

Sum-weights.....One implementation of the local correlation algorithm

Ceiling(i).....Function that returns the smallest integer that follows *i*

Floor(i).....Function that returns the largest integer that precedes *i*

Chapter 1

Introduction

1.1 On Concepts and Generalization

Concepts are the most fundamental constructs in theories of the mind. In psychology, a wide variety of questionable definitions of concepts exist such as “should concepts be thought of as bundles of features, or do they embody mental theories?” or “are concepts mental representations, or might they be abstract entities?” [1]. In our thesis, we define a concept as a mental representation which partially corresponds to the words of the language. We further assume that a concept can be defined as a set of typical features [2].

We adopt the following definitions.

- 1. Concept categorization** is the process by which the concepts are differentiated.
- 2. Concept generalization** is the categorization of concepts into less specific and broader categories.
- 3. Early concept generalization** is the early stage of progressive differentiation of concepts [3], in which children acquire broad semantic distinctions.

Concept generalization is one of the primary tasks of human cognition. Generalization of new concepts (conceptual patterns) based on prior features (conceptual features) leads to categorization judgments that can be used for induction. For example, given that an entity has certain features including: four legs, two eyes, two ears, skin, and ability to move, one may generalize that the entity (specific concept) is an animal instance (general concept). Therefore, the process of generalization leads to the category judgments (being an animal instance) about the object. Based on the category to which the object belongs, we can induce some hidden properties of the concept. For example, given that a conceptual entity belongs to the category of animals, we can induce that the entity eats, drinks and sleeps.

In recent years, research in computational cognitive science has served to reveal much about the process of concept generalization [3-5].

1.2 Background and Related Studies

This section is divided into two sub-sections. The first part discusses the state of art in the field of concept acquisition and generalization, and the second part describes research in the field of Hubel Wiesel models of memory.

1.2.1 Concept acquisition and generalization

The idea of feature based concept acquisition and generalization has been well studied in the psychological literature. Vygotsky [6], Inhelder and Piaget [7] first

proposed that the representation of categories develop from immature representations that are based on accidental features (appearance similarities). Recent theoretical and practical developments in the study of mature categorization indicate that generalization is grounded on perceptual mechanisms capable of detecting multiple similarities [3, 8-10].

Tests such as the trial task [11] show the role of feature similarity in the generation of categorization. Further to this, works by McClelland and Rogers [3], Rumelhart [9, 12] etc. show evidence for bottom up acquisition of concepts in memory. Sloutsky [13-15] discuss how children group concepts based on, not just one, but multiple similarities and how such multiple similarities tap the fact that basic level categories have correlated structures (or features). The correlation of features is also discussed in McClelland and Rogers [3] where they refute Quillian's classic model [16] of a semantic hierarchy where concepts are stored in a hierarchy progressing from specific to general categories. They argue that general properties of objects should be more strongly bound to more specific properties than to the object itself. Furthermore, McClelland and Rogers argue that information should be stored at the individual concept level rather than at the super ordinate category level. Only under this condition, properties can be shared by many items. They cite the following example: Many plants have leaves, but not all do – pine trees have needles. If we store 'has leaves' with all plants, then we must somehow ensure that it is negated for those plants that do not have leaves. If instead we store it only with plants that have leaves, we cannot exploit the generalization. McClelland and Rogers counter propose a parallel distributed

processing (PDP) model, which is based on back propagation, and test it using 21 concepts, including trees, flowers, fish, birds and animals. Their network showed progressive differentiation. Progressive differentiation phenomenon refers to the fact that children acquire broader semantic distinctions earlier than more fine-grained distinctions [5]. Our model falls under the umbrella of bottom-up architectures, but is bio-inspired (within a Hubel Wiesel architecture) and explains categorization and progressive differentiation, accounting for local correlation of input features.

1.2.2 Hubel Wiesel models of memory

It is well known that the cortical system is organized in a hierarchy and that some regions are hierarchically above others. Further to this, Mountcastle [17, 18] showed that the brain is a modular structure and the cortical column is its fundamental unit. A hierarchical architecture has been found in various parts of the neocortex including the visual cortex [19-23], auditory cortex [24, 25] and the somato-sensory cortex [26, 27]. In addition to this, neurons in the higher levels of the visual cortex represent more complex features with neurons in the IT representing objects or object parts [28, 29].

On the spectrum of cognitively inspired architectures, Hubel Wiesel models are designed for object recognition. Beginning from the Neocognitron [30, 31] to HMAX [19, 20, 32, 33], SEEMORE [34], various bio inspired hierarchical models has been used for object recognition and categorization. The primary idea of these models is a hierarchy of simple (S) and complex (C) cells, inspired by

visual cortex cells. For example, in visual cortex each S cell responds selectively to particular features in the receptive field. Therefore, the S cell is a feature extractor which, at the lower levels, extracts local features and, at the higher layers, extracts global features. C cells allow for positional errors in the features. Therefore, a C cell is more invariant to shift in position of the input pattern. The combination of S cells and C cells, whose signals propagate up the hierarchy allows for scale and position invariant object recognition.

The Neocognitron [30, 31] applies the principles of hierarchical S and C cells to achieve deformation resistant character recognition. Neocognitron uses a competitive network to implement the S and C cells, following a winner-take all update mechanism. HMAX is a related model based on a quantitative theory of the ventral stream of the visual cortex. Similar to Neocognitron, HMAX uses a combination of supervised and unsupervised learning to perform object categorization, but uses Gabor filters to extract primitive features. HMAX has been tested on benchmark image sets such as the Caltech 101 and the Streetscenes database. Lecun et al [35] have implemented object categorization using multi layered convoluted networks. All these mentioned models are deep hierarchical networks that are trained using back propagation. Wallis and Rolls [36-38] showed that increasing the number of hierarchical levels leads to an increase in invariance and object selectivity. Wersing and Koener [39] discuss the effects of different transfer functions over the sparseness of the data distribution in an unsupervised hierarchical network. Wolf et al [40] discuss alternative hierarchical

architectures for visual models and test their strategies on the Caltech 101 database.

1.2.3 Hubel Wiesel models of concept representation

In a recent work Ramanathan et al [41] have extended Hubel Wiesel models of the visual cortex [20, 32] to model concept representation. The resulting architecture, trained using competitive learning units arranged in a modular, hierarchical fashion, shares some properties with the Parallel Distributed Processing (PDP) model of semantic cognition [3]. To our knowledge, this is the first implementation of a Hubel Wiesel approach to non- natural medium such as text, and has attempted to model hierarchical representation of keywords to form concepts.

Their model exploits the S and C cell configuration of Hubel Wiesel models by implementing a bottom up, modular, hierarchical structure of concept acquisition and representation, which lays a possible framework for how concepts are represented in the cortex.

However the architecture of this model is similar to that of visual Hubel Wiesel models, there's still a gap between the process of feature extraction and integration in their model and the one in its counterpart visual models. In the existing visual models, small patches of the picture are input to the S cells where neighboring S cells extract neighboring patches of the picture. Then, C cells integrate several neighboring S cells. The neighborhood of the visual inputs

within small patches extracted by S cells and the neighborhood of the small patches integrated in C cells explain a coherent a local correlation of inputs preserved all over the hierarchy. On the other hand, in the conceptual Hubel Wiesel model proposed by Ramanathan et al [41], there is no provision to account for the local correlation of inputs and how it should be preserved through the hierarchy.

1.3 Objectives of the Thesis

The objective of this dissertation is to capture the quality of early concept generalization and progressive differentiation of concepts within a Hubel Wiesel architecture that accounts for local correlation of inputs and category coherence. Category coherence [42] refers to the quality of a category being natural, intuitive and useful for inductive inferences. We assume that preserving the natural correlation of inputs through the hierarchy is the necessary condition for the achievement of coherent categories at the top level of the hierarchy. Definition of such correlations in visual models is intuitive - spatial neighborhood -, while being a challenge in conceptual models. If we assume that the brain uses a hierarchical Hubel Wiesel like architecture to represent concepts, it is important to account for this local correlation factor. Moreover, it is likely that the categorization results at the top level of the hierarchy are dependent on the input integration framework of the hierarchy. Hence, we argue one possible metric based on which a local correlation model among conceptual features can be

achieved. Then, we propose an input integration framework to maintain such correlation through hierarchy.

Interestingly, it was observed that the proposed correlation model along with its corresponding input integration framework succeed to facilitate the achievement of coherent categorization - which admits our prior assumption in this regard. The proposed model not only effectively captures coherent categorization but also ensures revealing of the broadest differentiation of its conceptual inputs. Based on our literature survey, revealing the broadest differentiation is one of the qualities of early concept generalization. Therefore, our model shares this quality with early concept generalization. The flow chart of our model of early concept generalization is presented in Figure 1.1. Based on our knowledge about concept generalization, first it facilitates acquiring of broad distinctions and only as a matter of time leads to acquiring of the finer distinctions. This flow is called progressive differentiation of concepts which can also be captured by our model. The top-down iterative use of the proposed model over a data set and its corresponding subsets (broad categories generated by the model) results in creation of finer categories, similar to progressive differentiation. The flow chart of this top-down algorithm is presented in Figure 1.2.

1.4 Summary of the Model

Figure 1.1 illustrates the flow chart of the bottom-up algorithm for Hubel Wiesel model of early concept generalization proposed in this work. The details of the

model are presented in chapter 2. Figure 1.2 demonstrates the top-down algorithm which uses the bottom-up model iteratively to achieve finer categories similar to progressive differentiation. The details of this procedure are explained in section 3.4.

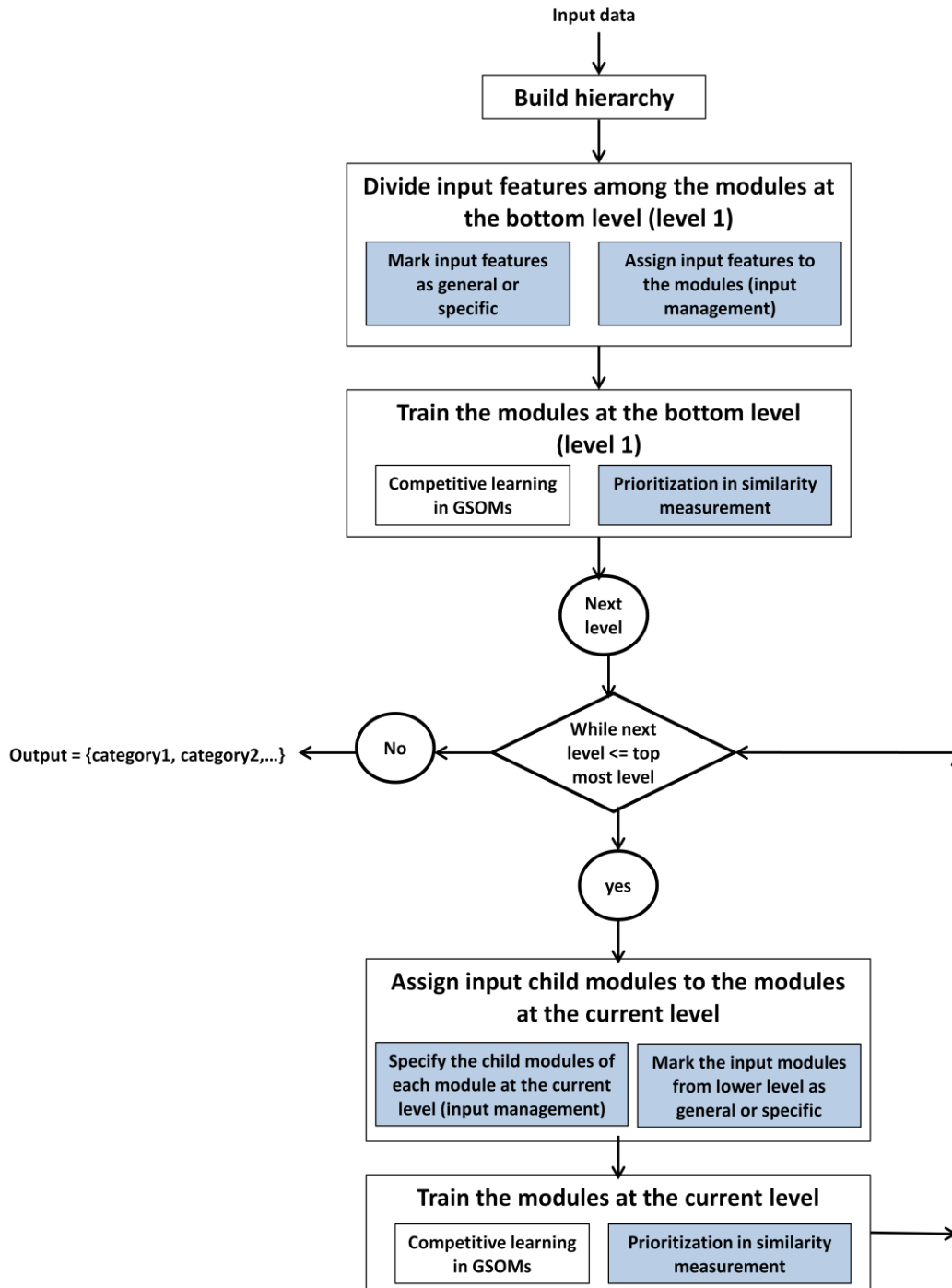


Figure 1.1: The flow chart of the bottom-up algorithm - Hubel Wiesel model of early concept generalization. The highlighted rectangles demonstrate local correlations operations.

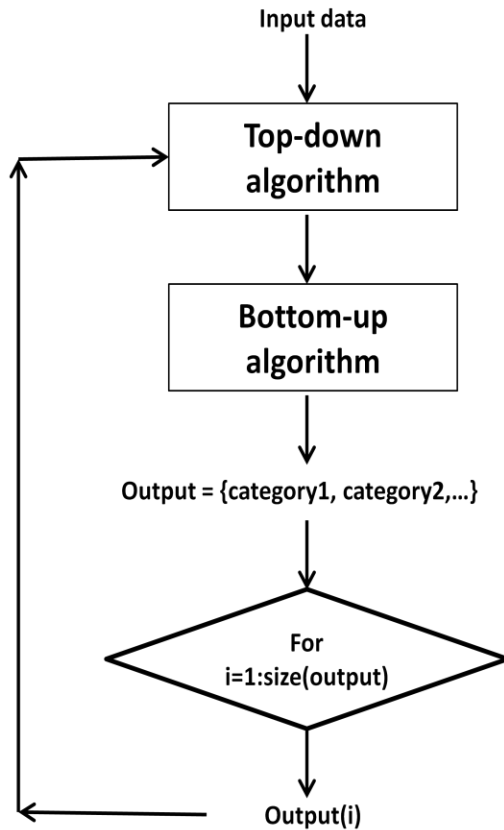


Figure 1.2: The flow chart of the top-down algorithm – to model progressive differentiation.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows:

- Chapter 2 presents the methodology to enable Hubel Wiesel model to obtain coherent broad categorization of concepts.
- Chapter 3 illustrates the impacts of applying the proposed input integration framework to a Hubel Wiesel conceptual model. It presents the

results over various datasets while counting for the effect of related computational parameters on the strength of the impacts.

- Chapter 4 presents concluding remarks and the future recommendations to improvise the proposed bottom-up model and simulate the next stages of the progressive differentiation of concepts within the bottom-up pass.

Chapter 2

Methodology

This Chapter presents the detailed description of the approach by which we captured the quality of early concept generalization within a Hubel Wiesel like architecture equipped with our proposed input integration framework. The building blocks of our model are two algorithms: 1) Hierarchical learning algorithm, and 2) Input integration algorithm corresponding to the proposed local correlation model – we may use ‘local correlation algorithm/model’ or ‘input integration algorithm’ interchangeably to refer to this algorithm. Local correlation algorithm extracts the correlated input features (at the bottom layer) and the correlated input child modules (at the intermediate layers) and groups them in batches. Each module will receive one of these batches as its inputs.

This chapter is divided into three broad sections: 1) System Architecture, 2) Hypothesis, and 3) Local Correlation Algorithm. Section 1 presents the details of the architecture and hierarchical learning algorithm. Sections 2 and 3 detail the proposed local correlation model along with the hypothesis behind that.

2.1. System Architecture

2.1.1 Architecture

The system that we describe here is organized in a bottom up hierarchy. This means that the conceptual features are represented before the representation of conceptual patterns. Our learning algorithm exploits the property of this hierarchical structure. Each level in the hierarchy has several modules. These modules model cortical regions of concept memory. The modules are arranged in a tree structure, having several children and one parent. In this dissertation, we call the bottom most level of the hierarchy level 1, and the level number increases from bottom to top of the hierarchy. Each conceptual pattern is defined as a binary vector of conceptual features, where 1 encodes relevance and 0 encodes irrelevance of the corresponding feature to the target pattern. A matrix of all the pattern vectors is directly fed to level 1 as the input. Level 1 modules resemble simple cells of the cortex, in the sense that they receive their inputs from a small patch of the input space. In our model, the input features are distributed amongst the modules at Level 1. Several level 1 modules tile the feature space. A module at level 2 covers more of the feature space when compared to a level 1 module. It represents the union of the feature space of all its child modules from level 1. A level 2 module obtains its inputs only through its level 1 children. This pattern is repeated in the hierarchy. Thus, the module at the tree root (the top most level) covers the entire feature space, but it does so by pooling the inputs from its child

modules. In our model, level 1 can be considered analogous to the area V1 of the visual cortex, level 2 to the area V2 and so on.

The below pseudo code illustrates how the hierarchical levels and their modules are created in this work. The modules are not interconnected within a level k . The connections between the modules in the level k to the modules in level $(k+1)$ and level $(k-1)$ would be specified by local correlation algorithm. $nFeature$ encodes the number of features allowed in each module – module capacity. M encodes the total number of features in the input data. $nChild$ encodes the number of children allowed for the parent modules – though it is not a constraint and some modules might receive $(nChild+1)$ child modules. $nModule(i)$ represents the number of modules created at level i and $nLevel$ represent the level number.

Build hierarchy()

1. $nLevel = 1$
 2. $nModule(nLevel) = ceiling(M/nFeatures)$
 3. $n = nModule(nLevel)$
 2. *while* ($n > 1$)
 - a. $nLevel = nLevel + 1$
 - b. $nModule(nLevel) = floor(nModule(nLevel-1)/nChild)$
 - c. $n = nModule(nLevel)$
-

Figure 2.1 demonstrates the inputs of the learning modules and the propagation of their outputs within the hierarchy through an example. In this figure, rectangles

demonstrate learning modules and the circles demonstrate the generated neurons inside them after their training is finished. Some modules and neurons are numbered to be referred in the explanation of the following example.

The input data to this hierarchical structure would be a matrix of n conceptual patterns each of which being defined as a binary vector of m features. Therefore, the input data is a $m \times n$ binary matrix, where each column encodes a pattern. The element (i, j) in such matrix corresponds to the correlation of the feature f_i and the pattern \mathbf{P}_j . The value of this element is encoded by $f_{j,i}$ and is equal to one if the feature f_i is correlated with the pattern \mathbf{P}_j , otherwise it equals to zero. The modules at the bottom of the hierarchy extract subsets of such input matrix and apply them as their input matrixes. Suppose that the input data includes 4 patterns and 12 features. Furthermore, assume that the number of features allowed per module as a user defined parameter is set to 3.

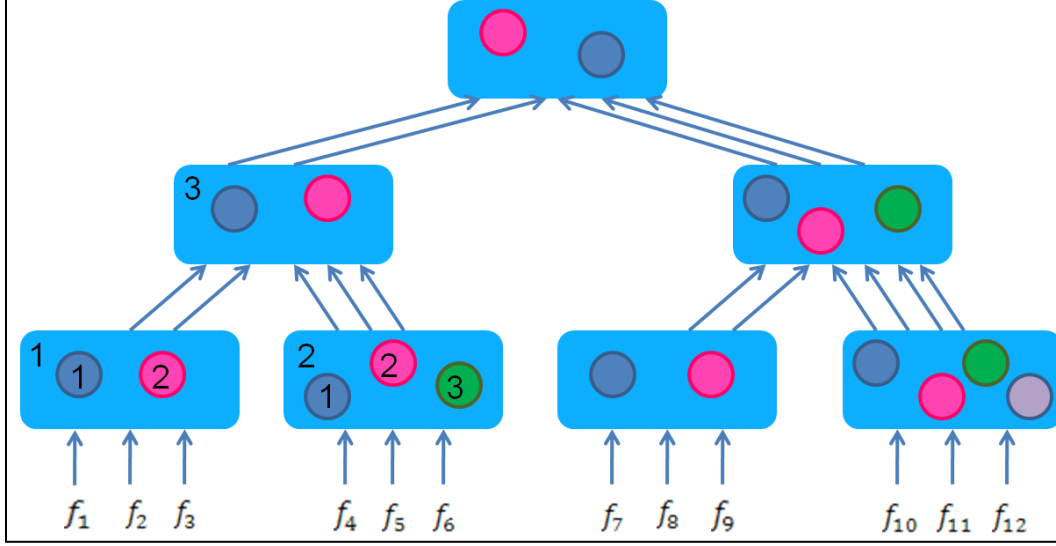


Figure 2.1: Hierarchical structure of the learning algorithm when the data includes 12 features.

$$\mathbf{input\ Matrix}_1 = \begin{bmatrix} f_{1,1} & \dots & f_{4,1} \\ f_{1,2} & \dots & f_{4,2} \\ f_{1,3} & \dots & f_{4,3} \end{bmatrix}, \quad \mathbf{input\ Matrix}_2 = \begin{bmatrix} f_{1,4} & \dots & f_{4,4} \\ f_{1,5} & \dots & f_{4,5} \\ f_{1,6} & \dots & f_{4,6} \end{bmatrix} \quad (2.1)$$

Equation 2.1 shows the corresponding input matrixes to modules 1 and 2 as exemplar input matrixes for the modules from the bottom layer. A number of neurons would be generated in each module after it finishes training using its input matrix. In our hierarchical system training will be carried out layer by layer, starting from the bottom most layer. When all the modules from layer 1 finish training, the training for the layer 2 will start. In order to train the modules from layer 2 we need to generate the input matrixes for the modules at this layer. In this endeavor, once again all the bottom modules would be exposed to the input patterns. After exposure to each of these input patterns, one neuron will be fired inside each of the modules. Therefore, the exposure to each pattern generates a

specific pattern of activations across the bottom modules. Such generated activation patterns would be used as the corresponding input to level 2 modules. They represent the original input pattern seen at level 1 for the modules at level 2. To illustrate how the outputs of child modules function as the inputs for the parent modules, let us consider the child modules 1 and 2 and the parent module 3. Equation 2.1 shows the input matrixes for modules 1 and 2. Module 1 has 2 neurons inside and module 2 has 3 neurons inside. The corresponding activation values of all these 5 neurons belonging to the children of module 3 function as the inputs to this module. Equation 2.2 illustrates such input matrix for module 3. The activation value of the neuron number i , inside module number j , in response to pattern number k is encoded by $activation_{i,j,k}$.

$$\mathbf{input\ Matrix}_3 = \begin{bmatrix} activation_{1,1,1} & \dots & activation_{1,1,4} \\ activation_{2,1,1} & \dots & activation_{2,1,4} \\ activation_{1,2,1} & \dots & activation_{1,2,4} \\ activation_{2,2,1} & \dots & activation_{2,2,4} \\ activation_{3,2,1} & \dots & activation_{3,2,4} \end{bmatrix} \quad (2.2)$$

2.1.2 Bottom up hierarchical learning

In our model, learning is managed in an unsupervised manner by the learning modules throughout the hierarchy. A variation of Self Organizing Map (SOM) is used to implement the learning modules. SOM is an unsupervised neural network which traditionally is used to map high dimensional data to low (2 or 3) dimensional data. The number of neurons in a SOM is fixed and predetermined. Therefore, often it is needed to run the learning algorithm several times for a

particular data to find out the appropriate number of neurons to present the data. To avoid this problem and provide more flexibility in our learning modules, we use Growing Self Organizing map (GSOM) [43] as the learning modules in our model. GSOM explained in [43] is a variation of SOM which allows the neurons inside the module to grow. It starts with a very small grid of neurons and generates the new neurons only on the basis of need. GSOM applies a user defined parameter “growth threshold” to control the growth of the neurons inside the module. When the distance between a new input pattern and all the existing spatial centers of data - neurons’ weights - in the module is more than the growth threshold, a new neuron would be generated. In our implementations, the initial number of neurons in each GSOM is two.

To understand how the model learns, let us consider the inputs and outputs of a single module $m_{k,i}$ in level k of the system as shown in Figure 2.2(a). Let \mathbf{x} , representing connections $\{x_j\}$ be the input pattern to the module $m_{k,i}$. \mathbf{x} is the output of the child modules of $m_{k,i}$ from the level $k-1$, and \mathbf{a} represent the weights of the competitive network. The vector \mathbf{a} is used to represent the connections $\{a_j\}$ between \mathbf{x} and the neurons in the module $m_{k,i}$ - neuron weight. The output of a neuron in $m_{k,i}$ in response to an input $\{a_j\}$ is, 1 if the Euclidean distance between its weight vector and the input is the least compared with other neurons in the module. Otherwise, the output would be zero. The outputs of the neurons being 0 or 1 are called activation values.

During learning, each neuron in $m_{k,i}$ competes with other neurons in the vicinity. Of the large number of inputs to a given module, a neuron is activated by a subset

of them using a winner takes all mechanism. The neuron then becomes the spatial center of these patterns.

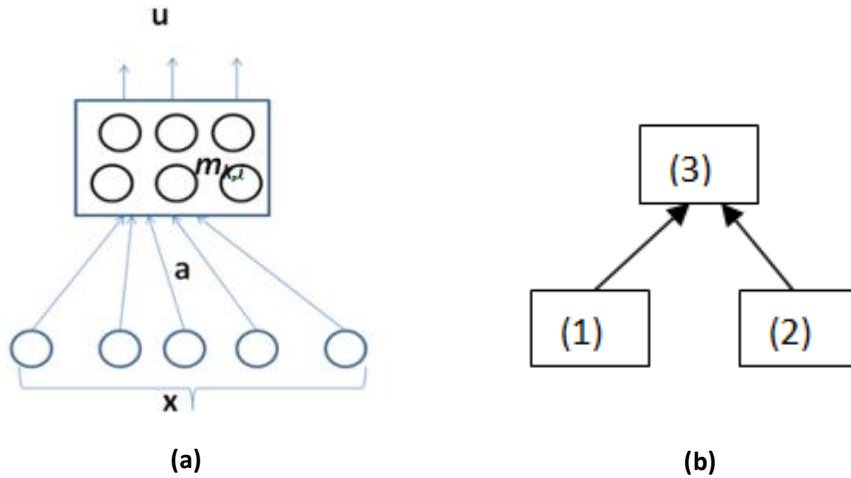


Figure 2.2: (a) Inputs and outputs to a single module $m_{k,i}$; (b) the concatenation of information from the child modules of the hierarchy to generate inputs for the parent module.

When all the modules at level k finish training, the subsequent stage of learning occurs. This comprises the process by which the parent modules learn from the outputs of the child modules. Here, consider the case shown in Figure 2.2(b) where the module 3 is the parent of modules 1 and 2. Let $\mathbf{x}(1)$ be the output vector of module 1 and $\mathbf{x}(2)$ be the output vector of module 2. $\mathbf{x}(i)$ represents a vector of activation values being the outputs of the neurons in the child modules. The input to module 3, $\mathbf{I}(3) = \mathbf{x}(1) || \mathbf{x}(2)$, is the concatenation of the outputs of modules 1 and 2. A particular concatenation represents a simultaneous occurrence of a combination of concepts in the child module. Depending on the statistics of the input data, some combinations will occur more frequently, while others will not. During this stage of learning, the parent module learns the most frequent

combinations of concepts in the levels below it. A GSOM is again used in the clustering of such combinations. The learning process thus defined can be repeated in a hierarchical manner.

2.2. Hypothesis

This section presents the hypothesis based on which the local correlation model of input features is proposed. It further explains all the assumptions, key-facts and empirical psychological evidences based on which we hypothesized this model.

As it was discussed in chapter 1, there is no intuitive correlation among the conceptual features. There are too many contexts with respect to which a correlation model among concepts can be captured. In this work, we focus on the concept correlations in the context of the concept categories.

Representative features of a category can be qualitatively regarded as general or specific [3]. General features are more commonly perceived among the members of the category. On the other hand, specific features are only associated with specific members of the category. Therefore, general features are better representatives of a category compared with specific ones. Subsequently, In the process of generalization, general features are weighed over specific features.

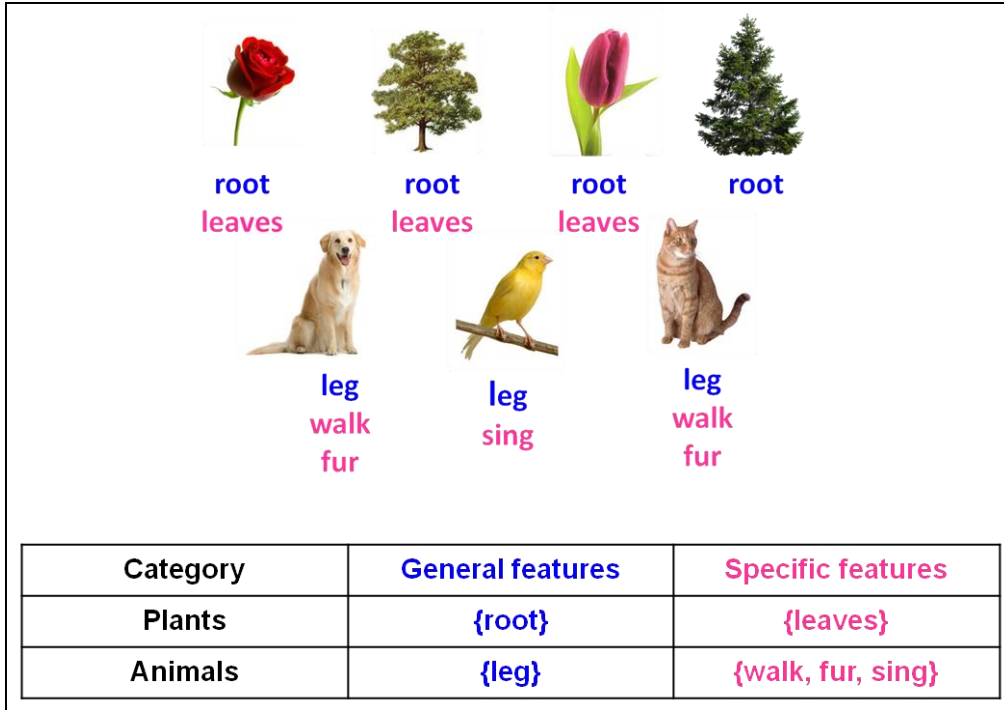


Figure 2.3: General features versus specific features.

In order to generalize a pattern, the similarity of the pattern and existing categories will be compared and the pattern will be assigned to the most similar category. To measure the similarity of a pattern and a category we compare the features of the category and the pattern while giving more weight to the similarity of general features in comparison with specific features. Consequently, the process of generalization needs prior knowledge about the existing categories and their corresponding general and specific features. However in early generalization, no prior knowledge about the categories and their general and specific features is available. Our hypothesis describes how prior knowledge about general and specific features (in early generalization) is built up. It further

explains the mechanism by which this prior knowledge is used to build up early categories.

Sloutsky et al [15] examined the underlying mechanism of early induction in light of comparing the role of appearance similarity¹ and kind information² – labeling rules. Figure 2.4 demonstrates the four bug-like patterns which they used to pit appearance similarity against labeling rules in the process of early induction. As can be seen in Figure 2.4, from the appearance point of view (the shape and color of the antennas, hands, fingers, bodies, and tails), the given patterns can be categorized into two pairs of patterns: (a,b) and (c,d). On the other hand, based on kind information provided in Figure 2.4, patterns are categorized differently resulting in a different set of pairs: (a,c) and (b,d). Based on the findings reported in this work, children of four or five years of age ignored the provided labeling rules in the course of induction, relying instead on the appearance similarities. Hence, they concluded that early induction is more biased towards the appearance features rather than kind information features.

¹ Visual similarities.

² Hand coded labeling rules to be used for categorization and induction of hidden attributes of a set of bug like patterns designed by the authors. Use of these labeling rules needed the children to compare the number of fingers with the number of buttons in each pattern. Use of labeling rules were designed to devise a different categorization from the one devised by appearance similarity.

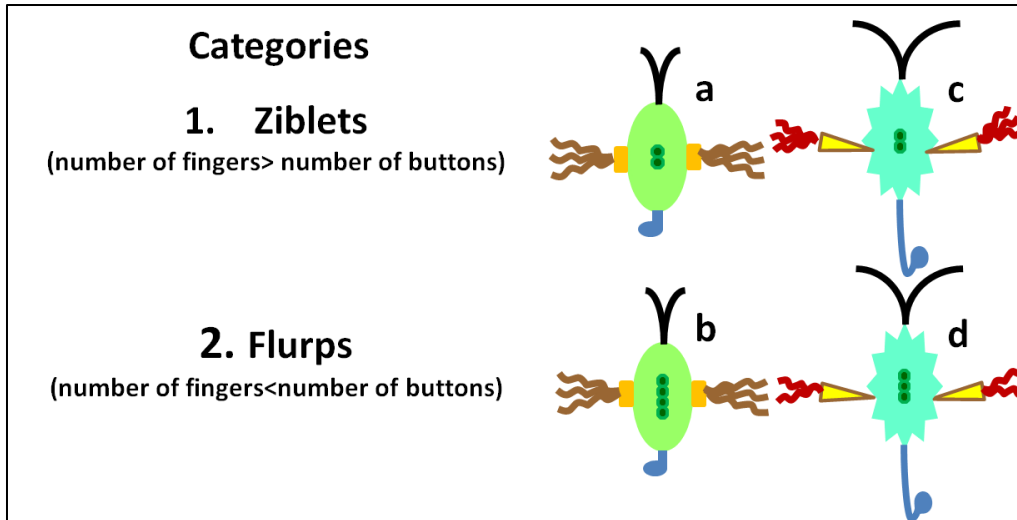


Figure 2.4: Bug-like patterns used in [15], and the corresponding labeling rules for the categorization task.

Since the process of induction uses the natural categorization judgments resulted by the process of generalization, we can conclude that early generalization is more governed on the basis of appearance similarities versus kind information. In below, this conclusion with regard to the details of experiments done by Sloutsky et al [15] is justified from two different points of view. Then, the justifications are used and generalized to present a hypothesis about the process of early generalization.

- Based on our assumption stated above, generalization is governed on the basis of general features of the categories. General features are the features which are more frequently perceived with the corresponding categories. Therefore, it is important to note that the children of four or five years of age are more frequently exposed to visual features rather than abstract knowledge and consequently the amount of their prior abstract knowledge

(abstract features) is considerably smaller than the amount of their prior visual knowledge (visual features). Hence, visual features being regarded as general in comparison with abstract features (like labeling rules) are more effective in the process of generalizations – natural and coherent categorizations that can be used for induction – made by the subjects of the study.

- In the set of patterns presented to the subjects of the study, the number of visual features leading to the categorization of $\{(a,b),(c,d)\}$ is more than the number of labeling rules leading to the categorization of $\{(a,c),(b,d)\}$. Therefore, the first categorization is consistently supported by more number of input features rather than the second categorization.

Based on all above, we hypothesize that,

1. In early generalization, the more frequently perceived prior features are regarded as general.
2. Weighting general features over specific ones (less frequently perceived features) leads to the detection of the broad distinctions of the observed patterns in the domain of the subject's prior knowledge (known features).

A parallel can be drawn between these hypotheses and Sloutsky's work [15] in that the frequently perceived appearance inputs being regarded as general features are weighted over kind information being regarded as specific features, making categorization biased towards appearance similarity.

2.3. Local Correlation Algorithm

We propose a model of local correlation of features which implements our hypothesis in the context of the Hubel Wiesel conceptual memory proposed in Ramanathan et al [41]. This model defines the correlation of input features based on the quality of features being general or specific. The proposed model accomplishes two tasks through the bottom-up hierarchy. (a) It marks the inputs of each layer of the hierarchy as general or specific and (b) It biases the categorization of each module on the basis of its general inputs.

The inputs of the model at the bottom most layer are vectors of conceptual features and at the intermediate layers are vectors of activation values generated by the neurons of the child modules. In order to mark the inputs as general or specific, we first need to weight the generality of each input. In this endeavor, we define two parameters: a) feature weight: to weight the generality of the conceptual features at the bottom layer, and b) module weight: computed for each child module in the hierarchy to weight the generality of its output activation values input to a parent module. In this case all the activation values output from a module would be equally weighted by the computed weight value for the module when being input to the parent module.

In each module, the input vectors and the weight vectors of the neurons are of the same dimension. For each element (feature/activation value) being a member of input vector, there is a weight value which will be incorporated to the model as a coefficient magnifying or trivializing the similarity of the given element in the

input vector and the corresponding element in the neuron weight. Therefore feature/module weights are different from the neuron weights that are used for training. However they determine what are the most important elements of the neuron weight vector, which need to be similar to the elements in the input vector for the neuron to be activated.

Let **inputData** = $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n\}$ where $\forall i: 1 \leq i \leq n$, \mathbf{P}_i represents an input pattern such that, $\mathbf{P}_i = (f_{i,1}, f_{i,2}, \dots, f_{i,m})$ and $\forall j: 1 \leq j \leq m$, $f_{i,j}$ is 1 if the feature is present in \mathbf{P}_i and 0 otherwise. The *Presence Number* N_j for each feature f_j is

$$N_j = \sum_{i=1}^n f_{i,j} \quad (2.3)$$

The *feature weight* w_j for each input feature and *module weight* w_m for each module within the hierarchy is defined as follows.

$$w_j = \frac{N_j}{n} \quad (2.4)$$

We compute w_m for a module as a function of its input weights. Two different operations for computing w_m are presented and compared in our paper – *sum-weights* and *max-weight*.

Where M represents the modules at level $p-1$ of the hierarchy, the *sum-weights* operation defines w_m at level p as

$$w_m = \frac{\sum_{k \in \text{inputs of } m} w_k}{\sum_{i \in M} w_i} \quad (2.5)$$

Whereas the *max-weight* operation evaluates w_m as

$$w_m = \frac{\max(w_k), k \in \text{inputs of } m}{\sum_{i \in M} w_i} \quad (2.6)$$

The *max-weight* operation is expected to have a greater bias of the results towards general features and broader categorizations than the *sum-weights* operation.

In the context of this thesis we refer to local correlation algorithm as ***max-weight algorithm***, if *max-weight* operation is used and we refer to it as ***sum-weights algorithm*** if *sum-weights* operation is used.

Each bottom or intermediate level module feeds to a higher level module (parent) and correspondingly each of its outputs (categories) functions as an input feature to the parent module. The w_j values of such inputs to the parent module are equal to the w_m values of the child modules originating them. In order to illustrate how w_j values are computed for the inputs at different layers of the hierarchy in the context of the two proposed operations - *sum-weights* and *max-weight* operations -, an example is provided in below.

Suppose that $module_1$ and $module_2$ are located at the bottom of the hierarchy.

Let n be the number of patterns in the database.

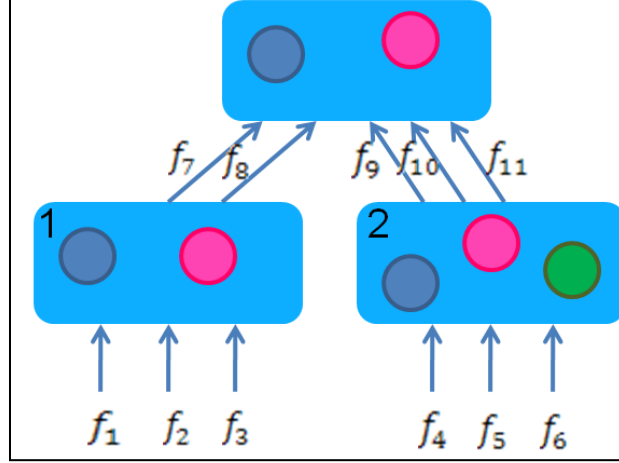


Figure 2.5: Inputs and outputs of the child modules. The outputs of child modules are the inputs to the parent module.

► $\forall j, 1 \leq j \leq 6; w_j = \frac{N_j}{n}$ (Bottom layer) (2.7)

► $w_7 = w_8 = w_{module_1}$ (Layers other than the bottom layer)

&

$w_9 = w_{10} = w_{11} = w_{module_2}$ (2.8)

• $w_{module_1} = \max(w_1, w_2, w_3)$ (*max-weight* algorithm)

&

$w_{module_2} = \max(w_4, w_5, w_6)$ (2.9)

• $w_{module_1} = \sum_{j=1}^3 w_j$ (*sum-weights* algorithm)

&

$w_{module_2} = \sum_{j=4}^6 w_j$ (2.10)

2.3.1. Marking features/modules as general or specific

The weight value of each feature or module represents its generality or specificity as seen by the system. In short, the higher the weight value, the more general the feature/module.

User defined parameters τ (for feature marking), and μ (for module marking) are applied to control the population of general set relative to the population of specific set. It is desired to keep the number of general features always higher than the number of specific features. At the bottom layer of the hierarchy, the number of general features will be set to be τ times the number of specific features and at the intermediate layers the number of general modules will be set to be μ times the number of specific modules. It is desired for the values of τ and μ to be greater than or equal to one, since they encode the ratio of the number of general to specific inputs (features/modules) of each layer of the hierarchy. For example, suppose the total number of features is equal to N and $\tau = 2$. So, it is desired to keep the number of general features twice as many as the number of specific features (since $\tau = 2$). Therefore, we assign the first $\text{ceiling}\left(\frac{\tau}{1+\tau} \times N\right)$ ³ number of the most high weighted features to the set of general features and the rest of the features to the set of specific features. Figure 2.6 illustrates one such example.

The pseudo code below is used to label a set of features/modules as being general or specific, depending on the user defined parameters τ (for feature marking) and μ (for module marking), where τ and μ are greater than or equal to 1.

³ *ceiling* maps a real number to its smallest following integer

Mark features/modules as general_specific()

1. Sort the features/modules in a decreasing order on the basis of their weights and push them into the queue **Q**

2. while $\sim(\text{isEmpty}(\mathbf{Q}))$

a. Pop τ/μ features/modules from the front of **Q** and push them into the queue **G** (general features/modules)

b. pop one feature/module from the rear of **Q** and push it into the queue **S** (specific features/modules)

Figure 2.6 illustrates the process of marking features at the bottom layer of the hierarchy through an example.

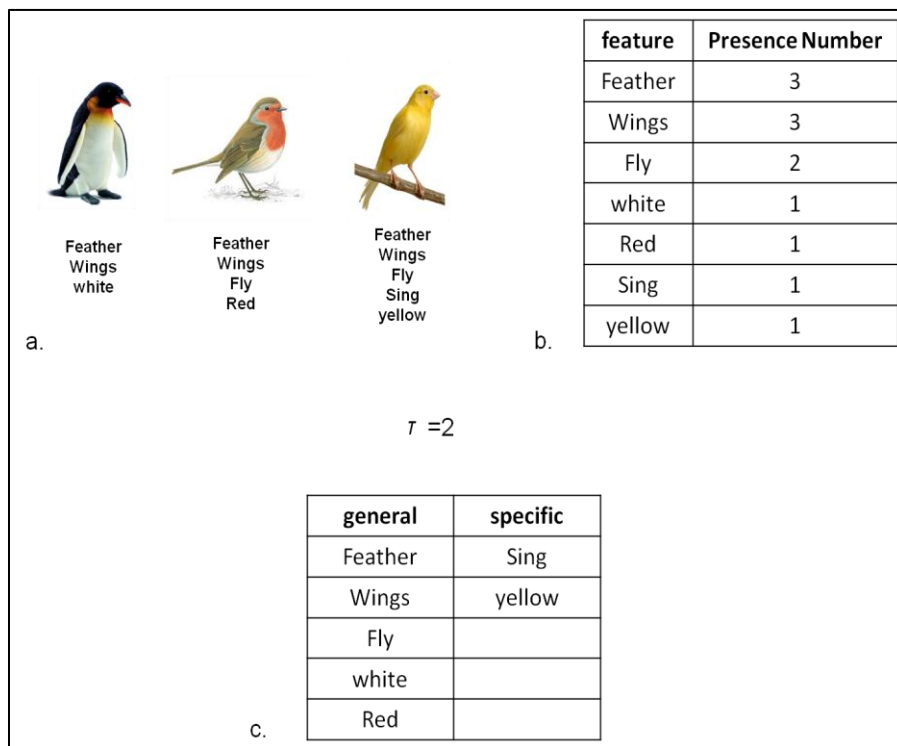


Figure 2.6: (a) A set of patterns and their corresponding features, (b) features sorted in non-increasing order on the basis of their N_j values, (c) features are marked according to the value of τ .

2.3.2. Generalization

In the process of generalization across the hierarchy, our model weighs general features/modules over specific ones by performing two main operations – input management and prioritization.

2.3.2.1. Input management

Input management ensures that the number of general features/modules input to each module of the hierarchy is greater or at least equal to the number of specific features/modules. The following pseudo code explains input management at the most bottom layer of the hierarchy with $\tau=2$. Let $nFeature$ represent the number of features per module. $nSpecific$ encodes the number of available features in the queue \mathbf{S} , including unused specific features. We desire to have a combination of general and specific features in each module so as to distribute the effect of general feature across the hierarchy. Hence, it is desired to input specific features into a module which shares a pattern with a general feature already added to the module. This is performed by *sharedPattern* which returns a Boolean indicating whether there is any pattern in which the values of the feature f_j and at least one of the previously added features of the module are one. The performance is dependent on the number of input features/children per module (user defined parameters) and the values of τ and μ .

Input features to the Module($nFeature$)

1. *if (isOdd($nFeature$))*
- a. *pop one feature from the rear of the queue \mathbf{G} and push it into the Module*
2. *for $i=1:\text{floor}(nFeature/2)$*
 - a. *pop one feature from the front of the queue \mathbf{G} and push it into the Module*
 - b. *if $\sim(\text{isEmpty}(\mathbf{S}))$*
 - i. *feature = **Pop specific(Module)***
 - ii. *push feature into the Module*
 - c. *else*
 - i. *pop one feature from the rear of the queue \mathbf{G} and push it into the Module*

Pop specific(Module)

1. *added = 0*
 2. *for $i= nSpecific:-1:1$*
 - a. *if (sharedPattern(Module, $\mathbf{S}(i)$))*
 - i. *pop $\mathbf{S}(i)$ from the queue \mathbf{S} and push it into the Module*
 - ii. *added = 1*
 - iii. *break*
 3. *if $\sim(\text{added})$*
 - a. *pop one feature from the rear of the queue \mathbf{S} and push it into the Module*
-

Figure 2.7 illustrates the process of input management at the bottom layer of the hierarchy in the context of the example demonstrated in Figure 2.6.

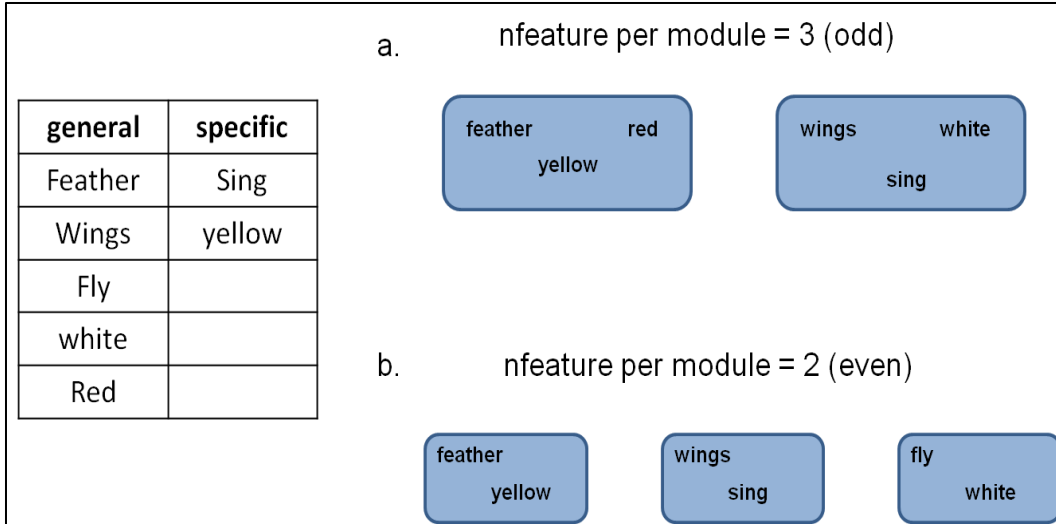


Figure 2.7: (a) The use of general, specific and an intermediate features (low weighted general features) in each module when the number of features per module is odd, (b) the use of general and specific features when the number of features per module is even.

2.3.2.2. Prioritization

Prioritization is a weighted similarity measure that interferes in the process of similarity measurement of the conceptual patterns. In this work, we define the similarity of any two concepts as the Euclidean distance between the representative neurons⁴. The prioritization operation magnifies or trivializes the similarity values of the pair-wise inputs on the basis of their corresponding weights. From equation 2.11, we can observe that the similarity values of general features with high feature weights would be more significant in the process selection of similar concepts and generalization. In equation 2.11, $gNum$ and $sNum$ represent the number of general and specific features in the module

⁴ As can be seen from equation 2.11, the effect of prioritization can be observed only when an integration of pair-wise feature similarity is used to measure concept similarity.

respectively, such that $gNum \geq sNum$. The indices P and N refer to the pattern (input vector) and the neuron (weight vector).

$distance(\mathbf{Pattern}, \mathbf{Neuron}) =$

$$\sqrt{\sum_{i=1}^{gNum} (f_{iP} - f_{iN})^2 \times w_i + \sum_{j=1}^{sNum} (f_{jP} - f_{jN})^2 \times w_j} \quad (2.11)$$

2.3.3 The effect of local correlation model on the categorization of single modules

Figure 2.7 illustrates an example of how effectively the inclusion of local correlation of input features leads to coherent - natural - categorizations in single modules. In the following example the value of τ is equal to one and therefore:

general fetatures = {*root, leaves, legs*} &

specific fetatures = {*walk, fur, sing*}

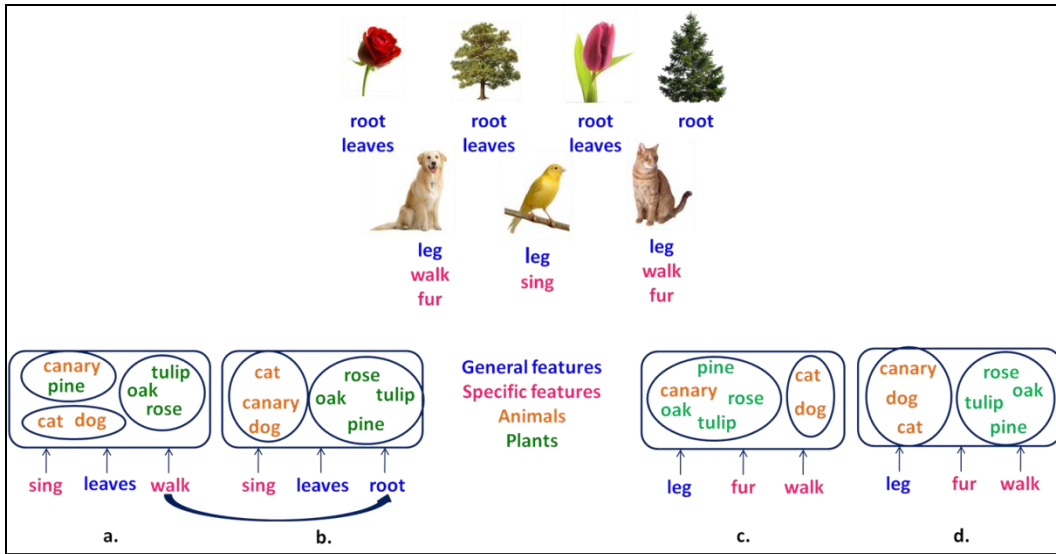


Figure 2.8: (a) ‘canary’ as an animal is mistakenly grouped with ‘pine’ as a plant when prioritization and input management are not included, (b) substituting the specific feature ‘walk’ with the general feature ‘root’ fixes the categorization due to inclusion of input management, (c) canary’ as animal is mistakenly grouped with plants when prioritization and input management are not included, (d) applying prioritization, fixes categorization to be coherent.

Chapter 3

Results and Discussions

In this chapter, two different types of input data for the model are discussed. Then, the experimental results of applying the model - *max-weight* and *sum-weights* operations - over three different sets of data, in the light of different computational parameters and different types of input data are presented. In the end, the model is applied for the discovery of the hierarchical structural form of data.

3.1. Two Types of Input Data

Every feature in a database can divide the pattern space of the data into two separate categories (Figure 3.1). The core of our model is to weight the corresponding categories of general features over specific features in the process of categorization. On this basis, two types of data structure namely ‘unique structured’ and ‘multiple structured’ can be discussed.

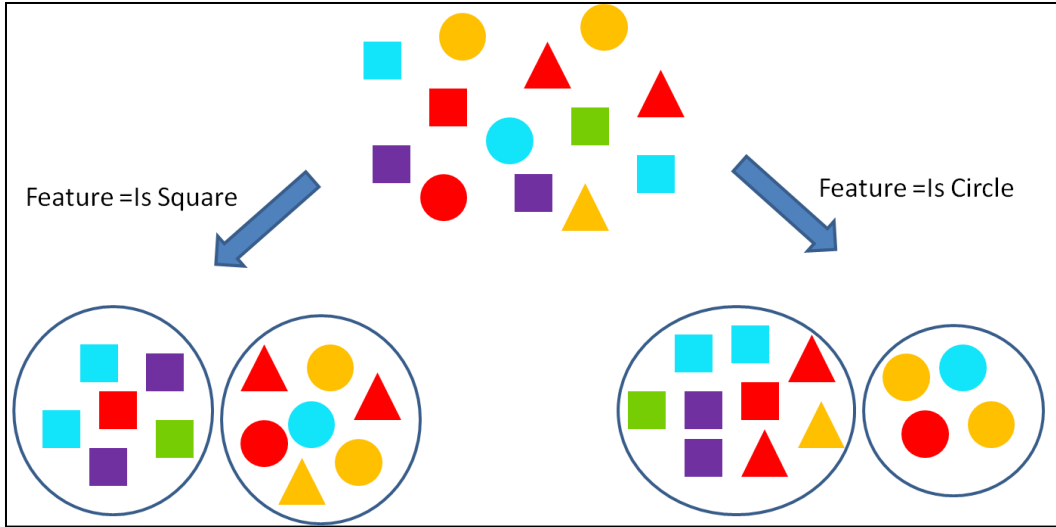
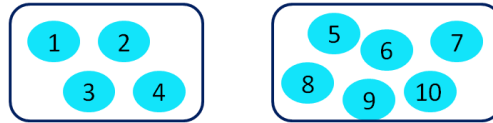


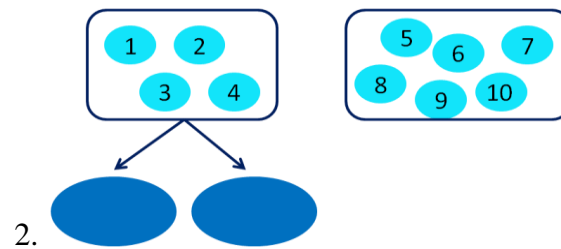
Figure 3.1: single features divide the pattern space into two groups.

We call a data unique structured if, for every two general features f_i and f_j from the database, where $w_j \leq w_i$, one the following conditions hold. Under first condition, both features should categorize the pattern space similarly. Under second condition, f_j should not divide the pattern space of more than one of the categories created on the basis of f_i . In below, the definition of the unique structured data is illustrated through an example. Suppose that **InputData** is the input matrix including n features and 10 patterns. **InputData** is labeled as a unique structured data if for each feature f_i and f_j from the dataset, it holds one of the conditions (1) or (2) as illustrated in Figure 3.2.

$\forall f_i, f_j \in InputData, w_j \leq w_i$, & Categorization of patterns on the basis of f_i :



Then, the categorization on the basis of f_j is similar to (1) or (2)



Or

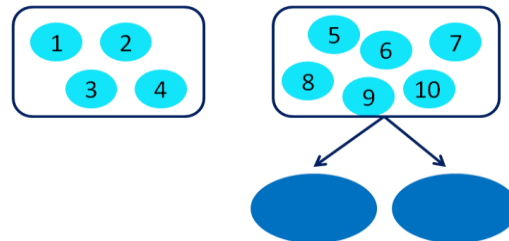


Figure 3.2: Unique structured data. (1) Categorization of the patterns on the basis of f_j must be similar to the categorization on the basis of f_i , or (2) only one of the previous categories built on the basis of f_i is divided on the basis of f_j .

In other words, unique structured data displays a binary hierarchical structure, since each feature divides the database into two groups and each new feature is only allowed to categorize one of the branches made by the last feature. In contrast, any data which does not fit a binary hierarchical structure or might possibly fit in multiple binary hierarchies fails to always hold one of the above conditions and is regarded as multiple structured data. An example is provided in below, to illustrate the structure of a unique structured data versus that of a multiple structured data.

Figure 3.3 demonstrates a set of patterns and Table 3.1 lists their corresponding features along with their w_j values. Figure 3.4 illustrates the progressive top-down categorization of the patterns based on their w_j values. The categorization starts on the basis of highest weighted feature and it continues using lower weighted ones. It is of importance to note that the progressive top-down categorization explained here is just to explain the difference between the two types of data. This type of categorization is not intended to be assumed as equivalent to the bottom-up generalization procedure explained in chapter 2. In fact, the iterative application of our model (bottom-up approach in chapter 2) over the entire pattern space and its newly emerged subsets (categories) should be assumed as equivalent to this top-down progressive categorization. As can be seen, using features: ‘Is square’, ‘Is circle’, ‘Is red’, ‘Is triangle’, and ‘Is purple’ in a sequential order keeps the conditions of a unique structured data satisfied. On the other hand, lower in the top-down hierarchy, using features with lower weights: ‘Is blue’ and ‘Is orange’ divide the pattern space of more than one previous categories built by

features of higher weights. This means that if for some reason our model selects features like ‘Is blue’ prior to features like ‘Is triangle’ (to bias the generalization on its basis) a different structure of data would emerge (not a unique structured data). For example, if the weights of such features (like ‘Is blue’) are not sufficiently lower than the previous features (like ‘Is triangle’), they might be weighted over higher weighted features – due to random initialization of neuron weights and the effect of weights on final categorization. Hence, in Figure 3.4 the structure of data stemming from the branch enclosed in a circle is not unique. Figure 3.5 and Figure 3.6 demonstrate the two possible hierarchical structures corresponding to this branch. Table 3.2 demonstrates the features corresponding to the patterns of this branch.

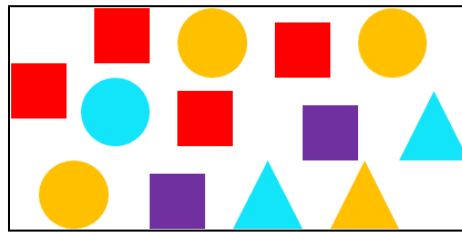


Figure 3.3: Input patterns.

Table 3.1: Features and their f_w values sorted in a decreasing order.

| Feature | f_w |
|----------------|-------------------------|
| Is square | 6/13 |
| Is circle | 4/13 |
| Is red | 4/13 |
| Is triangle | 3/13 |
| Is orange | 3/13 |
| Is blue | 3/13 |
| Is purple | 2/13 |

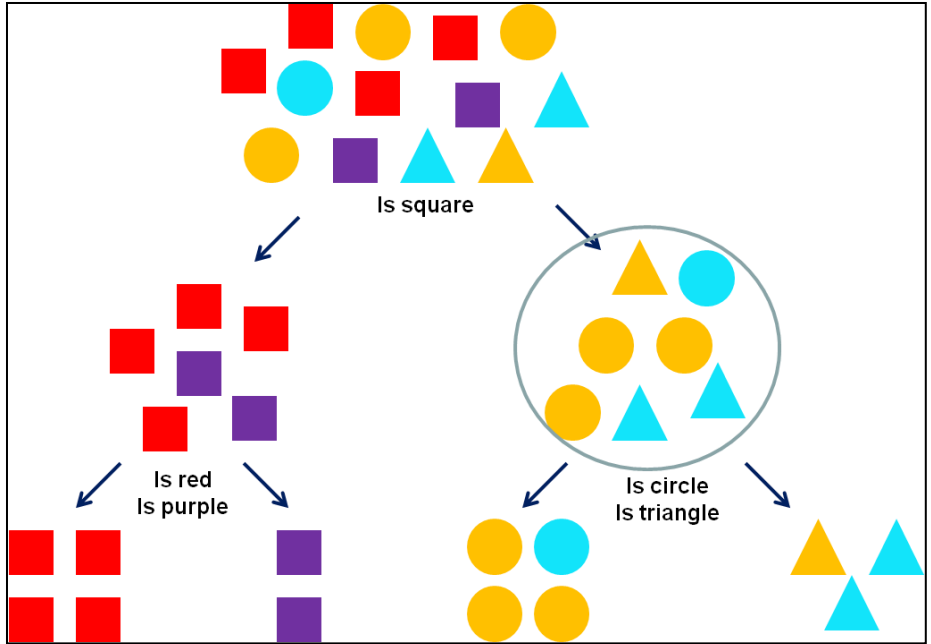


Figure 3.4: The hierarchical structure of data in Figure 3.3 when features: ‘Is blue’ and ‘Is orange’ are disregarded.

Table 3.2: Features and their weights.

| Feature | Weight |
|----------|--------|
| circle | 4/13 |
| triangle | 3/13 |
| orange | 3/13 |
| blue | 3/13 |

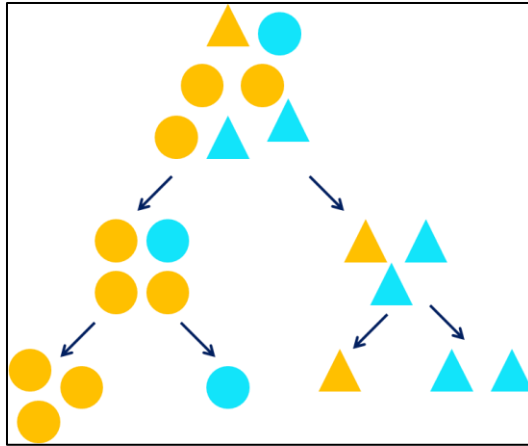


Figure 3.5: The hierarchical structure of the right branch in Figure 3.4 when the categorization is biased on the basis of shape.

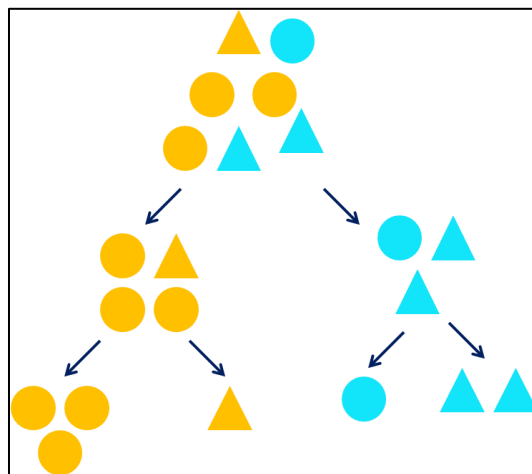


Figure 3.6. The hierarchical structure of the right branch in Figure 3.4 when the categorization is biased on the basis of color.

In the context of this article we call a data unique structured if the top most categorization - focusing on the broadest distinctions - of the patterns is unique in all different hierarchies corresponding to the data. Correspondingly, if different hierarchies of the data demonstrate contradicting categorizations at the top most level, we call the data multiple structured.

Table 3.3 illustrates three sets of data that have been applied in this thesis to verify the model. In all the experiments, the parameter τ is equal to 2 and the parameter μ is equal to 1 (each parent module is fed with one general module and one specific module).

Table 3.3. Datasets used in the simulations.

| Label | Source | Data type | Details |
|--------------|---------------|---------------------|-----------------------------|
| Set A | [4] | Unique structured | 21 patterns 26 features |
| Set B | [4] | Multiple structured | 13 patterns 14 features |
| Set C | [44] | Multiple structured | 33 patterns 102 features |

3.2. Generalization

Figures 3.7 and 3.8 illustrate the contribution of local correlation to the categorization results of the Hubel Wiesel conceptual memory over Set A and Set C. We tested the model under different hierarchical structures, initialized by different number of modules and different number of features per module at the bottom of the hierarchy. As can be seen the local correlation operations, regardless of the structure of the hierarchy and the type of the dataset (unique structured or multiple structured), successfully biases the categorization on the basis of the high weighted general features in the context of input data. For example, in Figure 3.7, the highest weighted features within input data are ‘root’,

and ‘move’. ‘move’ is the present feature in all animal instances and ‘root’ is the present feature in all plant instances. As can be seen, the final categorization (generalization) replicates the differentiation of the patterns on the basis of these general features. Our generalization method if applied on the natural data⁵ assures the achievement of the broadest coherent categorization. The resulted categorization over both set A and set C corresponds to the broadest biological distinction of their patterns. The categorization over set A reveals two basic kingdoms of patterns and the categorization over set C reveals two phylums of animals (Arthropods versus ~ Arthropods). Based on our results, when local correlation model is not included, the categorization of data is incoherent and also alternates per runtime.

⁵ We adopt a definition for natural data. Natural data refers to a data wherein the relative frequency (generality) of its features in the context of a its limited patterns is proportional to the relative frequency of those features in the context of unlimited real world patterns.

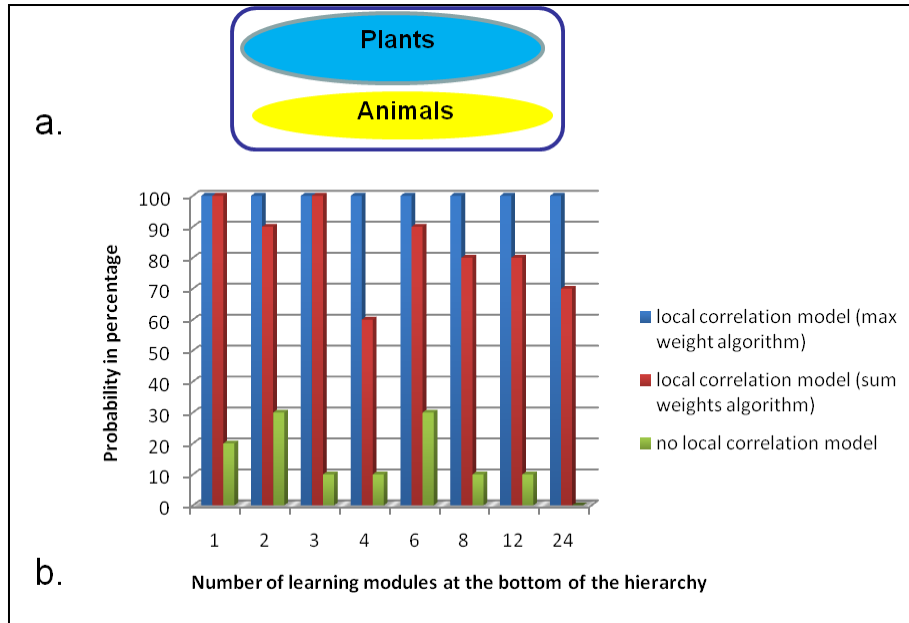


Figure 3.7: (a) the most frequent/common outcome categorization of dataset A by local correlation model – successful categorization, (b) Illustrating the probability of successful categorization over set A, being obtained in a set of trials using *sum-weights*, *max-weight* and no correlation model under different hierarchies of learning. Each probability demonstrates the ratio of the number successful categorizations obtained over 10 trials carried out using a specific correlation operation and under specific hierarchy of learning.

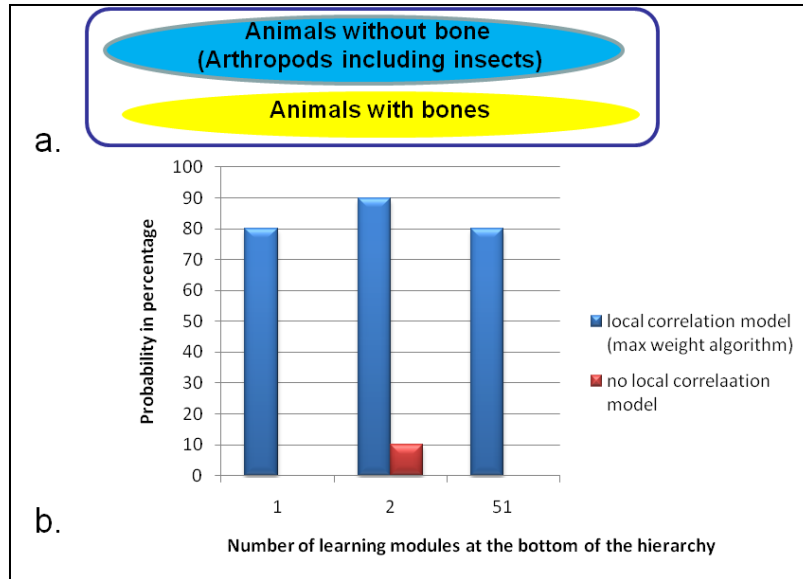


Figure 3.8: (a) The most frequent categorization of dataset C by local correlation model – successful categorization, (b) Illustrating the probability of successful categorization over set C, being obtained in a set of trials using *sum-weights* and *max-weight* operations under different hierarchies of learning. Each probability is computed in the same way as explained in (3.7 b).

3.3. Local Correlation Operations and Computational Parameters

In this section, we compare the categorization performance of the *sum-weights* and *max-weight* operations with respect to the effect of different computational parameters. Growth threshold [43] is a computational parameter used in the learning modules of our model. This parameter controls the growth of the neurons (categories) inside a module by applying a threshold on the distance values of the input patterns and the closest existing neuron weight in the module. If the corresponding distance value for an input pattern is larger than the threshold, a new neuron will be initialized in the modules. Therefore, lower values of growth

threshold facilitate the generation of more number of categories and consequently finer distinctions within the corresponding module.

Figures 3.9 and 3.10 illustrate the effect of the growth threshold over set A and set C. The specific categories obtained by applying two different correlation operations and various thresholds to the multiple structured set B, is shown in Tables 3.4 and 3.5.

According to Figure 3.7, Figure 3.9, Tables 3.4 and 3.5, regardless of the hierarchical structure, type of data and growth threshold values, the *max-weight* operation is always more significant than *sum-weights* in biasing the categorization. As can be seen, this conclusion is admitted by higher probability values reported for *max-weight* dominant categorizations in comparison with those reported for *sum-weights*.

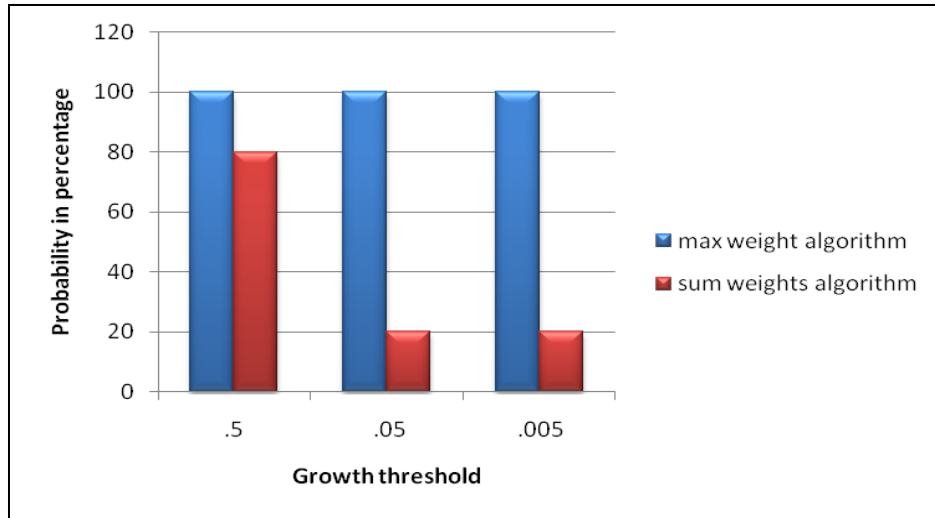


Figure 3.9: The probability of categorization in Figure 3.7(a) over dataset A. A comparison of *sum-weights* and *max-weight* under different growth thresholds (8 learning modules at the bottom layer)

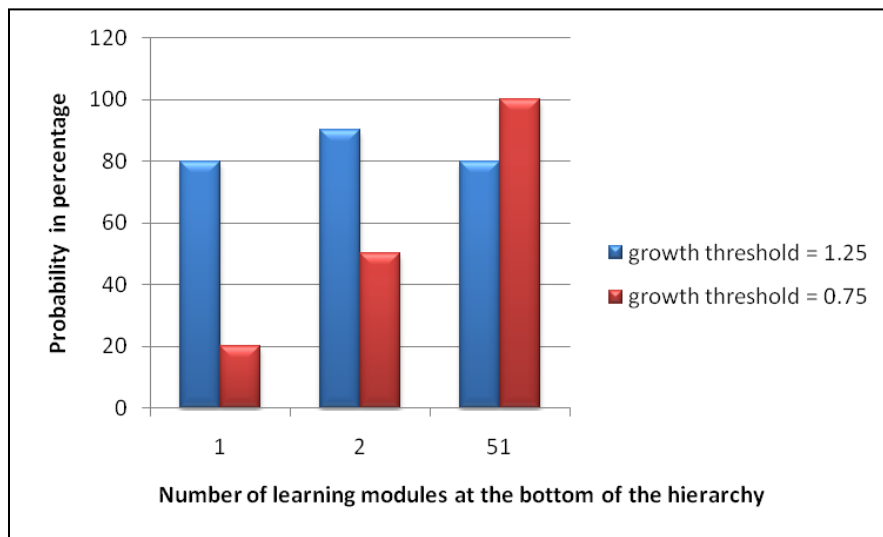


Figure 3.10: The probability of categorization in Figure 3.8(a) over dataset C. using *max-weight* operation under different growth thresholds in different hierarchical structures.

Table 3.4: The effect of growth threshold on the quality of categorization biasing, using *max-weight* operation over dataset B (7 modules at the bottom layer).

| Growth threshold | The most probable categorization (dominant categorization) | Probability of the dominant categorization |
|------------------|------------------------------------------------------------|--------------------------------------------|
| 0.5 | | 100% |
| 0.05 | | 60% |
| 0.005 | | 60% |

Table 3.5: The effect of growth threshold on the quality of categorization biasing, using *sum-weights* operation over dataset B (7 modules at the bottom layer).

| Growth threshold | The most probable categorization (dominant categorization) | Probability of the dominant categorization |
|------------------|------------------------------------------------------------|--------------------------------------------|
| 0.5 | | 50% |
| 0.05 | | 40% |
| 0.005 | | 40% |

As can be seen in Figures 3.9, using *sum-weights* over a unique structured data, probability of getting broad distinctions decreases with the decrease of growth threshold. However, this probability stays robust when using *max-weight* operation. On the other hand, according to Figure 3.10, applying *max-weight* over a multiple structured data, probability of getting broad distinctions does not stay robust against changes in growth threshold. It is also important to notice that in this case, the probability of getting broad distinctions does not necessarily decrease with the decrease of the growth threshold (3.10, 51 modules). This evidence, suggests that the geometry of the hierarchy is another effective factor that along with growth threshold and the structure of data influences the broadness and possibly coherence of the resultant categorization.

According to Tables 3.4 and 3.5, using *max-weight* operation over a multiple structured data the dominant categorization gets finer and more coherent (naturally descriptive of data) with the decrease of growth threshold. It is also noticeable that the same effect is not observed using *sum-weights*.

Table 3.6, summarizes the details and purpose of the experiments demonstrated in this section.

Table 3.6: Summary of the experiments.

| Experiment | | 1 | 2 | 3 | 4 | 5 | 6 |
|----------------------------|---------------------------------------------|--------------------------------------------------------------------------------------------|------------|---------------------------------------------------------------------------|-------------|-----------------------------------------------------------------|----------------|
| Purpose of experiment | | Comparison of the probability of broad and coherent categorizations using different models | | The effect of growth threshold on the probability of broad categorization | | The effect of growth threshold on the quality of categorization | |
| Illustration of results | | Figure 3.7 | Figure 3.8 | Figure 3.9 | Figure 3.10 | Table 3.4 | Table 3.5 |
| Hierarchical learning | Number of learning modules | 1,2,3,4,6,4,2,1 | 1,2,51 | 8 | 1,2,51 | 7 | 7 |
| | Number of input features allowed per module | 24,12,8,6,4,2,1 | 51,2,1 | 3 | 51,2,1 | 2 | 2 |
| Local correlation model | <i>Max-weight</i> algorithm | √ | √ | √ | √ | √ | --- |
| | <i>Sum-weights</i> algorithm | √ | --- | √ | --- | --- | √ |
| No local correlation model | | √ | √ | --- | --- | --- | --- |
| Data type | Unique Structured data | √ | --- | √ | --- | --- | --- |
| | Multiple Structured data | --- | √ | --- | √ | √ | √ |
| Growth threshold | | 0.5 | 0.5 | 0.5,0.05,0.005 | 1.25,0.75 | 0.5,0.05,0.005 | 0.5,0.05,0.005 |

3.4. Building Hierarchical Structures of Data

In this section we use the *max-weight* (a bottom-up generalization algorithm) in a top-down hierarchical manner to build a hierarchical structure of the data. Given a database, we first apply the model over whole data (the root node of the hierarchy) which results in the creation of several categories. Each of these categories - containing only a portion of input patterns along with their corresponding features would be regarded as a new dataset (nodes branching from the root node). We apply the proposed concept generalization model over new datasets (subsets of patterns) iteratively until the desired depth and breadth of the hierarchy in different branches is reached. The below pseudo code illustrates such top-down algorithm which uses the bottom-up algorithm to build a hierarchical structure of data. In this pseudo code, the function *bottom-up()* refers to our proposed Hubel Wiesel model of early concept generalization explained in chapter 2. Top-down function is a recursive function which uses the bottom-up function iteratively and results in finer categories similar to progressive differentiation.

Top-down(Input Data)

1. *output = Bottom-up(Input Data)*

2. *for i=1:size(output)*

a.Top-down(output(i))

{category1, category2, ...}=Bottom-up(Input Data)

The results of applying this procedure over dataset A, and dataset C are provided in Figure 3.11 and Figure 3.12. It is important to note that changing the growth threshold of the model can change the number of resulted categories (the number of branches stemming from the corresponding node).

We assume that humans are capable of performing categorization and subsequently labeling over any given set of patterns represented in the format of feature data [44]. Since, category labels can always be organized into hierarchies [45], therefore regardless of the underlying structural form of the data [44] human mind is considered to be capable of fitting any given feature data into a hierarchical structure. For example, geographical places are naturally organized in a spherical structural form, while human mind is capable of projecting geographical data in hierarchical structure through developing and using concepts like continent, country, state, and city. In other words, we assume that one of the cognitive properties of human mind is the ability to build hierarchical structure for any given feature data which makes it able to develop abstract but not necessarily natural knowledge about its environment.

Furthermore, it can be discussed that the same set of entities can be represented within different structural forms each of which capturing a different aspect of the relationship among the entities. For example, the temporal relationship among seasons, months, and weeks can be captured within cycles while their spatial relationship can be represented in hierarchies (Figure 3.13). Additionally, different spatial representations of the data within a given structural form reflect different levels of the abstraction of the patterns' relationship. For instance, a set

of animals including three different classes: mammals, birds, amphibians can be categorized in two different ways. As can be seen in Figure 3.14.b, animals can be categorized to two big groups of mammal and ~mammal at the top level of the hierarchy. ~mammal can be subsequently categorized into two classes: birds and amphibians. According to Figure 3.14.b ~mammal is not a leaf in the hierarchical structure and it only provides more abstraction of data. As can be seen in Figure 3.14.a, another way of categorization is to remove the middle abstract concept of ~mammal. In any kind of categorization there might be non-leaf concepts functioning as more abstractions of the data. Based on the number of abstract groups that might be desired different spatial representation of the data can be emerged within a particular structural form.

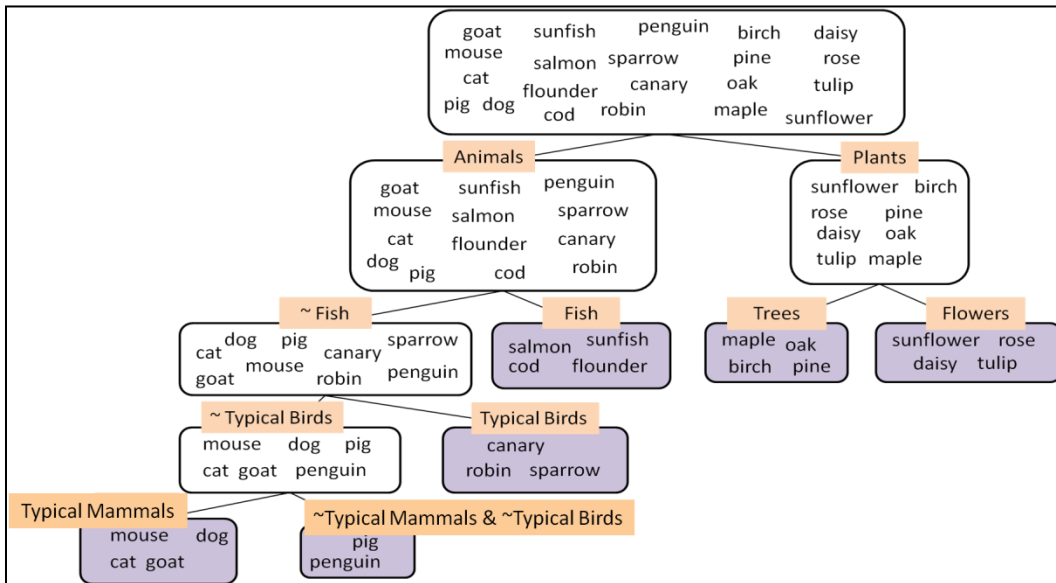


Figure 3.11: Hierarchical structure of dataset A.

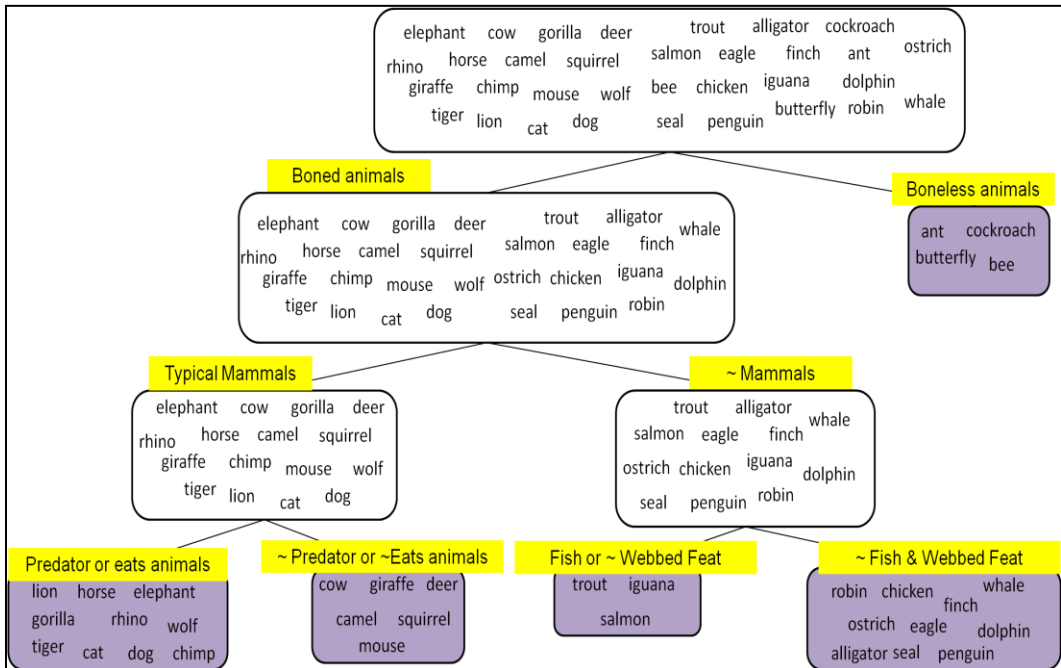


Figure 3.12: Hierarchical structure of dataset C.

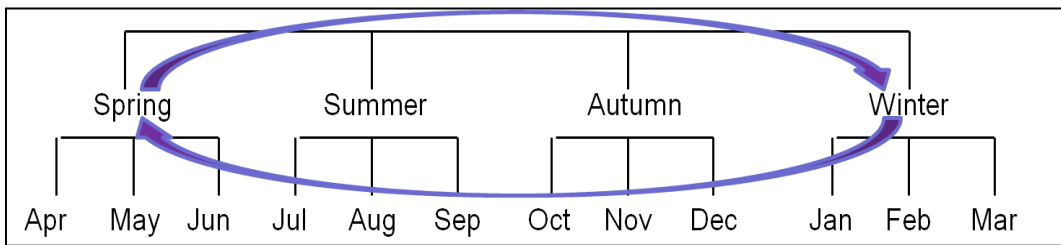


Figure 3.13: Temporal (cycle) and spatial (hierarchy) relationships of seasons and months.

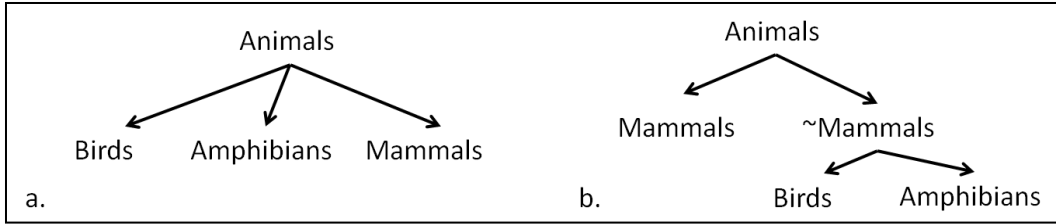


Figure 3.14: (a) Less abstraction in the categorization, (b) higher levels of abstraction in the categorization due to the use of the non-leaf concept ‘~ mammals’.

Chapter 4

Conclusion

4.1. Concluding Remarks

In summary, our model is an input integration framework for a Hubel Wiesel conceptual memory to bias the generalization process such that it contributes to the categorization of concepts in two ways. First, it decreases the probability of multiple incoherent categorizations while facilitating the achievement of natural coherent categorizations over coherent datasets. Second, it increases the probability of achieving the broadest distinction - the quality of early concept differentiation due to progressive differentiation phenomena - of the data. Assuming that changes in input integration framework of a hierarchical memory is one of the sources of the progressive differentiation of concepts, our local correlation model can be regarded as a basic input integration framework corresponding to early local correlations of input features leading to early concept differentiation.

Furthermore, we designed two different algorithms to implement our model. The potential performance of these two algorithms have been studied and compared

under different situations including, different hierarchical structures, different growth thresholds, and different types of input data. These two operations are different in the way they handle the strength of biasing the categorization towards general features. The quality of features being general or specific is subject to continuous change upon receiving new inputs – new features and new patterns. Therefore, it is questionable whether or not *max-weight* operation which gives a very high weight to the detected general features within a single entry might be a brain-like operation. Though, our simulations show that the *max-weight* operation produces more coherent results which are also consistent with the expected broad distinctions perceived in early childhood. May be in early stages of learning, an operation like *max-weight* is used to perceive broad distinctions and build the basic wirings in the brain. While, later a more moderate operation like *sum-weights* is used which does not bias the categorization as strongly as *max-weight* does.

In general, *max-weight* algorithm is always more significant than *sum-weights* algorithm in biasing the categorization. The table below summarizes the effect of the decrease of the growth threshold on the probability and quality of the broad distinctions emerged by the local correlation model.

Table 4.1: The effect of decreasing growth threshold on the categorization of the local correlation model.

| Algorithm | Data type | Probability of broad distinctions | Quality of broad distinctions |
|--------------------|---------------------|------------------------------------------|--------------------------------------|
| <i>Sum-weights</i> | Unique structured | Decrease | --- |
| <i>Max-weight</i> | Multiple structured | Unchanged | --- |
| <i>Sum-weights</i> | Unique structured | Randomly changed (decrease/increase) | --- |
| <i>Max-weight</i> | Multiple structured | --- | Grows finer and more coherent |

Our model can be also used to fit any given feature data into a hierarchical structure and provides a possible explanation on how human mind assigns a hierarchical structure to a given data. The Changes of the growth threshold over the input data at each node of the hierarchy is capable of changing the number of branches stemming from the corresponding node. Therefore, the output of iterative application of our model over input data should not be expected to be always a binary hierarchical structure.

4.2. Future Works

In the last section, the success of the model – specifically *max-weight* algorithm – to effectively bias the generalization towards the broadest coherent

differentiations of data is demonstrated. Assuming that changes in input integration framework of a hierarchical memory is one of the sources of the progressive differentiation of concepts, two possible directions for future works can be discussed.

1) Further study can be conducted to simulate the later developments (detection of finer distinctions) of the progressive concept differentiation - in the bottom-up pass - on the basis of prior broad distinctions and smooth changes in the input integration framework. The smooth changes to the input integration framework may include both operations of input management and prioritization. Based on the prior knowledge of broad categories, the model of local correlation should be evolved through further trainings.

2) Another direction to continue this work is to find a consistent model of incremental learning with the progressive differentiation phenomena. It can be discussed that changes to the input integration framework are caused by two main sources. The first source is the change of the quantity and quality of the prior categorization judgments. The second source is the incremental learning procedure. In the course of incremental learning, new patterns and features might be input to the system and they shall be used to provide the system with more coherent categorization judgments. In this endeavor, further study needs to be conducted on the mechanism by which necessary changes to input management and prioritization operations should be made upon the entrance of new features and patterns.

Bibliography

- [1] Margolis, E., & Laurence, S., *Concepts: Core Readings*, MIT Press, 1999.
- [2] Stanford encyclopedia of philosophy: Cognitive Science, <http://plato.stanford.edu/entries/concepts/>.
- [3] McClelland, J. L., & Rogers, T. T., The parallel distributed processing approach to semantic cognition, *Nature reviews Neuroscience*, 4, 310-322, 2003.
- [4] Rogers, T. T., & McClelland, J. L., *Semantic Cognition, A parallel distributed processing approach*. MIT press, 2004.
- [5] Rogers, T. T., & McClelland, J. L., Précis of Semantic Cognition, A parallel distributed processing approach, *Brain and Behavioral Sciences*, 31, 689-749, 2008.
- [6] Vygotsky, L. S., *Thought and Language*, MIT Press, 1986.
- [7] Inhelder, B., & Piaget, J., *The Early growth of logic in a child, Classification and serialization*, London: Routledge Kegan Paul, 1964.
- [8] Murphy, G. L., *The big book of concepts*, MIT Press, 2002.
- [9] Rumelhart, D. E., *An introduction to Neural and Electronic Networks*, New York: Academic Press, pp 405-420, 1990.
- [10] Jones, S. S., & Smith, L. B., How children know the relevant properties or generalizing object names, *Developmental Science*, 5, 219-232, 2002.

- [11] Gelman, S. A. & Markman, E., Categories and Induction in young children, *Cognition*, 23, 183-209, 1986.
- [12] Rumelhart, D. E. & Todd, P. M., *Learning and Connectionist Representations*. in Attention and Performance XIV, Synergies in Experimental Psychology, Artificial Intelligence, and Cognitive Neuroscience, MIT Press, pp 3-30, 1993.
- [13] Sloutsky, V. M., Lo, Y.-F., & Fisher, A., How much does a shared name make things similar, Linguistic labels, Similarity and the development of inductive inference, *Child Development*, 72, 1695-1709, 2001.
- [14] Sloutsky, V. M., The role of similarity in the development of categorization, *Trends in Cognitive Sciences*, 7, 246-251, 2003.
- [15] Sloutsky, V. M., Kloos, H., & Fisher, A. V., When looks are everything: appearance similarity versus kind information in early induction, *Psychological Science*, 18 (2), 179-185, 2007.
- [16] Quillian, M. R., *Semantic Information Processing*, MIT Press, pp 227-270, 1965.
- [17] Mountcastle, V., The columnar organization of the neocortex. *Brain*, 120, 701-722, 1997.
- [18] Mountcastle, V., *An Organizing Principle for Cerebral Function: The Unit Model and the Distributed System*, The Mindful Brain (Gerald M. Edelman and Vernon B. Mountcastle, eds.) Cambridge, MA: MIT Press, 1978.

- [19] Cadieu, C., Kouh, M., Pasupathy, A., Conner, C., Riesenhuber, M., & Poggio, T. A., A Model of V4 Shape Selectivity and Invariance, *Journal of Neurophysiology*, 98, 1733-1750, 2007.
- [20] Riesenhuber, M., & Poggio, T., Hierarchical models of object recognition in cortex, *Nature Neuroscience*, 2(11), 1019-1025, 1999.
- [21] Van Essen, D. C., Anderson, C. H., & Felleman, D. J., Information Processing in the primate visual system: An integrated systems perspective, *Science*, 255 (5043), 419-423, 1992.
- [22] Felleman, D. J., & Van Essen, D. C., Distributed hierarchical processing in the primate cerebral cortex, *Cerebral Cortex*, 1(1), 1-47, 1991.
- [23] Anderson, R. J., Snowden, R. J., Treue, S., & Grazia, M., Hierarchical Processing of motion in the visual cortex of a monkey, *Cold Spring Harbor Symposia on quantitative Biology*, 55, 741-748, 1990.
- [24] Wessinger, C. M., Vanmeter, J., Tian, B., Van Lare, J., Pekar, J., & Rauschecker, J. P., Hierarchical organization of the human auditory cortex revealed by functional magnetic resonance imaging, *Journal of Cognitive Neuroscience*, 13(1), 1-7, 2001.
- [25] Read, H.L., Winer, J. A., & Schreiner, C. E., Functional Architecture of Auditory Cortex, *Current Opinions in Neurobiology*, 12(4), 433-40, 2002.

- [26] Iwamura, Y., Dynamic and hierarchical processing in the monkey somatosensory cortex, *International symposium on the dynamics of perception*, 14, 107-111, 1993.
- [27] Toda, T., & Taoka, M., Hierarchical somesthetic processing of tongue inputs in the post central somatosensory cortex of conscious macaque monkeys, *Experimental brain research*, 147(2), 243-251, 2002.
- [28] Hubel, D., & Weisel, T., Receptive fields and functional architecture in two non striate visual areas (18 and 19) of a cat, *Journal of Neurophysiology*, 28, 229-289, 1965.
- [29] Lampl, I., Ferster, D., Poggio, T., & Riesenhuber, M., Intracellular Measurements of Spatial Integration and the MAX Operation in Complex Cells of the Cat Primary Visual Cortex, *Journal of Neurophysiology*, 92, 2704-2713, 2004.
- [30] Fukushima, K., Neocognitron for handwritten digit recognition 1, *Neurocomputing*, 51C, 161-180, 2003.
- [31] Fukushima, K., Neocognitron capable of incremental learning, *Neural networks*, 17, 37-46, 2004.
- [32] Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., & Poggio, T., Robust Object Recognition with Cortex-like Mechanisms, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3), 411-426, 2007.

- [33] LeCun, Y., Lowe, D. J., Malik, J., Mutch, J., Perona, P., & Poggio, T., Object Recognition, Computer Vision, and the Caltech 101: A Response to Pinto et al., *PLoS Computational Biology*, Posted Online March 2008.
- [34] Mel, B. W., SEEMORE: Combining Color, Shape and Texture Histogramming in a neurally inspired approach to visual object recognition, *Neural Computation*, 9, 777-804, 1997.
- [35] LeCun, Y., Huang, F. J., & Bottou, L., Learning methods for Generic object recognition with invariance to pose and lighting, *IEEE Conference on Computer Vision and pattern Recognition*, 2004.
- [36] Wallis, G., & Rolls, E., A model of invariant object recognition in the visual system, *Progress in neurobiology* 51, 167-194, 1997.
- [37] Rolls, E. T., & Stringer, S. M., Invariant Global Motion Recognition in the Dorsal Visual System: A Unifying Theory, *Neural Computation*, 19(1), 139-169, 2007.
- [38] Stringer, S. M., & Rolls, E. T., Invariant Object Recognition in the Visual System with Novel Views of 3D Objects, *Neural Computation*, 14(11), 2585-2596, 2001.
- [39] Wersing, H., & Koener, E., Learning Optimized features for hierarchical models of invariant recognition, *Neural Computation*, 15 (7), 1559-1588, 2003.

- [40] Wolf, L., Bileschi, S., & Meyers, E., Perception strategies in hierarchical vision systems, *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.
- [41] Ramanathan, K., & Shi, S., & Chong, T. C., A Hubel Weisel model for hierarchical representation of concepts in textual documents, *The Annual Meeting of the Cognitive Society (COGSCI)*, 1106-1111, 2010.
- [42] Murphy, G. L., & Medin, D. L., The role of theories in conceptual coherence. *Psychological review*, 92, 289- 316, 1985.
- [43] Alahakhoon, D., Halgamuge, S. K., & Srinivasan, B., Dynamic Self Organizing maps with controlled growth for Knowledge discovery, *IEEE Transactions on neural networks*, 11(3), 601-614, 2000.
- [44] Kemp, C., & Tenenbaum, J. B., The discovery of structural form, *Proceedings of the National Academy of Science*, 105(31), 10687-10692, 2008.
- [45] Rosch, E., Principles of categorization. *Cognition and Categorization*, eds Rosch E, Lloyd BB (Lawrence Erlbaum, New York), pp 27-48, 1978.

Appendix A-1: Dataset A - List of Entities

Following are the complete lists of patterns, features in Dataset A from Rogers and McClelland Corpus (2004). It includes 21 patterns and 26 features.

List of Patterns

Pine, Oak, Maple, Birch, Rose, Daisy, Tulip, Sunflower, Robin, Canary, Sparrow, Penguin, Sunfish, Salmon, Flounder, Cod, Cat, Dog, Mouse, Goat and Pig.

List of Features

Pretty, Big, Living, Green, Red Yellow, White, Twirly, Grow, Move, Swim, Fly, Walk, Sing, Leaves, Roots, Skin, Legs, Bark, Branches, Petals, Wings, Feathers, Scales, Gills and Fur.

Dataset A – Input Matrix, columns 1-12:

| | pine | oak | maple | birch | rose | daisy | tulip | sunflower | robin | canary | sparrow | penguin |
|----------|------|-----|-------|-------|------|-------|-------|-----------|-------|--------|---------|---------|
| Living | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| pretty | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| green | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| big | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| red | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| yellow | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| white | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| twirly | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| grow | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| move | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| swim | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| fly | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| walk | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| sing | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| bark | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| branches | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| petals | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| wings | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| feathers | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| scales | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gills | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| leaves | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| roots | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| skin | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| legs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| fur | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Dataset A – Input Matrix, columns 12-21:

| | sunfish | salmon | flounder | cod | dog | cat | mouse | goat | pig |
|----------|---------|--------|----------|-----|-----|-----|-------|------|-----|
| Living | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| pretty | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| green | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| big | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| red | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| yellow | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| white | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| twirly | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| grow | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| move | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| swim | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| fly | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| walk | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| sing | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| bark | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| branches | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| petals | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| wings | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| feathers | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| scales | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| gills | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| leaves | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| roots | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| skin | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| legs | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| fur | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

Appendix A-2: Dataset B - List of Entities

Following are the complete lists of patterns, features in Dataset B from Rogers and McClelland Corpus (2004). It includes 13 patterns and 14 features.

List of Patterns

Robin, Canary, Sparrow, Penguin, Sunfish, Salmon, Flounder, Cod, Cat, Dog, Mouse, Goat and Pig.

List of Features

Pretty, Big, Red, Yellow, White, Swim, Fly, Walk, Sing, Wings, Feathers, Scales, Gills and Fur.

Dataset B – Input Matrix

| | robin | canary | sparrow | penguin | sunfish | salmon | flounder | cod | dog | cat | mouse | goat | pig |
|----------|-------|--------|---------|---------|---------|--------|----------|-----|-----|-----|-------|------|-----|
| pretty | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| big | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| red | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| yellow | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| white | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| swim | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| fly | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| walk | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| sing | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| wings | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| feathers | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| scales | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| gills | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| fur | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

Appendix A-3: Dataset C - List of Entities

Following are the complete lists of patterns, features in Dataset C from Kemp and Tenenbaum Corpus (2008). It includes 33 patterns and 102 features.

List of Patterns

Elephant, Rhino, Horse, Cow, Camel, Giraffe, Chimp, Gorilla, Mouse, Squirrel, Tiger, Lion, Cat, Dog, Wolf, Seal, Dolphin, Robin, Eagle, Chicken, Salmon, Trout, Bee, Iguana, Alligator, Butterfly, Ant, Finch, Penguin, Cockroach, Whale, Ostrich, and Dear.

List of Features

Lungs, Large brain, Spinal cord, Warm blooded, Teeth, Feet, 2 legs, 6 legs, Tongue, Visible ears, Nose, Paws, Lives in groups, Tough skin, Long neck, Fins, Long legs, Fish, Snout, Antennae, Eats rodents, Travels in groups, Long, Large, Roars, Claws, Wings, Green, Tusks, Carnivore, Slender, Dangerous, Eats grass, Tall, Beak, Slow, Fast, Lives in trees, eats leaves, Smooth, Lizard, Eats seeds, Poisonous, Soft, Bird, Black, Hunts, Howls, Gills, Feline, Stripes, Lives in the forest, 4 legs, Strong, Predator, Rodent, Lives in hot climates, Webbed feet, Eats mice, Lives in lakes, Squawks, Ferocious, Lives in cold climates, Yellow, Lives in the ocean, Hooves, Feathers, Makes loud noises, Eats bugs, Runs, Bites, Crawls, Swims, Flies, Insect, Lives in water, Sings, Horns, Eats nuts, Brown, Eats fish, Lays eggs, Scaly, Eats animals, Furry, Smart, Blue, Tail, Flippers, Reptile, Lives on land, Colorful, Lives in houses, Digs holes, Lives in grass, Mammal, White, Canine, Womb, Subcutaneous fat, red blood, and Bones.

Dataset C – Input Matrix, columns 1-12

| | elephant | rhino | horse | cow | camel | giraffe | chimp | gorilla | mouse | squirrel | tiger | lion |
|-----------------------|----------|-------|-------|-----|-------|---------|-------|---------|-------|----------|-------|------|
| lungs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| large brain | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| spinal cord | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| warm blooded | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| teeth | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| feet | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 legs | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 6 legs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| tongue | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| visible ears | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| nose | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| paws | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| lives in groups | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| tough skin | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| long neck | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| pins | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| long legs | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| fish | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| snout | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| antennae | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats rodents | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| travels in groups | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| long | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| large | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| roars | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| claws | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| wings | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| green | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| tusks | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| carnivore | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| slender | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| dangerous | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| eats grass | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| tall | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| beak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| slow | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| fast | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| lives in trees | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| eats leaves | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| smooth | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lizard | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats seeds | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| poisonous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| soft | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| bird | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| black | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| hunts | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| howls | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gills | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| feline | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| stripes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| lives in the forest | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 4 legs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| strong | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| predator | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| rodent | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| lives in hot climates | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| webbed feet | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats mice | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives in lakes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| squawks | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ferocious | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| lives in old climates | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| yellow | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| lives in the ocean | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hooves | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| feathers | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| makes loud noises | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| eats bugs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| runs | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| bites | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| crawls | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| swims | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| flies | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| insect | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives in water | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| sings | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| horns | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats nuts | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| brown | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| eats fish | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lays eggs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| scaly | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats animals | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| furry | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| smart | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| blue | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| tail | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| flippers | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| reptile | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives on land | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| colorful | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| lives in houses | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| diges holes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| lives in grass | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| mammal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| white | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| canine | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| womb | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| subcutaneous fat | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| red blood | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| bones | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Dataset C – Input Matrix, columns 12-24

| | cat | dog | wolf | seal | dolphin | robin | eagle | chicken | salmon | trout | bee | iguana |
|-----------------------|-----|-----|------|------|---------|-------|-------|---------|--------|-------|-----|--------|
| lungs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| large brain | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| spinal cord | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| warm blooded | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| teeth | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| feet | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 2 legs | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 6 legs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| tongue | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| visible ears | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| nose | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| paws | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives in groups | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| tough skin | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| long neck | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| fins | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| long legs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| fish | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| snout | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| antennae | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| eats rodents | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| travels in groups | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| long | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| large | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| roars | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| claws | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| wings | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| green | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| tusks | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| carnivore | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| slender | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| dangerous | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| eats grass | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| tall | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| beak | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| slow | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| fast | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| lives in trees | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| eats leaves | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| smooth | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lizard | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| eats seeds | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| poisonous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| soft | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| bird | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| black | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| hunts | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| howls | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gills | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| feline | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| stripes | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| lives in the forest | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 4 legs | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| strong | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| predator | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| rodent | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives in hot climates | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| webbed feet | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| eats mice | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| lives in lakes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| squawks | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| ferocious | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| lives in old climates | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| yellow | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| lives in the ocean | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| hooves | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| feathers | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| makes loud noises | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| eats bugs | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| runs | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| bites | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| crawls | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| swims | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| flies | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| insect | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| lives in water | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| sings | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| horns | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats nuts | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| brown | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| eats fish | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| lays eggs | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| scaly | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| eats animals | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| furry | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| smart | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| blue | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| tail | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| flippers | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| reptile | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| lives on land | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| colorful | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| lives in houses | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| diges holes | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives in grass | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| mammal | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| white | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| canine | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| womb | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| subcutaneous fat | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| red blood | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| bones | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

Dataset C – Input Matrix, columns 24-33

| | alligator | butterfly | ant | finch | penguin | cockroach | whale | ostrich | dear |
|-----------------------|-----------|-----------|-----|-------|---------|-----------|-------|---------|------|
| lungs | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| large brain | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| spinal cord | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| warm blooded | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| teeth | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| feet | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 2 legs | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 6 legs | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| tongue | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| visible ears | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| nose | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| paws | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| lives in groups | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| tough skin | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| long neck | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| fins | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| long legs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| fish | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| snout | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| antennae | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| eats rodents | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| travels in groups | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| long | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| large | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| roars | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| claws | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| wings | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| green | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| tusks | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| carnivore | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| slender | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| dangerous | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats grass | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| tall | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| beak | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| slow | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| fast | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| lives in trees | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| eats leaves | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| smooth | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| lizard | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats seeds | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| poisonous | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| soft | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| bird | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| black | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| hunts | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| howls | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gills | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| feline | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| stripes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives in the forest | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 4 legs | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| strong | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| predator | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| rodent | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives in hot climates | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| webbed feet | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats mice | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives in lakes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| squawks | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| ferocious | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| lives in old climates | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| yellow | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| lives in the ocean | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| hooves | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| feathers | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| makes loud noises | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| eats bugs | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| runs | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| bites | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| crawls | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| swims | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| flies | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| insect | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| lives in water | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| sings | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| horns | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats nuts | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| brown | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| eats fish | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| lays eggs | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| scaly | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eats animals | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| furry | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| smart | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| blue | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| tail | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| flippers | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| reptile | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lives on land | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| colorful | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| lives in houses | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| diges holes | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| lives in grass | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| mammal | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| white | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| canine | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| womb | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| subcutaneous fat | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| red blood | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| bones | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |