Holographic Reduced Representations for Dimensional Attention Learning

By

Huizhi Wang

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science

in

Computer Science

Middle Tennessee State University

December 2019

Thesis Committee:

Dr. Joshua L. Phillips, Chair

Dr. Suk Jai Seo

Dr. Yi Gu

ACKNOWLEDGEMENTS

I would like to thank Dr. Joshua L. Phillips for his expert advice and encouragement throughout this project. I also wish to thank anonymous reviewers from the MTSU writing center for their suggestions and Dr. Suk Jai Seo and Dr. Yi Gu for revising this thesis.

ABSTRACT

One aspect of machine learning is that it has sought to mimic how the brain learns. In 1992, Kruschke published a computer model called ALCOVE which sought to model category learning. In 2004, Phillips and Noelle made this model more realistic and closer to human brain activity by incorporating Temporal Difference (TD) Learning. However, this model is currently incompatible with readily available encoding techniques like Holographic Reduced Representations (HRRs) used in related cognitive architectures. In this study, the standard ALCOVE mapping function is replaced with the convolution method employed by the HRRs. The original function learned in classic 6-type category learning tasks consists of only three dimensions, but other tasks may require more dimensions. This study empirically demonstrates that convolution could compress more features into the learning procedure. It also empirically demonstrates this method to be valid for dimensional attention learning by replicating ALCOVE performance on category learning tasks with binary, separable and integral feature tasks.

Keywords: Machine learning, ALCOVE, Category learning, Holographic Reduced Representations

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I.

Introduction

Dimensional attention learning is a property of human learning where people concentrate on only a few, task-relevant features when there are multiple features available. Nosofsky did an experiment in 1984 [6], showing that when people are given a circle and a square, they immediately understand that their shapes are different. But if they are given two circles with similar radii, it will take some time for them to distinguish the difference between the two patterns. After lots of training, people can determine the important features of a given pattern because they learn the dimensions that change among given patterns. This is an example of dimensional learning.

In 1992, Kruschke implemented a technique in computer software called ALCOVE [4] that was based on Nosofsky's research [6]. It only focused on three dimensions because the research at that time only experimented with three dimensions (shape, color and size). Though Kruschke's experiment matched the data from Nosofsky's paper, this method is limited and difficult to flexibly adapt to other learning experiments which include more dimensions without modifying the underlying software model.

This study is based on the ALCOVE model and incorporates a mapping mechanism coming from Holographic Reduced Representations (HRRs) [8]. It proposes using convolution to compress several vectors into one fixed vector for presentation of model inputs. This feature overcomes the defect of limited input dimensions, allowing for more dimensions in learning procedures, and may be a more realistic and generalizable method which can be applied in other experiments in the future.

CHAPTER II.

Background

In 1984, Nosofsky, a psychology researcher, addressed the thesis that humans learn things categorically with attention to different features and explained how some categories were learned faster than others [6]. This thesis later was developed by Kruschke as a computer model, named ALCOVE [4]. It provided a model of human-like category learning in machine learning, allowing researchers to build models that can be executed on the computer to simulate the process of category learning.

The ALCOVE model used a neural network with backpropagation to achieve the simulation. The neural network was built with attention weights for the features, a hidden layer and an association matrix. They worked together on the inputs to produce an output vector. Then the backpropagation procedure compared the difference between output vectors and targets and updated the attention weights on the feature dimensions and association matrix [5]. The model was trained using 6-type category learning problems. When the simulation data from the ALCOVE model was compared to data from Nosofsky's experiment [6], ALCOVE gave a good fit to the data on learning categories. In 2004, Phillips and Noelle replaced the backpropagation function used in ALCOVE with temporal difference(TD) learning to make it similar to learning in the human brain, which means the whole simulation is closer to the behavior of humans because backpropagation is "computationally powerful but biologically implausible" [7]. However, this method is also not realistic because it still only fits to the exact data dimensions used in ALCOVE and is not applicable for other data with arbitrary dimensions. This led to a search for another method to improve the structure of the ALCOVE model, to make it more general, to allow for flexible inputs with more dimensions and still maintain high learning accuracy.

Holographic Reduced Representations(HRRs) were proposed by Plate in 1995 [8]; they have properties which should be useful in ALCOVE research. The reason that ALCOVE

alone is currently not sufficient for other research is because the dimensions of the objects are static. That means the features of objects are simple and there can only be a fixed numbers of features. But it is impossible to use this model to simulate human daily life. To describe any objects in human daily life would require more than a few fixed features. To overcome this barrier, it was necessary to increase the dimensions of inputs to the model in a flexible, scalable manner.

Vectors are usually distributed representations of discrete items. Conventional convolution (also known as aperiodic convolution) can compress several vectors into one, but the results still suffer from the problem of expanding dimensionality. For example, two n-element vectors will produce a 2n-1 length vector with aperiodic convolution. Circular convolution of distributed representations in HRRs can solve expanding dimensionality problem from this perspective and also compress more dimensions. With this method, ”arbitrary variable bindings, short sequences of various lengths, simple frame-like structures, and reduced representations can be represented in a fixed length vector” [8]. In other words, circular convolution can compress arbitrary number of vectors with any length N into a single vector of length N. The model described in this paper is adapted from a paper by Borsellino and Poggio [1] which also addressed distributed representations and improved them with the convolution method. The advanced feature is that this method could compress several vectors recursively into one fixed length. Moreover, the compressed vector with convolution can be decoded into components by the correlation operation. It is easy for researchers to obtain the component vectors from a convolved vector and use them in later experiments. ALCOVE would be more realistic and flexible if convolution and correlation could be used in the model.

Materials on more recent HRR-based models lack dimensional attention [2] [3] [9], these models use HRRs but lack a mechanism for dimensional attention. Incorporation of HRRs into ALCOVE would provide a general method of dimensional attention that is

compatible with HRRs and potentially of use for speeding up learning in these models.

CHAPTER III.

Methods

In this chapter we will introduce the methods of inserting Holographic Reduced Representations(HRRs) into ALCOVE model and create the new model HALCOVE for the experiments. Futhermore, because ALCOVE model can be divided into two models, one is for the discrete data and the other is for the continuous data. Because two kinds of data sets have different structures, we will use different approach to build HALCOVE models according to the difference between discrete data and continuous data.

ALCOVE Implementation

The original ALCOVE model and procedure described in Kruschke's paper [4] is implemented for the experiments. We run the model with different data sets and obtain the learning performance as a benchmark for the HALCOVE model.

In the discrete data model, there are six different category learning models and the model learns each problem cross fifty epochs(pass through the data set). Problems are designed from easy to difficult; they all have three dimensions, the easiest problem (first level problem) only requires one dimension, the more difficult one (second level problem) requires two dimensions, and the others use all three dimensions with increasing complexity.

In the continuous data model, there are two kinds of data in this model. One is called the integral data set: this data has six problems of category learning. The other is called the separable data set: it only has four problems of category learning. The difference between discrete and continuous data sets is that the features in the discrete data set are binary while in the continuous data sets they are not. For example, when we are asked to distinguish the shape during the learning, shapes are explict, either sqaure or circle. It is a binary option. But other attributes like the length of a stick are implict, it may be 1 inch or 2 inches or some magnitude between 1 inch and 2 inches; the value is not binary in this situation.

The ALCOVE model is used in order to determine the expected performance and learning accuracy for the different category problems. The goal with the HALCOVE is to achieve similar performance to ALCOVE when confronting the same problem.

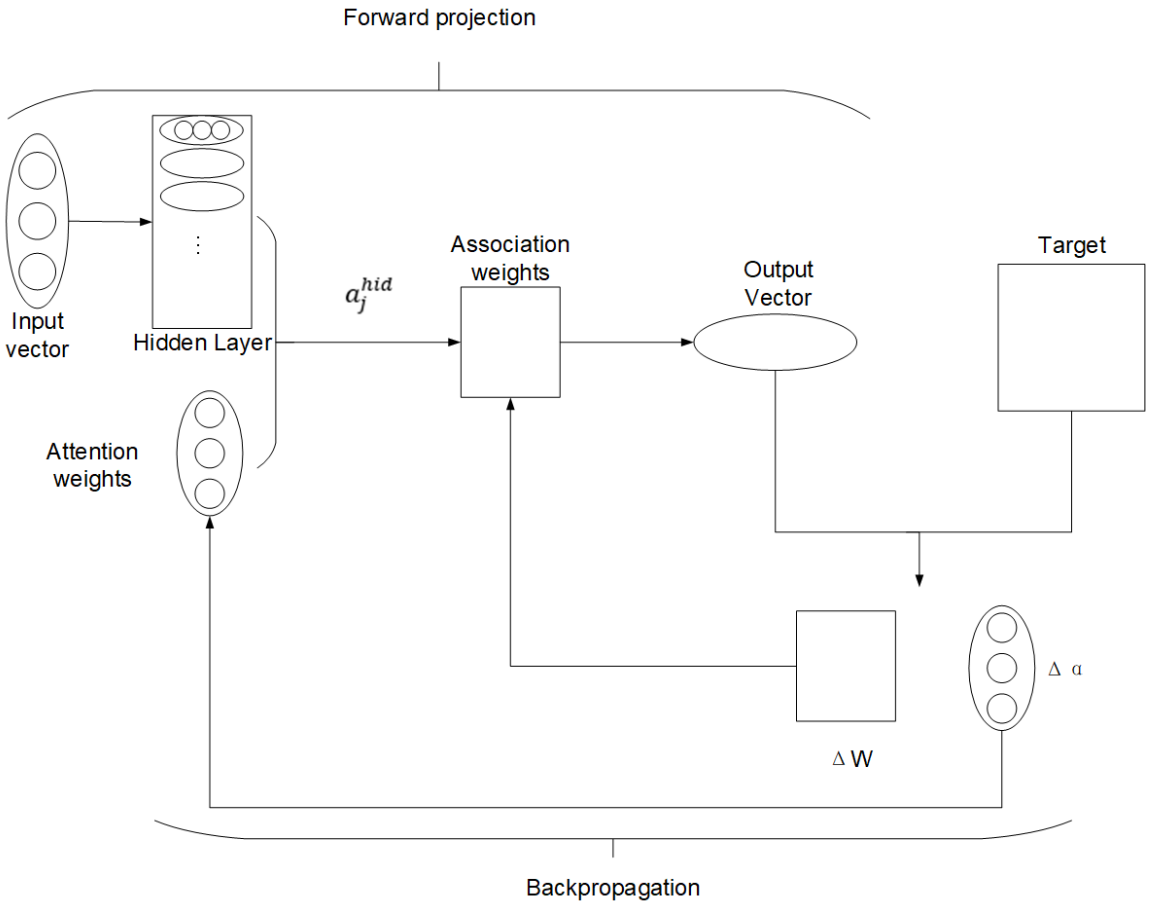


Figure 1: ALCOVE Model Structure

Figure 1 illustrates the ALCOVE model which consists of input vector, attention weights, a hidden layer and an association matrix. The standard procedure has two stages, forward projection and backpropagation [4]. Attention weights are used to determine the weights of dimensions. Higher attention weights cause the model to focus more on the dimensions with which they are associated.

The whole procedure can be represented with the following step:

1. Forward Learning
   (a) Compute the activation level of hidden layer
   (b) Compute the activation level of output layer
2. Backpropagation
   (a) Compute the delta attention weights
   (b) Compute the delta association weight matrix

1(a) Compute the activation level of hidden layer

The first value produced by ALCOVE is called the activation level of each hidden unit in the hidden layer. It is determined by using the following equation [7]:

$$a_j^{hid} = \exp[(-c) * (\sum_i \alpha_i |h_{ji} - a_i^{in}|^r)^{q/r}]$$

where a_j^{hid} is the activation of hidden units j, α_i is the attention weight for dimension i, c is the specificity of the hidden units, h_{ji} is the hidden unit j along the dimension i, a_i^{in} is the input value on the dimension i, r is the psychological distance metric and q is the similarity gradient.

We get the input vector from hidden units. In each epoch in learning, we will pick a hidden unit as our input and iterate this unitl we pick all of the hidden units.

1(b) Compute the activation level of output layer

The activation level of output a_k^{out} can be obtained by using matrix multiplication with

the activation level of the hidden layer and the association matrix.

$$a_k^{out} = \sum_j w_{kj} a_j^{hid}$$

2(a) Compute the delta attention weights

In backpropagation, the difference between the output and the target matrix is computed, and two error signals which are used to update the attention weights and association matrix can be retrieved by using the following equations [7]:

$$\Delta\alpha_i = (-\lambda_\alpha) * \sum_j [\sum_k (t_k - a_k^{out}) w_{kj}] a_j^{hid} c |h_{ji} - a_i^{in}|$$

$\Delta\alpha_i$ is the error signal for the attention weight on dimension i, the learning rate parameter for attention weights is λ_α. The target value for the output unit k is t_k. The equation uses the activation level of hidden layer a_j^{hid} that we computed in the forward learning procedure and the specificity c in the computation. It also compute the distance between the hidden layer units and input vector.

2(b) Compute the activation level of output layer

We use the follwing equation for computing the output value in forward learning:

$$\Delta w_{kj} = \lambda_w (t_k - a_k^{out}) a_j^{hid}$$

where Δw_{kj} stands for the weightupdate for the association matrix weight from hidden unit j to output unit k. The learning rate parameter for the association weights is λ_w. The target value for the output unit k is t_k.

The Δw is for the association matrix and $\Delta\alpha$ is for the attention weights. They are used

to update the attention weights and association matrix by a simple addition operation after each input pattern is processed by the network.

Approch for HALCOVE model

Generate and unitize the vectors

HRR use special vectors for the calculation. First, all the elements are generated by the normal distribution with a mean of 0 and a viariance of 1/N, N is the amount of elements in the vector.

Second, the vector needs unitization. The unitization in HRR is different from normal vector normalization. We apply Fourier transfrom and inverse Fourier transform on the vector and unitize it with the following fomulas [8]:

$$f_j(\vec{x}) = \sum_{k=0}^{n-1} x_k e^{-i2\pi jk/n}$$

$$f_j^{-1}(\vec{x}) = \frac{1}{n} \sum_{k=0}^{n-1} x_k e^{i2\pi jk/n}$$

When we transform the vector into frequency domain, the unitization is applied on the vector, and then we apply inverse Fourier transform to transform the result back to the time domain for the future operation. Assume it is applied on $\vec{x}$. It follows the following fomula:

$$x_{unit} = f_j^{-1}(f_j(x)/|f_j(x)|)$$

Convolution

HRRs introduced a method of compressing vectors by circular convolution.Assuming $\otimes$

stands for the convolution operation between two vectors and there are two vectors, $\vec{x}$ and $\vec{c}$, with the same length, N, and $\vec{t}$ is the product of the convolution, the convolution will follow the equation [1]:

$$\vec{t} = \vec{c} \otimes \vec{x}$$

$$\vec{t}_j = \sum_{k=0}^{N} \vec{c}_k * \vec{x}_{(j+N-k)\%N}$$

An advantage of circular convolution is that it maintains the same vector length (N) after convolution. What's more, if the two vectors before the convolution are unit vectors, the convolved vector will also be an unit vector. This means we can apply convolution recursively with potentially many unit vectors. Therefore, it is possible to overcome the limitation of a fixed number of dimensions imposed by the ALCOVE model. More vectors can be compressed into one vector in the model and its flexibility thereby could be enhanced.

Correlation

Another useful method concomitant with convolution is called correlation [8]. Plate demonstrated that convolution could be decoded by correlation when a series of vectors satisfy the following requirements: all elements in the vectors are generated from a normal distribution with a mean of zero and variance of 1/N (N stands for the length of the vector); the length of each vector is 1, which means that all the vectors are subsequently unitized; any two vectors constructed in this manner are approximately orthogonal.Assuming $\oplus$ stands for the correlation operation between two vectors. When vectors meet these qualifications, the component vector $\vec{y}$ can be retrieved from the convolved product $\vec{t}$ and the other component vector $\vec{c}$ by using the following equation [8]:

$$\vec{y} = \vec{t} \oplus \vec{c}$$

$$\vec{y}_j = \sum_{i=0}^{N} \vec{t}_i * \vec{c}_{(i-j+N)\%N}$$

This method forms the foundation of this study, because when composing all input vectors and hidden layer vectors by convolving them with vectors from each dimension, we subsquently need to break them into components when computing absolute differences $|h_{ji} - a_i^{in}|$ as required by the aforementioned ALCOVE equations. Correlation thereby offers a feasible function for HALCOVE to decode the convolved vectors and use the absolute differences to update its dimensional attention weights.

Convolutional power

In continuous model, because the data values are not binary, it is difficult to use simple convolution and correlation to build the hidden layer.Therefore, We introduce convolutional powers into the model. The convolutional power is a method that can exponentially raise a single vector to different different values so that we can represen continuous values in HRR vectors and intergrate them into HALCOVE. This method still requires vectors generated from HRRs as aforementioned and then one simply uses the power function to raise the power for a vector. We use the following fomula [8]:

$$f_j(\vec{x}^p) = (f_j(\vec{x}))^p$$

p is the power value in the fomula and is also the continuous value from the continuous model. We generate one feature vector for each dimension in the continuous model and raise it to an appropriate scaled power according to values accpeted in the ALCOVE model in order to produce the correspoding vectors used as patterns in the HALCOVE model.

Continuous convolution and correlation

In the continuous model, because the values along each dimension are continuous and they are specific for each hidden layer units, we introduce a different method of convolution and correlation to operate the hidden units in HALCOVE.

We use the following formula to build the hidden layer for the continuous model:

$$\vec{hid}_j = \sum_{i=0}^{N} \vec{b}_i \otimes (f_j(\vec{x}_j))^{p_j}$$

N is the amount of dimensions in the continuous model and j stands for the hidden layer unit. $(f_j(\vec{x}_j))^{p_j}$ is the convolutional power we aforementioned in the above section. For each unit in the hidden layer, we need to compress N dimensions together and produce one vector for the usage in HALCOVE.

First, we generate two vectors for each dimension, they are base vector $\vec{b}$ and feature vector $\vec{x}$. Base and feature vectors are randomly generated, unitized HRRs as described eariler. Second, we need to apply the method of convolutional power to combine the continuous feature values from the ALCOVE data so that we get new feature vectors(feature vector raised to a fractional exponential power) along each dimension for each hidden layer unit. Finally the convolution operation is applied. We convolve the base vector and power raised feature vector for each dimension and sum them up to one vector and that represents one hidden layer unit for HALCOVE model. Similar to how each feature vector mapped to each hidden layer unit in the ALCOVE model.

Hidden layer generated by this method will have an ability of decoding. Feature vectors can be derived from the hidden layer unit by correlating the input vector with the corresponding base vector of that dimension. It can be represented with following

calculation:

$$(f_j(\vec{x}_j))^{p_j} \approx \vec{hid}_j \oplus \vec{b}_i$$

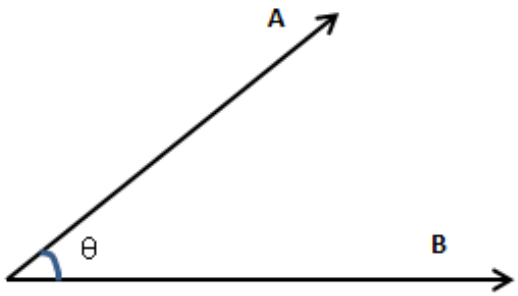
From the above fomula, we can retrieve the power raised feature vector along dimension i by correlating hidden layer unit with the base vector of dimension i and it can be used in the further calculation in HALCOVE model.

The only flaw of this operation is the introduction of noise when we apply the correlation. Because we sum several vectors from different dimensions together, they will interfere each other and produce noise. But we demonstrate in our experiments later on how it can be resolved by using long length of vectors in the calculation. For instance, the noise will be reduced when we use a vector of 1024 elements instead of a vector of 16 elements as the feature vector in the operation. This caveat is common to all summation methods for distributed representations.

Cosine distance

When convolution and correlation are introduced into ALCOVE, the method for calculating the activation level of the hidden layer needs to be adapted. Instead of using a single scalar to represent feature dimensions of the objects, HALCOVE now uses vectors to represent the dimensions. In ALCOVE, the model calculates the distance between the elements of two vectors and sums them after multiplying them by the attention weights. When using vectors, an alternative method is used to calculate the distance between inputs and the hidden layer.

For HALCOVE, the cosine distance is also applied in learning (Figure 2). Two vectors, $\vec{a}$ and $\vec{b}$, constitute an angle θ, and it is calculated by the formula adapted from Figure 2:



$$\boldsymbol{A} \cdot \boldsymbol{B} = |\boldsymbol{A}||\boldsymbol{B}|cos\theta$$

Figure 2: Dot Product of Two Vectors

$$\cos\theta = \frac{\vec{a} * \vec{b}}{|\vec{a}| * |\vec{b}|}$$

The cosine value of angle θ is used as a new criterion to measure the distance between two vectors; if two vectors are the same, $\cos\theta$ will be 1 because they overlap from the geometric perspective. But in HALCOVE, the distance between two vectors should be 0 when they are same because it means the features are matched similarly to the absolute distance calculated by ALCOVE. To correct this, $\cos\theta$ must be subtracted from 1. This results in the standard cosine distance, computed using the following formula:

$$d_{cos}(\vec{a},\vec{b}) = 1 - \cos\theta = 1 - \frac{\vec{a} * \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Structure and implementation

In HALCOVE, the structures are diffferent due to two different data sets-discrete data sets and continuous data sets. In this section we are going to introduce the structures of HALCOVE for discrete data sets and continuous data sets.

Structure for discrete data sets

As we aforementioned, the features in this data sets are binary. So when we build our hidden layer with vectors, we can pick one feature vector from each dimension and apply convolution on them to build the hidden unit layer. The model works as following steps:

1. Build hidden layer for HALCOVE
    (a) Generate two feature vectors for each dimension
    (b) Apply convolution to build hidden layer
2. Forward learning
    (a) Compute the activation level of hidden layer
    (b) Compute the activation level of output layer
3. Backpropagation
    (a) Compute delta attention weights
    (b) Compute delta association weight matrix

We will explain the methods if the formula or step is different from ALCOVE model. The steps that are not listed below mean that they are the same as ALCOVE model.

1(a) Generate two feature vectors for each dimension

As mentioned above, each dimension has binary options. It means we also need to generate two feature vectors in HALCOVE correspoding to that. We use the normal distribution with a mean of 0 and a viarance of 1/N (N is the amount of elements in the vector/the length of the vector) to generate the elements for the vector. To satisfy the prerequests for the convolution and correlation approaches, we also need to unitize the vector for the following experiments.

1(b) Apply convolution to build hidden layer

After we get all feature vectors for dimensions, we will apply the convolution which is introduced in approch section to compress them and make a hidden unit for the hidden layer. We will pick one feature vector from each dimension for each hidden unit. For example, assuming there are three dimensions and each dimension has two feature vectors, to build a hidden unit, we may pick the first vector of dimension 1, the second vector of dimension 2 and the second vector from dimension 3 as components and compress them by convolution.

2(a) Compute the activation level of hidden layer

Cosine distance is used to replace the absolute scalar distance in the ALCOVE model $|h_{ji} - a_i^{in}|$. In HALCOVE, since the hidden layer is composed of several vectors, correlation is used to break them up into the component vectors, $\vec{hid}_{ji}^{comp}$. The input vectors are broken up into component vectors, $\vec{input}_i^{comp}$ before calculating the cosine distance by correlation, HALCOVE model will compute the forward procedure by using the following equation:

$$a_j^{hid} = \exp[(-c) * (\sum_i \alpha_i * d_{cos}(\vec{hid}_{ji}^{comp}, \vec{input}_i^{comp})^r)^{r/q}]$$

3(a) Compute delta attention weights

The absolute scalar distance $|h_{ji} - a_i^{in}|$ is also used in computing the $\Delta\alpha$ procedure in the backpropagation. We also replace it with cosine distance and get the following fomula in backpropagation for HALCOVE model:

$$\Delta\alpha_i = (-\lambda_\alpha) * \sum_j [\sum_k (t_k - a_k^{out}) w_{kj}] a_j^{hid} c |d_{cos}(\vec{hid}_{ji}^{comp}, \vec{input}_i^{comp})|$$

After adapting all relative parts in ALCOVE, the HALCOVE model can feasibly compress

an arbitrary number of dimensions. In other words, for problems with an arbitrary number of dimensions, no changes to the archtecture are required across or between different learning tasks. Thus, HALCOVE represents a flexible learning method for dimensional attention using HRRs. With this model we can start our experiments and compare the data with that coming from ALCOVE.

Structure for continuous data sets

1. Build hidden layer for HALCOVE
    (a) Generate base vector and feature vector for each dimension
    (b) Apply convolutional power
    (c) Apply continuous convolution
2. Forward learning
    (a) Compute the activation level of hidden layer
    (b) Compute the activation level of output layer
3. Backpropagation
    (a) Compute delta attention weights
    (b) Compute delta association weight matrix

1(a) Generate base vector and feature vector for each dimension

In continuous data sets, we use the approach continuous convolution. It uses a base vector and a feature vector for building the hidden layer, since the data in continuous data sets is continuous, we will raise the feature vector to different powers according to the value

we have in ALCOVE. The way we generate base vector and feature vector is still the same as what we do in the HALCOVE for discrete data sets.

1(b) Apply convolutional power

This step is the key step for building the hidden layer in continuous HALCOVE model, we apply convolutional power on the feature vector according to the value in the hidden layer pattern in ALCOVE. For each dimension, it will have a series of float numbers. Firstly, we find the maximum and minimum value among them. Then we scale them from their range to the range between 1 and 2 and these scaled values will be used as the power value for the convolutional power. At last we can apply convolutional power approach introduced in the Approach chapter on feature vectors.

1(c) Apply continuous convolution

When we get power raised feature vectors, we can build our hidden layer with them and base vectors by apply continuous convolution. In each dimension, we multiply the base vector by the power raised feature vector. Then we sum all the products up from all dimensions to get the hidden layer unit.

Experiments

In the experiments, we first use a normal distribution with a mean of zero and a variance of 1/N (N is the length of vector) to generate three pairs of vectors. Then unitization is applied on these vectors to produce an unit vector. We can easily use the ideas from Plate to write on equation for this. The vectors generated follow the constraints can apply convolution and correlation. This means the convolved vectors can be decoded in the HALCOVE model since all vectors meet the requirements of convolution.

The input pattern and hidden layer (both convolved vectors) are sent to the HALCOVE

model. There is a difference between discrete data and continuous data. In discrete data, HALCOVE will pick one feature vector from each dimension and convolve them and send the convolved product as one hidden unit in hidden layer. For example, if discrete data has three dimensions and all dimensions have two features, there will be 8 (2^3) units in the hidden layer. So we use simple convolution and correlation in this model. However, continuous data has many different feature values in the patterns. This means shall create the same number of feature vectors in the HALCOVE model. We apply the convolutional power method when confronting continuous data.

When the hidden layer is built. HALCOVE begins the learning procedure just like ALCOVE with two key differences as mentioned in the former sections in this chapter: the cosine distance for discrete inputs and correlation for contiuous inputs. To determine whether or not the learning curves for ALCOVE and the new model are in agreement after experiments, we note the learning accuracy percent correct classification of each problem every five epochs and plot them. All the codes of the experiments can be found on Github: `https://github.com/huizhiw/alcoveWithHRR/tree/master/HALCOVE`.

Comparing models

We use mean squared error (MSE) to measure the distance between the data accuracy retrieved from HALCOVE model with different vector length N and the accuracy data from ALCOVE. The equation is

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (V_i^{HALCOVE} - V_i^{ALCOVE})^2$$

where V_i is the data from the HALCOVE with different length N and V_i^{ALCOVE} is the data from the ALCOVE model.

From the equation, when all data are matched with the data from ALCOVE, the MSE value will be 0. It means we can check the similarity of two models by comparing the MSE

value. Smaller MSE value means closer performances for the two models. Our expectation is to get an MSE value lower than 0.01. Less than one percent different on average for considering that two models are the same.

CHAPTER IV.

Results

In this chapter, we show the results of experiments for discrete data and continuous data, respectively, since the implementation of two data sets are different.

Discrete data

In implementing the ALCOVE model, learning accuracy is recorded every five epochs and the performance accuracy of six categories for the discrete model in the learning procedure is plotted (Figure 3). As we discussed in the method above, these six learning problems are set from easy to difficult. The type one problem requires only one dimension to solve the task; the type two problem needs two dimensions. All other problems require all three dimensions. In the neural network models without dimension attention, the curve of the type two problem shows slower learning than some of more difficult types of problem [4]. So Kruschke inserted dimensional attention into the model and the model learned all problems at a relative performance level comparable to human learners. The speed to learn these problems matches the difficulty of the problem settings. The curve of type one problems is fastest followed by learning the type two problem. And these are learned faster than other higher numbered type problems which require all dimensions to facilitate learning.

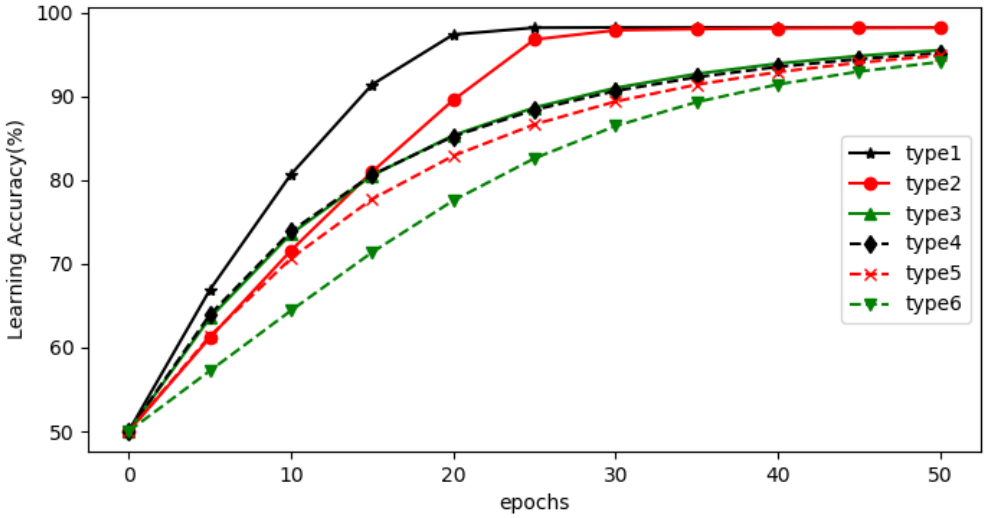


Figure 3: ALCOVE performance for discrete data

In HALCOVE, three pairs of vectors with length N are created for testing the performance of the model. In HRRs [8], the vectors have some requirements as mentioned above such as orthogonality. Plate suggested using a normal distribution with a mean value of 0 and a variance of 1/N to generate the elements for the vectors, followed by unitization.

When the vectors are generated and processed to meet all requirements, they are fed into the HALCOVE model which is designed for the vectors. To create an equal environment for the comparison, they still learn 50 times and we record the learning accuracy every 5 epochs. Different lengths of vectors N are tested for the model and the results are plotted (Figures 4, 5, 6, 7, 8 and 9).

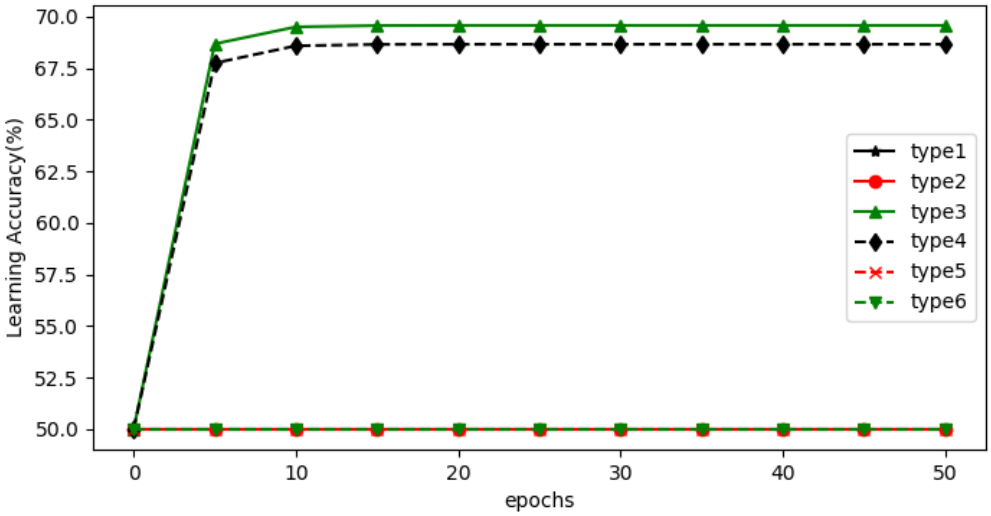


Figure 4: HALCOVE performance for discrete data with N = 2

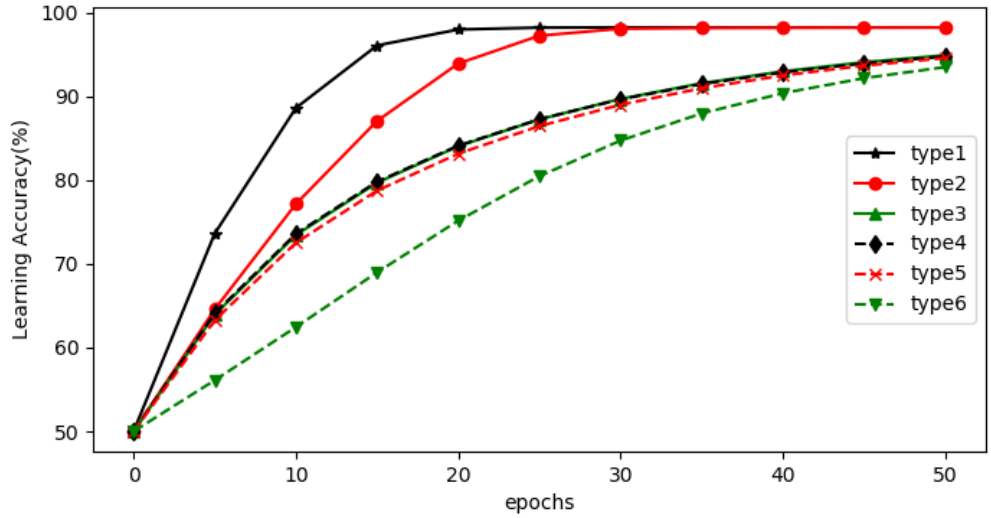


Figure 5: HALCOVE performance for discrete data with N = 4

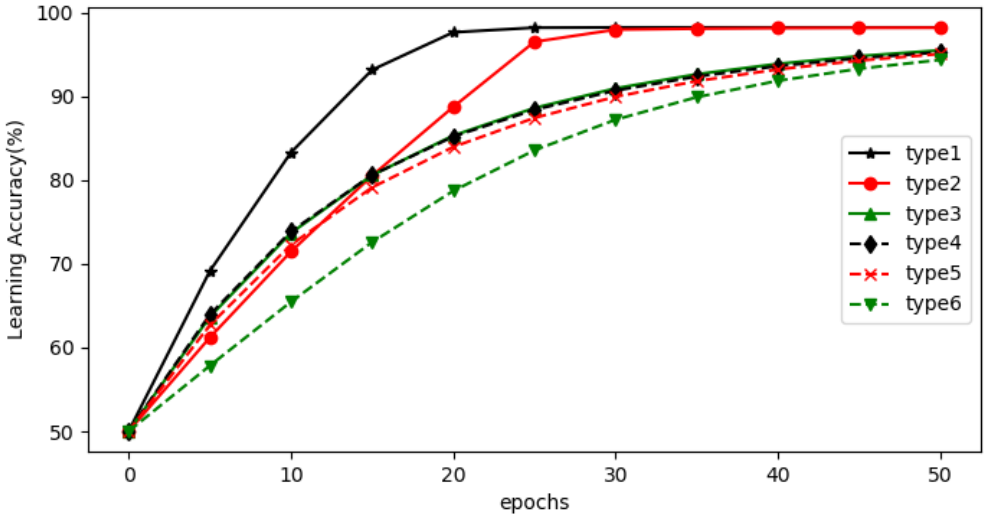


Figure 6: HALCOVE performance for discrete data with N = 16

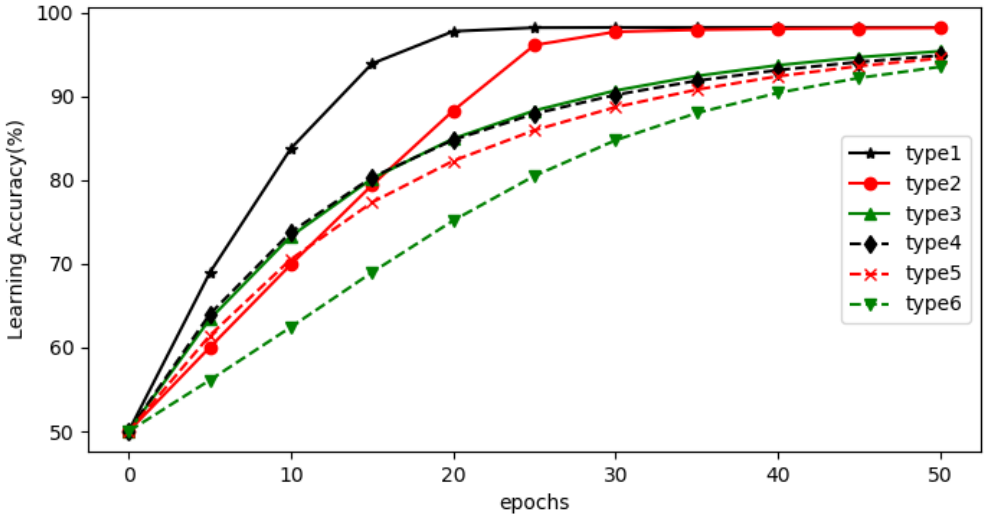


Figure 7: HALCOVE performance for discrete data with N = 64

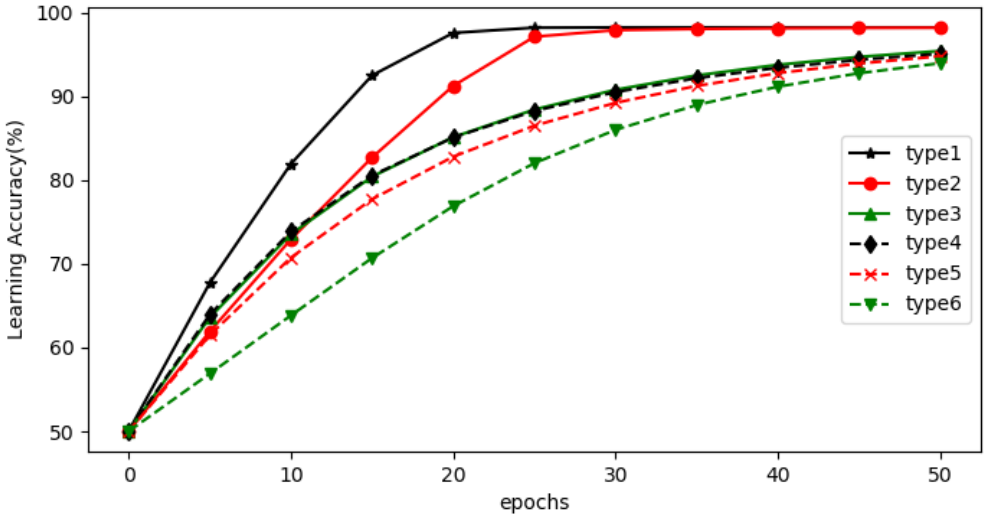


Figure 8: HALCOVE performance for discrete data with N = 256

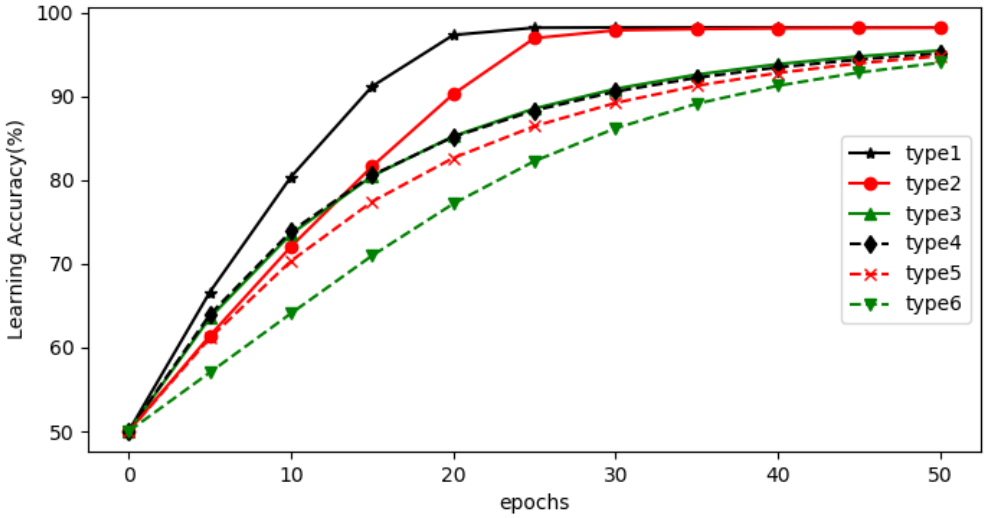


Figure 9: HALCOVE performance for discrete data with N = 1024

When compared to ALCOVE (Figure 3), the learning tendency and accuracy in HALCOVE are similar to each type of problem in ALCOVE (Table 1), much like human learning. However, the curves plotted with a small N have distortions, only type 3 and type 4 problems have learning behavior while others do not learn at all (Figure 4). It means the model has

a threshold beyond which it starts to be similar to ALCOVE. There is a slight difference in the learning behavior when N changs(N>=4 in this experiment) but the general learning tendency is the same: a larger N makes HALCOVE model's learning curve closer to ALCOVE.

Table 1 shows the mean squared error we mentioned in methods to compare the difference between HALCOVE and ALCOVE. The data of HALCOVE models are from Figures 4, 5, 6, 7 and 8 and the data for ALCOVE model is from Figure 3.

Table 1: Distance between the data of HALCOVE with different length vectors and the data of ALCOVE for discrete data

Compared Objects	**Mean Squared Error**
ALCOVE	0
HALCOVE with N = 2	1018.42
HALCOVE with N = 4	4.26656
HALCOVE with N = 16	0.919278
HALCOVE with N = 64	0.472935
HALCOVE with N = 256	0.218548
HALCOVE with N = 1024	0.0404466
HALCOVE with N = 4096	0.0161562
HALCOVE with N = 8192	0.00623279

From Table 1, when the data exactly matches the data in ALCOVE, the mean squared error (MSE) value will be 0. This means that smaller the value is, the more similar the data is to the data produced by ALCOVE model. The MSE value gets smaller when the vector length becomes bigger in HALCOVE. So HALCOVE model with larger length vectors get closer performance to the ALCOVE model.

Continuous data

We divide our experiment into two parts because the ways we build the hidden layer for discrete data sets and continuous data sets are different. We use the convolutional power in building the hidden layer for the continuous data. We raise the power for the feature vector of each dimension according to the values of patterns in data set. When the hidden layer is built, the learning procedure is the same as what we do in HALCOVE for the discrete data. We use the cosine distance to compute the distance between two vectors and then use it for learning and backpropagation.

We use two sets of continuous data in this section; one is called integral data and the other is called separable data. The first step is still to run them in the ALCOVE model and get the plots (Figure 10 and 16).

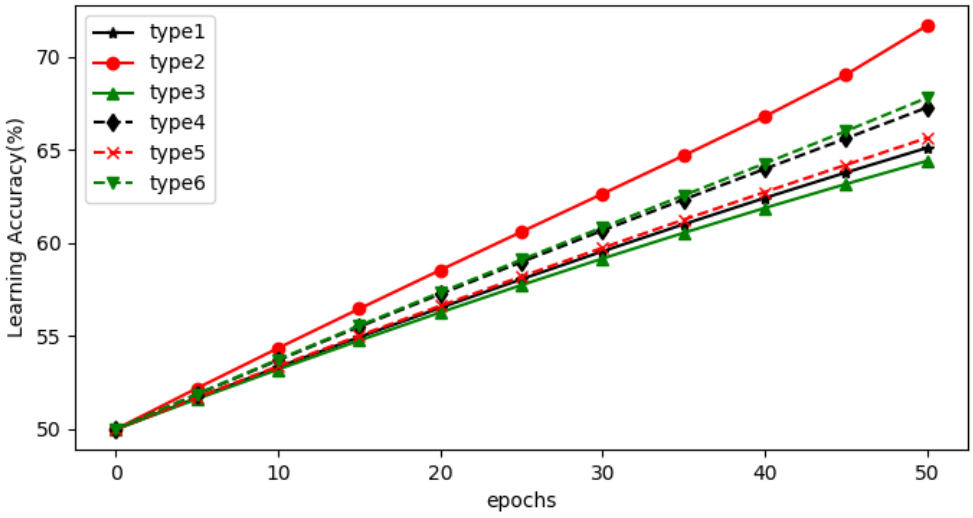


Figure 10: ALCOVE performance for integral data

The performances in the plot are far from the performance in discrete data sets. For learning accuracy, ALCOVE with continuous data sets learns more slowly. We can see from the plots that the highest learning accuracy in continuous data sets is the type two problem in integral data sets and it only arrives at around 70% after 50 epochs of training.

Then we implement HALCOVE with the methods described above and feed the data into it with different length N of vectors and plot the performances for integral and separable data sets. We will compare the integral data first (Figure 11, 12, 13, 14 and 15) and then the separable data:

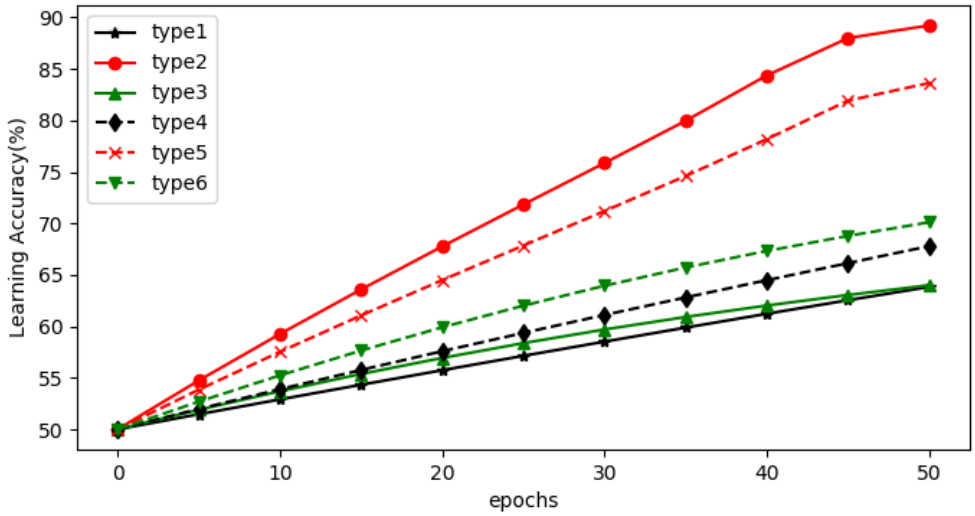


Figure 11: HALCOVE performance for integral data with N = 4

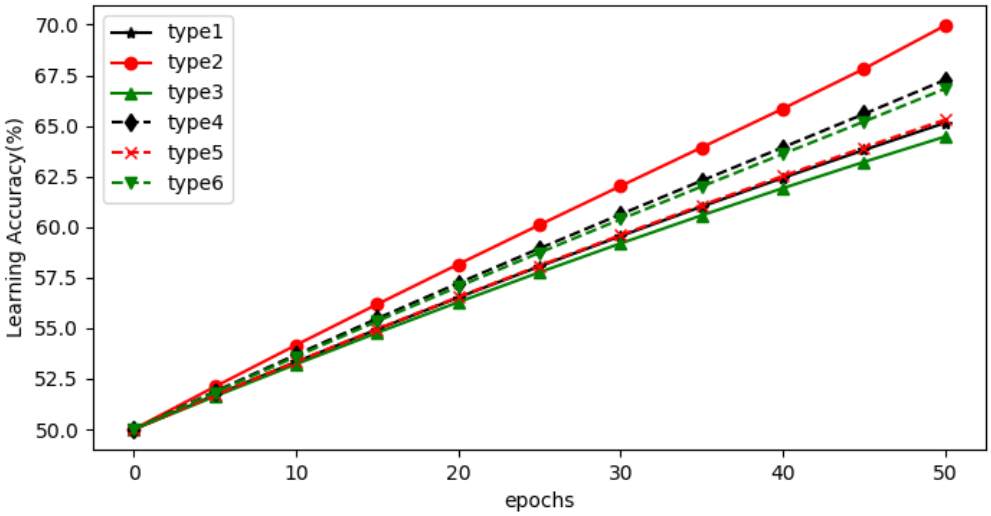


Figure 12: HALCOVE performance for integral data with N = 16

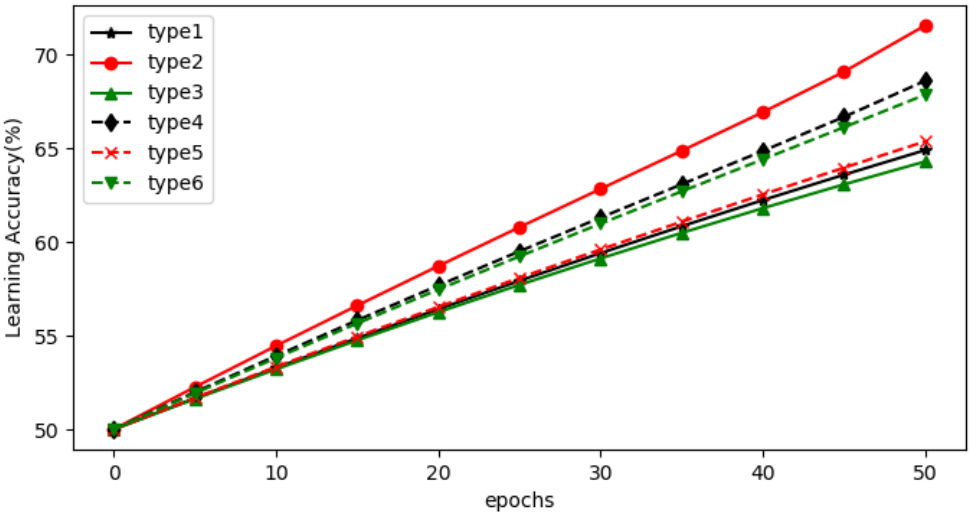


Figure 13: HALCOVE performance for integral data with N = 64

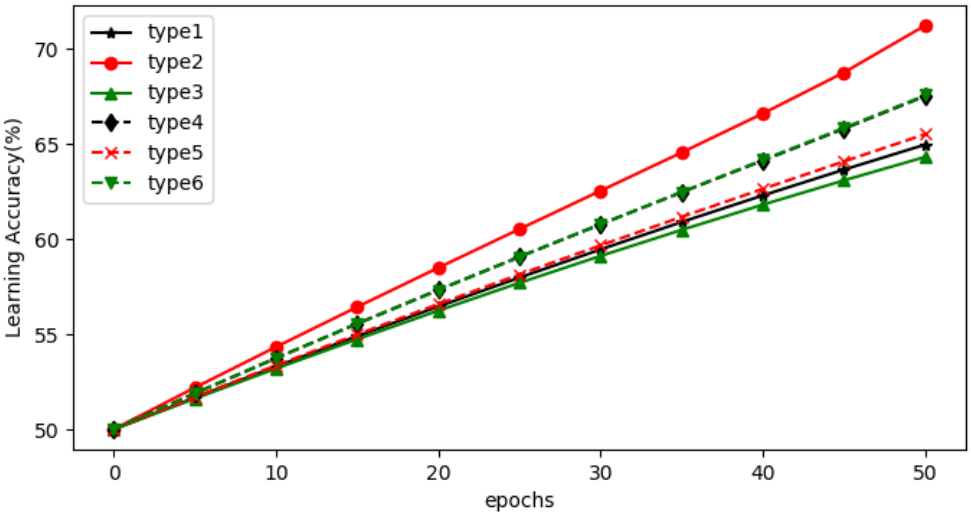


Figure 14: HALCOVE performance for integral data with N = 256

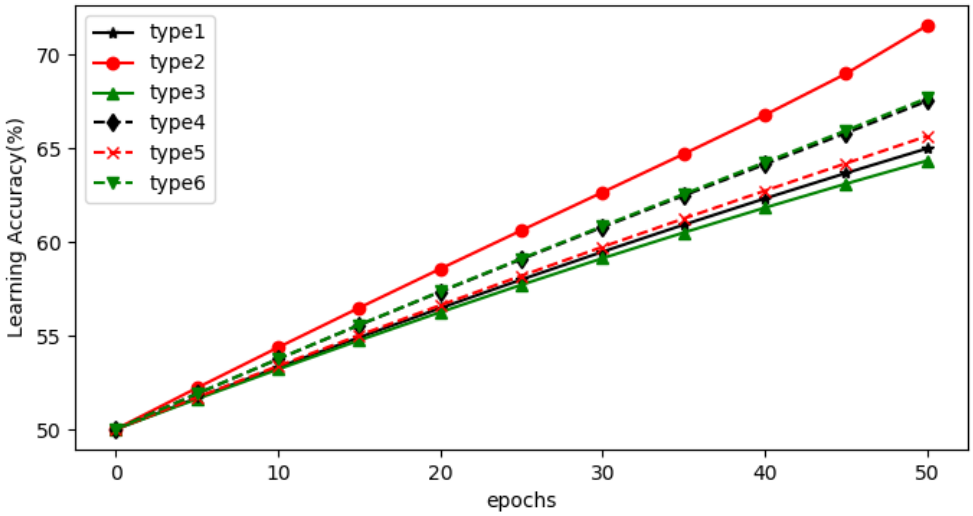


Figure 15: HALCOVE performance for integral data with N = 1024

Table 2: Distance between the data of HALCOVE with different length vectors and the data of ALCOVE for integral data

Compared Objects	**Mean Squared Error**
ALCOVE	0
HALCOVE with N = 4	47.8247
HALCOVE with N = 16	0.14802
HALCOVE with N = 64	0.0915405
HALCOVE with N = 256	0.0106795
HALCOVE with N = 1024	0.00447099

From Figures 11, 12, 13, 14 and 15 and Table 2 above, it is clear to see that the learning performance of HALCOVE is getting closer to ALCOVE when we use longer length vectors for the learning. When the length is small, the HALCOVE model can not learn (Figure 11) because the vector cannot hold enough information; this is the same as we found in the HALCOVE model for discrete data.

Then we show the results for the separable data sets, the first figure is the plot of ALCOVE model (Figure 16):

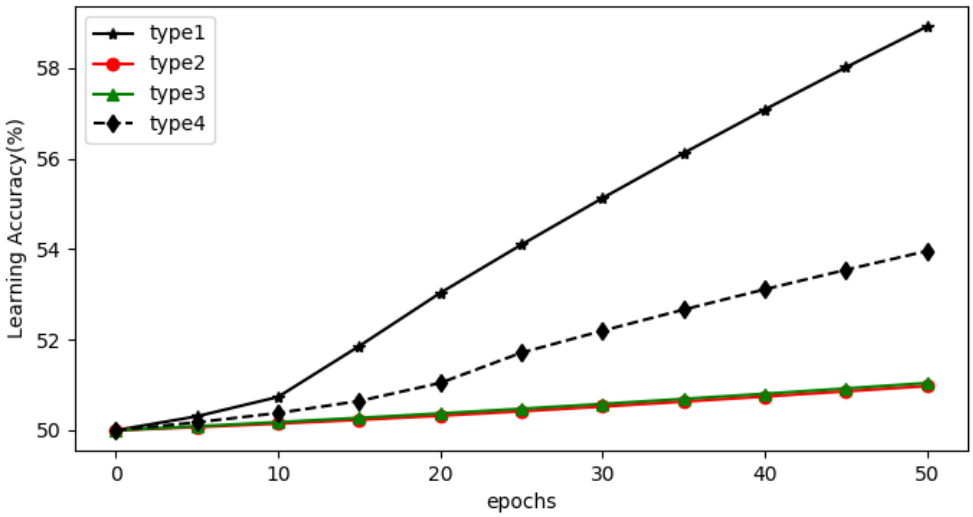


Figure 16: ALCOVE performance for separable data

Two types of problems in separable data sets show almost no gain in learning accuracy because the accuracy stays near 50%. The highest learning accuracy can reach around 58% in problem type 1.

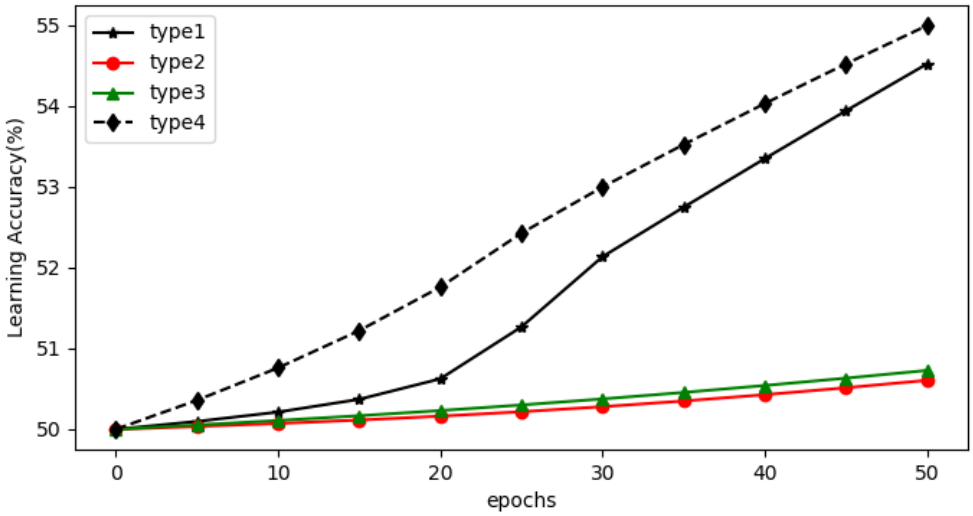


Figure 17: HALCOVE performance for separable data with N = 4

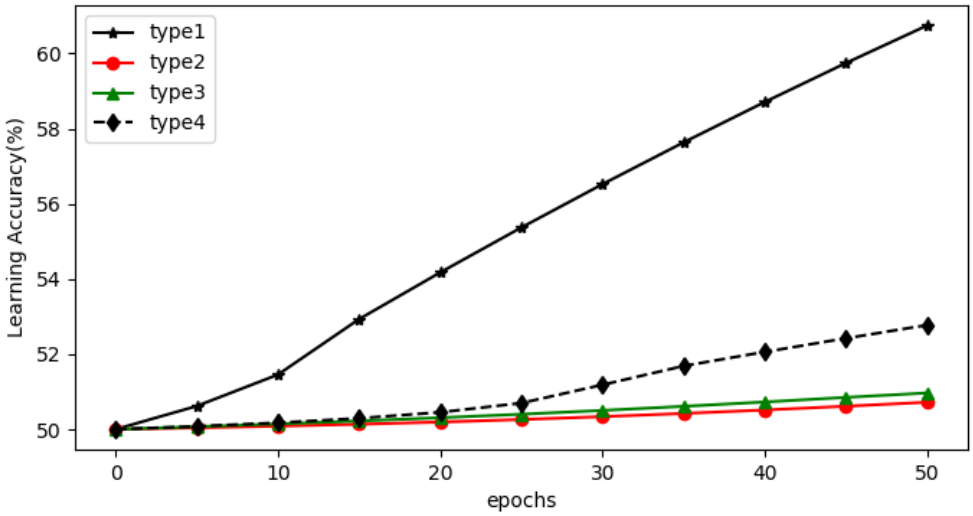


Figure 18: HALCOVE performance for separable data with N = 5

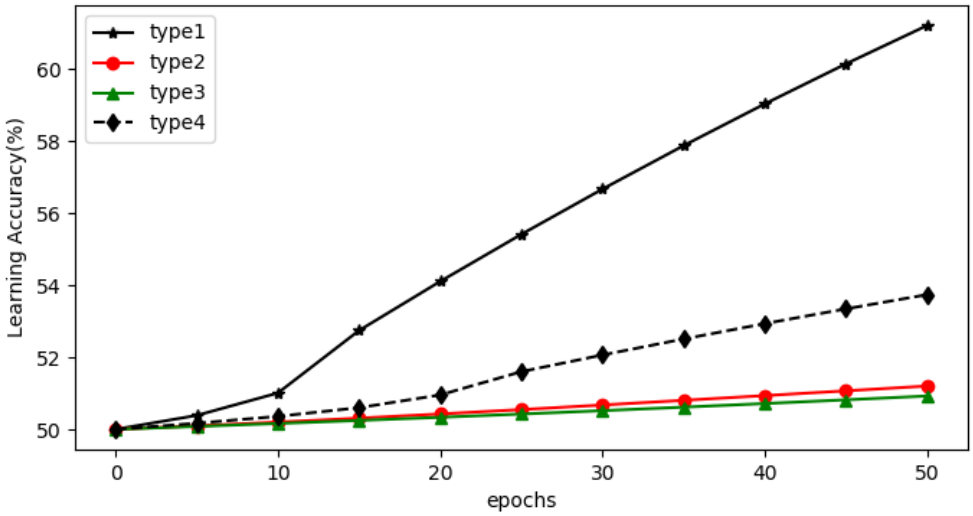


Figure 19: HALCOVE performance for separable data with N = 6

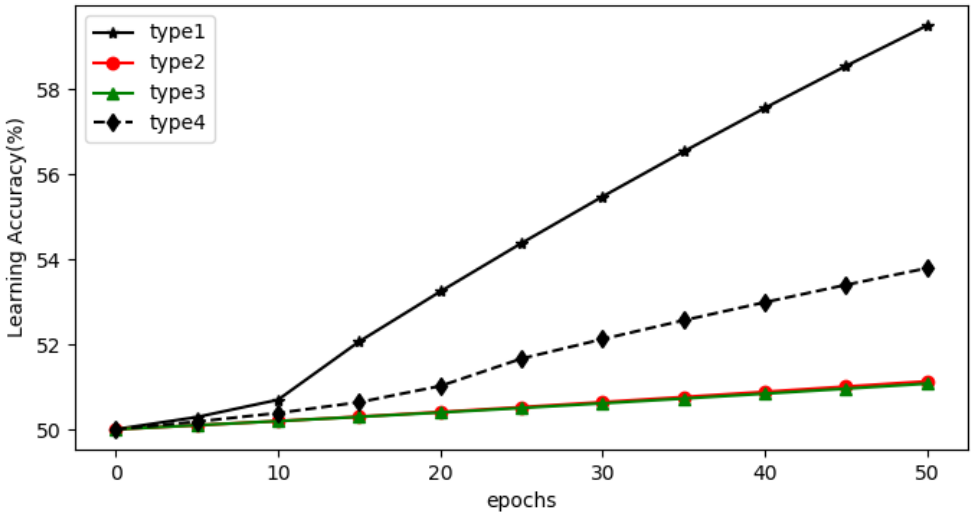


Figure 20: HALCOVE performance for separable data with N = 7

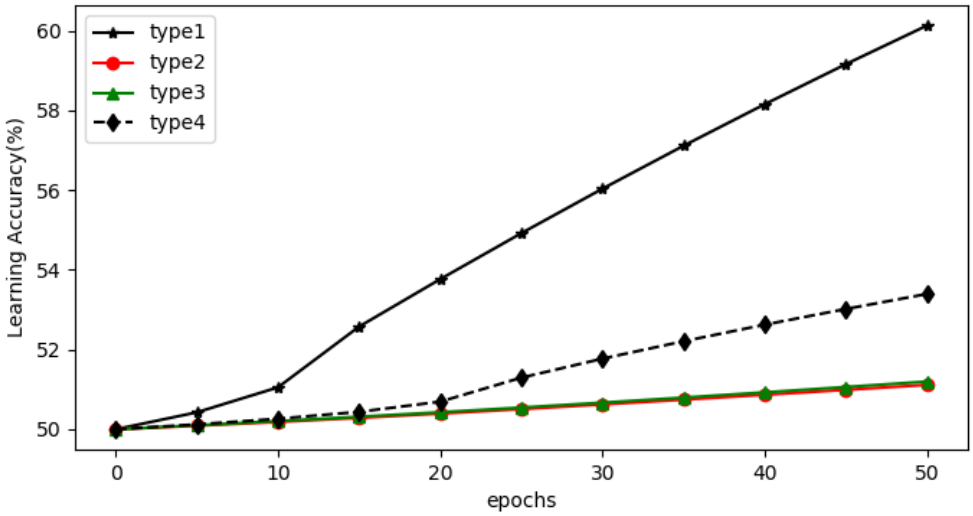


Figure 21: HALCOVE performance for separable data with N = 8

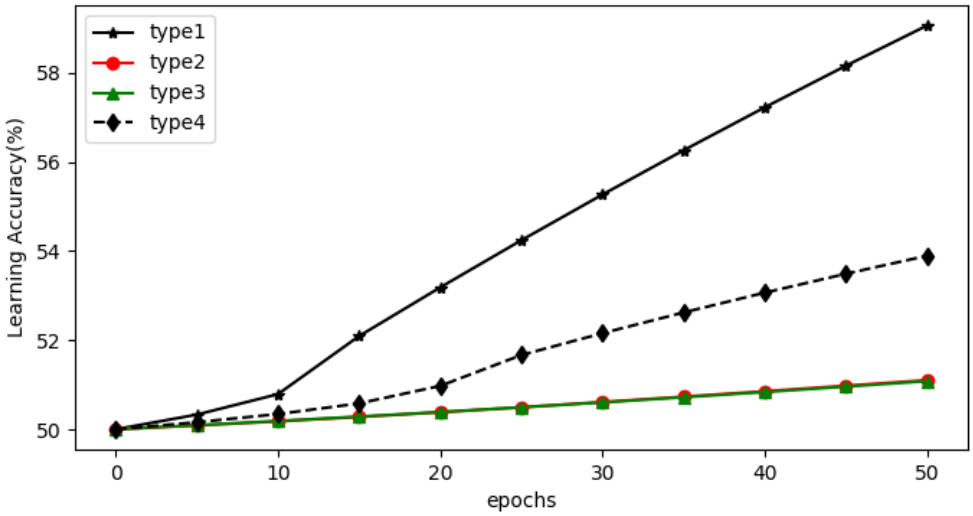


Figure 22: HALCOVE performance for separable data with N = 16

Table 3: Distance between the data of HALCOVE with different length vectors and the data of ALCOVE for separable data

Compared Objects	**Mean Squared Error**
ALCOVE	0
HALCOVE with N = 4	2.12213
HALCOVE with N = 5	0.581388
HALCOVE with N = 6	0.529589
HALCOVE with N = 7	0.34753
HALCOVE with N = 8	0.212212
HALCOVE with N = 16	0.006824

Figure 17, 18, 19, 20, 21 and 22 show the intuitive learning performance for the separable data sets in the HALCOVE model. Table 3 compares the similarity between HALCOVE with different lengths and the ALCOVE model. The MSE value decreases when the length of vectors increases.

Figure 17 does not show the similar learning curve to the ALCOVE model with the same data. The reason is still the same as we aforementioned; the length of vector is too short to hold the information and noise is severe when convolution and correlation are applied on short length vectors.

CHAPTER V.

Discussion and Conclusions

The new HALCOVE model demonstrated that convolution could be applied in the ALCOVE model and still maintained similar performance. This model overcame the weakness of the original model that it could only learn three-dimensional objects in the experiment by incorporating convolution and correlation. The new model included a new compression method from Holographic Reduced Representations(HRRs) and adjusted the way of representing the dimensions from numbers to vectors and tested it with the data from the ALCOVE model. For the three-dimension problem, the performance was similar and reached the expected performance from the data shown in Table 1.

For the continuous data sets, we also prove that convolutional power can be applied in the HALCOVE model. It uses one feature vector for each dimension in the data sets and raises it to different powers to build the hidden layer for learning. Table 2 and Table 3 show that two models will get almost the same performance when we use longer vectors in learning. Compared to discrete data sets, it's easier for continuous data to reach our expectation due to the datas and learning paramter settings.

Obtaining expected results from this new model HALCOVE means that convolution and correlation can be used to represent the dimensions without losing information in the learning and this could increase the dimensions available for the inputs without losing accuracy. It allowed more features in learning, which could be plugged easily in other learning experiments.

There are also some shortcomings in the HALCOVE model which might lead to future work. The model is still restricted to the ALCOVE model which has an input layer, attention weights, an association matrix and an output layer. It still used backpropagation to adjust the attention weights and the association matrix. Reinforcement Learning of Dimensional Attention for Categorization [7] suggests that Temprol Difference(TD) Learning could be

used to HALCOVE and make it more biological without losing accuracy. For future work, this approach could also be incorporated into the model proposed in this paper to make the learning behavior closer to human learning.

BIBLIOGRAPHY

[1] BORSELLINO, A., AND POGGIO, T. Convolution and correlation algebras. *Kybernetik 13*, 2 (1973), 113–122.

[2] DUBOIS, G. M., AND PHILLIPS, J. L. Working memory concept encoding using holographic reduced representations. In *MAICS* (2017), pp. 137–144.

[3] JOVANOVICH, M., AND PHILLIPS, J. n-task learning: Solving multiple or unknown numbers of reinforcement learning problems. In *CogSci* (2018).

[4] KRUSCHKE, J. K. Alcove: an exemplar-based connectionist model of category learning. *Psychological review 99*, 1 (1992), 22.

[5] LILLICRAP, T. P., COWNDEN, D., TWEED, D. B., AND AKERMAN, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications 7* (2016), 13276.

[6] NOSOFSKY, R. M. Choice, similarity, and the context theory of classification. *Journal of Experimental Psychology: Learning, memory, and cognition 10*, 1 (1984), 104.

[7] PHILIPS, J. L., AND NOELLE, D. C. Reinforcement learning of dimensional attention for categorization. In *Proceedings of the Annual Meeting of the Cognitive Science Society* (2004), vol. 26.

[8] PLATE, T. A. Holographic reduced representations. *IEEE Transactions on Neural networks 6*, 3 (1995), 623–641.

[9] WILLIAMS, A., AND PHILLIPS, J. Multilayer context reasoning in a neurobiologically inspired working memory model for cognitive robots. In *CogSci* (2018).