A RADIAL BASIS MEMORY MODEL
FOR HUMAN MAZE LEARNING

by

Lisa Y. Drewell

A thesis submitted to the School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen’s University
Kingston, Ontario, Canada
(May, 2008)

Copyright © Lisa Y. Drewell, 2008
Abstract

This research develops a memory model capable of performing in a human-like fashion on a maze traversal task. The model is based on and retains the underlying ideas of Minerva 2 but is executed with different mathematical operations and with some added parameters and procedures that enable more capabilities. When applied to the same maze traversal task as was used in a previous experiment with human subjects, the performance of a maze traversal agent with the developed model as its memory emulated the error rates of the human data remarkably well. As well, the maze traversal agent and memory model successfully emulated the human data when it was divided into two groups: fast maze learners and slow maze learners. It was able to account for individual differences in performance, specifically, individual differences in the learning rate. Because forgetting was not applied and therefore all experiences were flawlessly encoded in memory, the model additionally demonstrates that error can be due to interference between memories rather than forgetting.
Acknowledgements

I would like to thank my supervisor, Dr. Roger Browse for his wonderful insight, assistance, patience and dedication throughout this project. I would also like to extend sincere thanks to Dr. Brian Butler who provided valuable background information, Dr. Selim Akl who contributed to broadening my academic experience at Queen's and Dr. Robin Dawes for his helpful suggestions. I greatly appreciate Debby Robertson for her invaluable assistance and Sunny Gupta for being so generous with his technical expertise. Finally, I thank Eddie and my parents Loraine and Norbert for their constant support and encouragement.
# Table of Contents

Abstract ................................................................................................................................. i
Acknowledgements ............................................................................................................. ii
Table of Contents ................................................................................................................. iii
List of Figures ......................................................................................................................... v

## Chapter 1 INTRODUCTION ......................................................................................... 1
1.1 The maze navigation task ............................................................................................... 3
1.2 The Radial Basis Memory Model .................................................................................. 3
1.3 Modeling human performance for the maze navigation task ........................................ 5
1.4 Contributions ................................................................................................................ 6
1.5 Outline .......................................................................................................................... 6

## Chapter 2 BACKGROUND AND LITERATURE REVIEW ...................................... 8
2.1 Memory ......................................................................................................................... 8
    2.1.1 Global memory models .......................................................................................... 9
    2.1.2 Multiple-trace memory models .......................................................................... 10
    2.1.3 Minerva .............................................................................................................. 10
2.2 Radial basis functions ................................................................................................ 16
2.3 Browse and Gray’s Experiment B ................................................................................ 17
    2.3.1 Equipment and arrangement .............................................................................. 18
    2.3.2 The mazes ......................................................................................................... 18
    2.3.3 Procedure ......................................................................................................... 22
    2.3.4 Results ............................................................................................................. 22

## Chapter 3 CRITIQUE OF MINERVA ........................................................................ 24
3.1 The negative similarity problem .................................................................................. 24
3.2 The inconsistent treatment of the symbolic zero ......................................................... 26
3.3 Representing the real world with -1’s and 1’s ............................................................. 28
3.4 Interpolating into an undefined spaced followed by re-probing ................................ 29

## Chapter 4 A RADIAL BASIS MEMORY MODEL ....................................................... 31
4.1 Experiences and memory structures .......................................................................... 32
4.2 A trace’s utility and characteristic radius parameters ............................................... 35
List of Figures

Figure 1. The Minerva calculations involve probing secondary memory with a sample probe and three sample traces to arrive at an echo. The real-numbers are rounded off for display purposes. .......................................................... 13

Figure 2. The experimental setup and a view of inside a virtual maze. Reproduced from Gray (2005). .......................................................... 19

Figure 3. The mazes (1 to 6) used by Browse and Gray. Adapted from Gray (2005). .................. 20

Figure 4. The mazes (7 to 12) used by Browse and Gray. Adapted from Gray (2005). ........... 21

Figure 5. The total mean navigation error for the Head-Tracked Perspective and Fixed Position Perspective conditions. The navigation error is the percentage of wrong turns made out of the total number of T-intersections during a given attempt at a maze. Reproduced from Gray (2005). ........................................................................................................ 23

Figure 6. In Minerva, two opposite traces, $m$ and $m'$, make identical contributions toward the echo. This example is a special case in which an occurrence of the probe exists in memory as trace $m$. Because the probe and $m$ are identical, $m$ has the maximum possible similarity value of 1, and hence $m'$ has a similarity of -1. .......................................................... 25

Figure 7. A histogram of sample traces' (five identical and one unique) and probes' relative positions in space. Due to re-probing, $p_3$ to $p_4$, the consecutive echoes, $e_1$ to $e_3$, drift away from the trace that is closest to the initial probe, $p_1$, to settle on a more distant trace, $e_3$, that is part of a cluster of traces. .......................................................... 30

Figure 8. The current experience, called a probe when in primary memory, and the memories of past experiences, called memory traces when in secondary memory, are illustrated in terms of their features (the vertical lines dividing them are for display purposes only) which in turn are composed of varying numbers of components, whose values are real numbers in the range of -1 to 1, inclusive. The probe is missing values for its action feature (replaced with “nil”) because the action taken in the current situation has yet to be determined. There are two actions defined using two components of the action feature dedicated to each one. .......................................................... 33

Figure 9. Four sample traces and a probe are viewed as points in 2-dimensional space such that their two components are their coordinates in that space. These points are plotted in the top graph which also illustrates the area of influence that surrounds each trace. The area of influence is
defined by the trace’s characteristic radius. The gradation, which is to say the way in which a trace’s area of influence fades away, is according to a Gaussian function. The amplitude of a trace’s activation, which has been scaled by its utility, is given by the darkness of the trace’s area of influence. The probe does not have an area of influence but is simply a point that exists in the space and may or may not intersect traces’ areas of influence. The echo intensity is the sum over all activations, shown individually in the bottom graph.

Figure 10. The pseudo code that specifies the agent’s behaviour (normal text) and the RBMM’s behaviour (italic text) for all six attempts at a maze.

Figure 11. From left to right, the four features used for the maze traversal task, the instances of each of the four features, and a meaningful description of what the instances represent. The component values and the number of components dedicated to each feature were chosen by consideration of the relative weighting between and within features.

Figure 12. The percepts at every location where an experience can occur in an example maze.

Figure 13. The depths 0 to 6 at every location where an experience can occur in an example maze.

Figure 14. Dead reckoning at every location where an experience can occur in an example maze.

Figure 15. This instance of an action feature represents turning right. Hence, all four of the components assigned to a right turn are 1’s and all of the other components are 0’s.

Figure 16. The average human performance, and the RBMM’s average performance using the features: percept and depth, and the free parameter values: DefaultSigma = 2.2, SigmaScalar = 0.3, and UtilityScalar = 0.4. The error bars represent the standard deviation of the subjects’ average performance (black bars) and the model’s average performance (grey bars).

Figure 17. Each subject’s average performance on 4 mazes, produced from the experimental data from Browse and Gray (2006).

Figure 18. Fast versus slow maze learners’ average performance. The error bars represent the standard deviation of the fast learners’ average performance (black bars) and the slow learners’ average performance (grey bars).

Figure 19. The average of the fast maze learners’ performance and the RBMM’s average performance using the features: percept, depth and dead reckoning, and the free parameter values: DefaultSigma = 2.4, SigmaScalar = 0.7, and UtilityScalar = 0.7.
Figure 20. The average of the slow maze learners’ performance and the RBMM’s average performance using the features: percept and depth, and the free parameter values: 

DefaultSigma = 2.6, SigmaScalar = 0.25, and UtilityScalar = 0.4. ........................................... 65
Chapter 1

INTRODUCTION

Memory plays an important role in our daily lives as it supports just about all that we do. 

*Procedural memory* assists us in tasks from tying our shoelaces to navigating our way around campus. *Semantic memory* facilitates our knowledge, from summoning the meaning of words that we read to knowing how concepts are related. *Episodic memory* allows us to remember events, from recalling the face of the person who kindly opened a door for us this morning to knowing the details of our lives. Memory is an essential component of our intelligence that allows us to function productively in a variety of ways.

Modeling human memory is one approach to the study of memory that can lead to insights in understanding how memory operates. One such model that has enjoyed moderate success and much interest is the Minerva 2 (hereafter referred to simply as Minerva) model of human recognition memory that was developed to show that both *episodic memory* and memory for abstract concepts could be handled by a single system (Hintzman, 1984, 1986, 1988).

Perhaps because of the declarative nature of the information that models such as Minerva process, much of the testing done with memory models has been with the purpose of retrieving particular items from memory. That is, the task is to match a stimulus to the contents of memory to retrieve items from memory that are similar. However, a more well-rounded basic understanding of memory can be achieved by shifting the focus from testing the storage and retrieval capabilities
of a model to also finding out whether a model could be used when memory plays a supporting role within a broader task. By using memory to carry out tasks, the potential scope of a model such as Minerva, with a few modifications to handle actions, can be expanded to operate with simple stimulus-response pairings as in *procedural memory*.

Many of the tasks used in memory studies and modeled in formal memory systems involve the explicit memorization of stimuli. Memorization certainly has a place in our daily lives, especially if one is taking a biology course, but it is by no means the way in which a good portion of our memories are formed. Rather many memories are formed unintentionally and coincident with ongoing activities. Thus, the act of intentional memorization is not the primary use of memory. To gain a better understanding of memory, the tasks that are typical of our more usual use of memory ought to be addressed.

The purpose of this research is therefore to study the use of memory to carry out tasks for which the information is not necessarily memorized. More specifically, the goal is to develop a general memory model that can complete a maze navigation task such that its performance is similar to human performance for the same task. The idea is that in learning a route, memories are stored but retrieval occurs only in support of the navigation task. That is, the agent will need to match up current situations with past situations in memory, but only toward the purpose of knowing what action to take next. To this end, the Radial Basis Memory Model (RBMM) was developed and its performance on a maze traversal task was compared to human performance for the same task.
1.1 The maze navigation task

Data for human performance on the maze navigation task already exists from an experiment by Browse and Gray (2006). This maze navigation task required that subjects traverse each virtual maze from a start location to a goal location within the maze. Initially, the subjects had no idea of the structure of the maze, and so their performance was poor. Each time the goal location was achieved, the subject started again at the beginning of the same maze, for a total of six attempts for each maze. All mazes were devoid of any landmarks and consisted of six left/right decision points at identical T-intersections from which one direction led to a dead end and the other led to another T-intersection, or in the case of the final T-intersection, to the goal. When at a T-intersection, the subject had no indication of the result that could be obtained in either direction. Among the data collected was each subject’s average error rate for each of the six attempts at a maze, where error was defined as turning in the direction of a dead end at a T-intersection. The paradigm just described is also the paradigm that was used in the current work. However, taking the place of the human subjects is a simple maze traversal agent that is able to perceive the maze environment and act in it, and that is outfitted with the RBMM as its memory. Because the maze traversal agent is not equipped with any strategies that allow it to solve the mazes on its own, all relevant decisions are made by referring to its RBMM memory.

1.2 The Radial Basis Memory Model

Minerva (Hintzman, 1984, 1986, 1988) was chosen as the basis for the RBMM because it is an implementable, multiple-trace and global model of human recognition memory. As originated in Minerva, and incorporated into the RBMM, there are two types of memory, primary memory and secondary memory. Primary memory contains a representation of an event that will be used to
probe the collection of memory traces in secondary memory to determine if the event was experienced previously. Probing secondary memory involves comparing the probe, which is a representation of the current event, to each memory trace to check for similarities. Based on its similarity, each trace becomes activated to some degree and contributes accordingly to secondary memory’s response. This response, also referred to as an echo, is a collective representation of all traces in memory, each scaled by a factor that depends on its similarity to the probe. The second part of the echo is its intensity, or the strength of the echo response, which is a single scalar value and indicates the probe’s familiarity. In Minerva, the echo intensity has also been regarded as indicative of the number of times that the probe occurs as a trace in secondary memory.

Although the general ideas underlying Minerva have merit, there are issues with its mathematical procedures. These issues, which are described in detail in Chapter 3, are addressed in the RBMM.

In the RBMM, traces are encoded in secondary memory along with the action that followed the experience. A primary feature of the RBMM is a new probing calculation; the comparison between the probe and a trace involves calculating the Euclidean distance between them and scaling this number with a Gaussian as a radial basis function centered on the trace. The width of the Gaussian function is determined by the characteristic radius, which is stored as one of the parameters of the trace. The Gaussian’s characteristic radius determines the volume surrounding the trace in which the probe is considered somewhat similar to the trace. The trace’s other parameter, its utility, scales the trace’s contribution to the echo according to the success with which that trace has been applied in the past. With each new experience, these two parameters are updated according to a reinforcement learning procedure that operates under the assumption that there is feedback available about the effectiveness of every new experience in performing the
agent’s task. In this way, the model has the ability to learn from its experiences. The *echo intensity* formula is similar to that used in Minerva, and the *echo content* is an accurate interpolation operation which requires no re-probing. The *action* attribute of the *echo* is used to determine which *action* the agent takes.

### 1.3 Modeling human performance for the maze navigation task

In applying the RBMM to the maze task, an experience consists of three aspects of the maze environment: (1) the *percept* representing the agent’s sensed perception of being at a T-intersection, at a dead end, or at the goal, (2) the *depth* representing the agent’s estimation of how far into the maze it is, and (3) *dead reckoning* representing the agent’s estimation of whether it was facing North, South, East or West.

In carrying out the maze traversal task, the RBMM has three global, free parameters which relate to the initial value and rate of variation of the memory traces’ *characteristic radii*, and to the scaling of reinforcement associated with the *actions* taken. Values of these three parameters have been determined that allow the RBMM agent to learn, on average, in almost exactly the way that human subjects did for the same maze task. Because the parameter for forgetting was not used, the model additionally demonstrates that error can be due to interference rather than the loss of information.

The agent that uses the RBMM in the maze task does not possess any specific maze navigation algorithm. The agent simply experiences its world, tries to remember which *actions* were good or bad, and makes decisions about its *actions* on the basis of its memory of these past experiences.
The same memory structure and decision process would apply if the agent were carrying out a survival task in an environment of food and predators.

1.4 Contributions

The work that this thesis represents offers the following contributions:

- Identification and discussion of issues concerning Minerva, particularly issues relating to its mathematical procedures.

- The development of a memory model based on radial basis functions that:
  - is capable of learning mazes such that its performance models human performance for the same task.
  - can flexibly model the human performance of two groups of subjects.

- Taken together, the model’s methodology and performance show that performance errors can result from interference among memories, and because forgetting was not used, not necessarily from the absence of information that is a result of forgetting.

1.5 Outline

A review of multiple-trace and global memory models with particular emphasis on the Minerva model is presented in Chapter 2. The chapter also contains a review of radial basis functions and an outline of the maze navigation experiment that Browse and Gray (2006) conducted on human subjects. Chapter 3 provides a discussion of issues, primarily of mathematical soundness, that are
problematic for the Minerva memory model. The Radial Basis Memory Model is described in
detail in Chapter 4. Chapter 5 includes an explanation of how the RBMM was applied to
complete Browse and Gray’s navigation task and gives the results of modeling human
performance for the same task. Chapter 6 provides conclusions and offers ideas for future work
with the RBMM.
This chapter provides background information on implementable models of episodic recognition memory such as global memory models and multiple-trace memory models and an instance of these, Hintzman’s (1984, 1986, 1988) Minerva. Background information about radial basis functions, particularly those having a Gaussian-type dependence on radius, follows as well as the details of a virtual maze navigation experiment conducted on human subjects by Browse and Gray (2006).

2.1 Memory

In the Encyclopedia of Cognitive Science, Bjorkland, Schneider and Blasi (2003, p. 1059) define memory as “the mental storage of information and the processes involved in the acquisition, retention, and retrieval of that information.” A working definition of memory, as suggested by Neath (1998, p. 4) is “the ability to use or revive information that was previously encoded or processed.”

Recognition memory is used to judge whether an item or situation has been experienced previously. Recognition may occur due to a sense of familiarity or because the situation has been directly recalled. The models described here deal with recognition memory.
2.1.1 Global memory models

Global memory models were proposed as a solution to previous search models, which were too slow because they searched through all memories individually. They were also a solution to direct-access models where local matching occurs and which do not account for recognition being affected by non-target items in memory (Clark & Gronlund, 1996). Accessing memory collectively, or in parallel, was the solution afforded by global memory models.


The commonality underlying all of these global memory models is that cues from the current state of the world are combined into a single memory probe, which is used to activate memory broadly when testing long-term memory for recognition. This is in contrast to allowing access and comparison to individual items in memory. Rather, global memory models compare the probe to memory as a whole and this collective comparison produces a scalar index that is a measure of the resulting activation of memory and is furthermore an indication of the familiarity of the probe. Determining whether the scalar index, or measure of familiarity, indicates recognition, or lack thereof, is done according to signal detection theory, such that if the index’s value exceeds a certain criterion value, the model reports that it recognizes the probe, and if the index falls short of the criterion value, then the probe is not recognized.
2.1.2 Multiple-trace memory models

The key feature of multiple-trace memory models is that they store each memory in long-term memory individually so that if, for example, a particular event is experienced twice, the corresponding memory trace appears twice in memory. The fact that every experience is stored as its own memory trace allows the possibility of retrieving a single event. In order to produce a signal of familiarity when memory is probed, at the time of retrieval, the collection of memory traces generalizes its contents in a weighted fashion based on the probe to produce a response. SAM and Minerva are examples of multiple-trace models.

In contrast, distributed memory models, such as TODAM and the Matrix model, generalize event information at the time of storage making it impossible to retrieve information about specific events.

2.1.3 Minerva

An Introduction to Minerva

In an effort to demonstrate that a single memory system is able to account for both episodic memory (memory for individual experiences) and memories for abstract concepts, Hintzman (1984, 1986, 1988) conceived of Minerva. It is a model of human recognition memory that is designed to distinguish between old and new experiences, primarily of long-term memory, also known as secondary memory, rather than a kind of short-term memory referred to as primary memory. Minerva is a multiple-trace memory model in that each experience is encoded in secondary memory as its own memory trace, and is an instance of a global memory model in that it responds with an overall familiarity value when memory is probed as a whole.
Based upon just a few basic assumptions, this model of memory is fully specified and implementable, and as such, its performance can be and has been compared against human data. In several such comparisons, Minerva has been successfully applied to a variety of memory phenomena, namely frequency judgments, a variety of prototype effects (Hintzman, 1988) and list-length effects (Clark and Gronlund, 1996), and has been developed further by Dougherty, Gettys and Ogden (1999), Green and Kittur (2006), and Reichle and Perfetti (2003). In other experiments reviewed by Clark and Gronlund (1996), Minerva failed to match the data from human experiments.

The representation of event information
In Minerva (Hintzman, 1984, 1986, 1988), each event is represented as an ordered vector of length $N$. Every element of the vector may be considered a feature of the experienced event and each feature is assigned a value from the set $\{-1, 0, 1\}$. Feature values of -1 and 1 could be interpreted, respectively, as an inhibition and an excitation of the feature. A feature value of 0 (zero) indicates that the given feature is irrelevant, unknown, or has been forgotten.

When devising event vectors, the ones that represent independent concepts, or concepts which are unrelated, should be orthogonal. Similar events that are members of a category of events are essentially slightly distorted versions of an event vector that is prototypical for the category. That is, some of the prototype’s feature values are randomly assigned new values from the set $\{-1, 0, 1\}$ to derive other members in the category.
The components of Minerva

Minerva consists of two components: a primary memory or temporary buffer, and a secondary or long-term memory.

Primary memory is able to contain a single event vector. Inputs to and outputs from secondary memory are buffered in primary memory. That is, (a) experienced events are temporarily stored in primary memory before being encoded in secondary memory, (b) an event in primary memory can serve to probe how familiar that event is based on the contents of secondary memory, and (c) information retrieved from secondary memory is passed to primary memory.

Each experienced event is represented in secondary memory as its own memory trace. As such, secondary memory consists of a collection of memory traces and involves processes that allow secondary memory to be probed on a global level.

Storing event vectors as memory traces in secondary memory

When an event vector passes from primary memory to secondary memory for storage as a memory trace, its features are degraded according to the learning rate, L, such that 0 < L < 1.

Any given feature is either accurately stored (with probability L), or forgotten (with probability 1-L) in which case the feature value is stored as 0 (zero). Accuracy of encoding is better with larger values of L. For example, when L=0.7, an average of 7 out of 10 features are accurately stored. The learning rate causes a loss of information during the encoding process, not a distortion of information. That is, zeros replace lost, or forgotten, information. Information is never erroneously encoded, meaning a feature value of, for example, 1 is never encoded as -1.
An example of encoding an experience into *secondary memory* is as follows: A sample event represented as \( (0, 1, -1, 1) \) in *primary memory* reflects the current state of the world. This experience is then passed to *secondary memory* for storage. In the process, the vector is degraded according to a *learning rate* of, for example, \( L=0.8 \). Therefore, each *feature* is properly encoded with the probability 0.8. If, for example, only the third *feature* fails to be encoded and is forgotten, this *feature* has a value of 0. The result is that the *memory trace*, \( (0, 1, 0, 1) \), is added to the collection of *memory traces* in *secondary memory*. In fact, this is “Trace 1” in the example illustrated in Figure 1.

![Table](image)

<table>
<thead>
<tr>
<th>Type of memory</th>
<th>Label</th>
<th>Contents of memory</th>
<th>Similarity</th>
<th>Activation</th>
<th>Contribution to the echo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary memory</td>
<td>Probe</td>
<td>( 0 ) ( 1 ) (-1) ( 1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secondary memory</td>
<td>Trace 1</td>
<td>( 0 ) ( 1 ) ( 0 ) ( 1 )</td>
<td>( S_1 = \frac{0 + 1 + 0 + 1}{3} = \frac{2}{3} )</td>
<td>( A_1 = 0.30 )</td>
<td>( 0 ) ( 0.30 ) ( 0 ) ( 0.30 )</td>
</tr>
<tr>
<td></td>
<td>Trace 2</td>
<td>( 1 ) ( -1 ) ( -1 ) ( 0 )</td>
<td>( S_2 = \frac{0 + 1 + 1 + 0}{4} = \frac{1}{2} )</td>
<td>( A_2 = 0.13 )</td>
<td>( 0.13 ) ( 0.13 ) (-0.13 ) ( 0 )</td>
</tr>
<tr>
<td></td>
<td>Trace 3</td>
<td>( -1 ) ( 0 ) ( 0 ) ( -1 )</td>
<td>( S_3 = \frac{0 + 0 + 0 - 1}{4} = \frac{1}{4} )</td>
<td>( A_3 = -0.02 )</td>
<td>( 0.02 ) ( 0 ) ( 0 ) ( 0.02 )</td>
</tr>
</tbody>
</table>

**Figure 1.** The Minerva calculations involved in probing *secondary memory* with a sample *probe* and three sample *traces* to arrive at an *echo*. The real-numbers are rounded off for display purposes.
Probing secondary memory for recognition

Retrieving information from secondary memory occurs when a probe is present in primary memory. The probe has the effect of activating all vectors in secondary memory in parallel which respond with an overall familiarity value and a composite echo. All vectors in secondary memory contribute to these two components according to the extent of their similarity to the probe. The greater their similarity to the probe, the greater their contribution to the familiarity value and composite echo.

Figure 1 illustrates an example of the following calculations involved in probing memory.

As mentioned, all traces are compared to the probe to determine their similarity to it. The similarity, \( S_i \) of trace \( i \) to the probe, is given by

\[
S_i = \frac{\sum_{j=1}^{N} P_j T_{i,j}}{N_i}
\]

where \( P_j \) is the value of feature \( j \) in the probe, \( T_{i,j} \) is the value of feature \( j \) in trace \( i \), \( N \) is the total number of features and \( N_i \) is the number of features relevant to the comparison of the probe and trace \( i \), that is, the number of features that are non-zero in either the probe or the trace. Restated, \( N_i \) is \( N \) minus the number of features that are zero in both the probe and the trace.

The similarity acts as a correlation coefficient (Neath, 1998) in that when the probe and a trace are identical, their similarity is 1. When they are orthogonal, their similarity is 0, and in the case where they are the opposite of one another, their similarity is -1.
The extent to which a trace is activated by the probe is simply the cube of its similarity. Thus, trace $i$’s activation, $A_i$, is given by

$$A_i = S_i^3$$

While preserving the sign of the similarity, the cubing function allows traces that are strongly similar to the probe to dominate many ill-matching traces, effectively increasing the signal-to-noise ratio in the echo.

The activation of all of the traces in parallel produces an echo that has two properties, intensity and content. The echo intensity, $I$, is the sum of all of the activation levels of all of the $M$ traces in memory and is given by

$$I = \sum_{i=1}^{M} A_i$$

Because the activation of a strongly matching trace is near 1 and because the positive and negative activation levels of the other traces in memory should approximately balance to zero, the echo intensity is a measure of the familiarity of the probe and is used in frequency judgments and modeling human recognition memory.

The echo content, $C$, is a vector that is derived from the sum of all traces in memory weighted by their respective activation. The $j$th element of the echo content, $C_j$, is given by
Because the *probe* most strongly activates only the *traces* that are most similar to it, the *echo* reflects the characteristics of the *probe* as recalled by the *memory traces* that best match the *probe*. However, because these *traces* may contain information not present in the *probe*, this other information can appear in the *echo* as well. This is how Minerva accomplishes associative recall.

Hintzman (1984, 1986, 1988) offers a method, the *intertrace resonance algorithm*, to overcome Minerva’s ambiguous recall problem. That is, a *probe* that is exactly identical to a *trace* in memory can elicit an *echo* that is not identical to the *probe* and *trace*. Rather, other *traces* in memory interfere and add noise to the *echo* which produces a blurry match to the *probe* and *trace*. According to Hintzman’s algorithm to deblur the *echo*, the *echo* is “normalized” to be in the range of -1 and 1 by dividing all *features* of the *echo* by the largest absolute *feature* value. This is then used as a second *probe*, and the *echoes* are repeatedly normalized and used to re-probe memory until the *echo* arrives at a steady state at which time the process stops. At this point, the *echo* is often almost identical to the original *probe* and matching *trace*.

### 2.2 Radial basis functions

Radial basis functions are a class of real-valued continuous functions that are radially symmetric about the center. The radial basis function \( \varphi \) has the form

\[
\varphi(\|x - c\|)
\]
where $\| x - c \|$ represents the distance between a point, $x$, and the center, $c$, in an $n$-dimensional coordinate space, where $n$ is a number appropriate to the situation at hand.

A commonly used type of radial basis function that decreases monotonically with distance is the Gaussian function, which has the form

$$
\varphi(\| x - c \|) = e^{-\left(\frac{\| x - c \|}{\sigma}\right)^2}
$$

where $\sigma$ is a parameter that is referred to here as the *characteristic radius*.1

Artificial neural network researchers have used radial basis functions as the activation function that is executed at nodes in the hidden layer to create Radial Basis Function Networks (Mehrotra, Mohan & Ranka, 2000).

### 2.3 Browse and Gray's Experiment B

Browse and Gray (2006) conducted an experiment to investigate whether the accuracy of perspective projection affected human performance for traversing mazes in a virtual environment. In the Head-Tracking Perspective condition, the position of the subject’s head was tracked and the display was updated to reflect the correct perspective projection. In the Fixed Position Perspective condition, the head tracking device was disabled and perspective projection was rendered according to a fixed head position. The results indicated that there was no significant difference in the mean navigational error between the two conditions.

---

1 Although $\sigma$ is used in the expression for the Gaussian function, it is used somewhat differently than is the normal practice, and in any case is not related to the standard deviation statistic. To make this distinction clear, the term *characteristic radius* is used when referring to $\sigma$. 
The relevant details about the human performance for the maze traversal task used in the experiment follow. Although Browse and Gray (2006) recorded data about the time taken for a subject to complete each attempt a maze, only the data about the percentage of wrong turns taken by the subjects is reported here.

### 2.3.1 Equipment and arrangement

Subjects were seated facing three 20” LCD displays upon which a virtual maze appeared. Using a steering wheel and pedal, subjects were able to drive through the mazes displayed in the LCDs. As well, subjects wore a helmet that had a sensor for an electromagnetic motion tracking device affixed to it. The sensor was connected via a cable to the device’s transmitter that was located in a wooden box behind the subjects. The motion tracking device allowed the perspective projection of the scene to be updated according to the position of the subject’s head. The setup appears in Figure 2, reproduced from Gray (2005).

### 2.3.2 The mazes

The twelve mazes used in the experiment appear in Figure 3 and Figure 4. All of the mazes consisted of a series of six decision points at identical T-intersections. That is, the mazes were constructed so that path solutions always contained exactly three left and three right turns, although the sequence of the turns varied. As well, in an effort to make the sequence of lefts and rights in the correct path to the goal appear random to the subjects, none of the mazes involved more than two consecutive lefts or rights. The views from a T-intersection in both the left and right directions were identical. The incorrect direction led directly to a dead end and the correct
Figure 2. The experimental setup and a view of inside a virtual maze. Reproduced from Gray (2005).

direction led to the next T-intersection in the maze. At the sixth and final T-intersection, an incorrect decision led to a dead end and a correct decision led to the goal. In order to prevent subjects from backtracking in the maze, away from the goal, “breadcrumbs” in the form of red squares were left along the path that subjects took through the maze.
Figure 3. The mazes (1 to 6) used by Browse and Gray. Adapted from Gray (2005).
Figure 4. The mazes (7 to 12) used by Browse and Gray. Adapted from Gray (2005).
2.3.3 Procedure

The 10 subjects completed six attempts at each of 5 mazes (out of the 12 possible mazes), such that the mazes that one subject traversed were not necessarily the same mazes as another subject experienced. The first maze was a practice maze to acquaint the subject with the task and environment. The experimental data were collected during the four maze tasks that followed. These mazes were presented under alternating Fixed Position Perspective and Head-Tracked Perspective conditions such that the condition changed after the completion of 6 attempts at a maze. Half of the subjects began in the Fixed Position Perspective condition and half began in the Head-Tracked Perspective condition. The subjects were not informed about the experimental condition in which they were performing.

2.3.4 Results

There was quite a bit of variation in individual results. However, all of the individual results improved significantly over the course of the attempts.

The number of wrong turns taken at T-intersections in a given attempt at a maze was divided by the total number of T-intersections (six) in the maze to get a subject’s navigation error during a particular attempt. There was no significant difference found in the mean navigation error between the Fixed Position Perspective and Head-Tracked Perspective conditions. Figure 5, reproduced from Gray (2005), depicts the overall mean navigation error in these conditions across all subjects for each attempt.
Figure 5. The total mean navigation error for the Head-Tracked Perspective and Fixed Position Perspective conditions. The navigation error is the percentage of wrong turns made out of the total number of T-intersections during a given attempt at a maze. Reproduced from Gray (2005).
Chapter 3

CRITIQUE OF MINERVA

Minerva (reviewed in section 2.1.3) was chosen as the basis for the new Radial Basis Memory Model because of its success and its simplicity, and because it is a multiple-trace, and a global memory model. However, to set the context for the new model, this chapter is devoted to a critique of Minerva in order to identify the aspects of the model that can be improved.

3.1 The negative similarity problem

When probing memory, Minerva uses a similarity measure that can produce negative values. That is, the similarity ranges from -1 to 1 for probes and traces ranging from opposite to identical. Cubing the similarity to ascertain the resulting activation level preserves the sign of the similarity. The subsequent echo content calculation propagates the sign of the similarity toward the echo as the activation scales the trace in question. The effect of allowing negative similarity values is that two traces, m and m', that are “opposites”, $m' = (-1 \cdot m)$, make identical contributions to the echo. Moreover, the more dissimilar a trace is to the probe, the more negative its activation and therefore the stronger it mimics its opposite in its contribution to the echo.

Specifically, for a memory trace, m, that has a positive similarity value, s, and therefore a positive activation level, a, then m’s contribution to the echo is $(a \cdot m)$. For the opposite memory trace m' such that $m' = -m$, the similarity is s' such that s' is $-s$, the activation level is $a'$ such that $a'$ is $-a$, and the echo contribution is $(a' \cdot m')$. The echo contribution of m' can be rewritten as $(a \cdot -m)$,
which simplifies to \((a \cdot m)\), the exact echo contribution of \(m\). Therefore, both a trace \(m\) and its opposite \(m'\) contribute equally to the echo with the value \((a \cdot m)\). An example of this, particularly the special case in which \(m\) is identical to the probe, appears in Figure 6. The example demonstrates that a memory which is as dissimilar as possible to the probe will have exactly the same contribution to the echo as a memory that is identical to the probe.

Also illustrated in Figure 6 is a secondary issue that when both \(m\) and \(m'\) exist in memory, their activations cancel one another out so that their combined contribution to the echo intensity, which is a measure of the familiarity as it is an estimate of the number of times the probe occurs in

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{Type of memory} & \text{Label} & \text{Contents of memory} & \text{Similarity} & \text{Activation} & \text{Contribution to the echo} \\
\hline
\text{Primary memory} & \text{Probe} & \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} & \left( \frac{1}{N} \sum_{j=1}^{N} t_{ij} \right) & \left( A_i = S^3 \right) & \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \\
\text{Secondary memory} & \text{Trace } m & \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} & s = \frac{1 + 1 + 0 + 1}{3} = 1 & a = 1 & \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \\
& \text{Trace } m' & \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \end{bmatrix} & s' = \frac{-1 - 1 + 0 - 1}{3} = -1 & a' = -1 & \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \\
\hline
\end{array}
\]

\[
\text{Intensity} = \sum_{i=1}^{M} A_i \\
\text{Echo content} = \left( l = \sum_{i=1}^{M} A_i \right)
\]

\(25\)

**Figure 6.** In Minerva, two opposite traces, \(m\) and \(m'\), make identical contributions toward the echo. This example is a special case in which an occurrence of the probe exists in memory as trace \(m\). Because the probe and \(m\) are identical, \(m\) has the maximum possible similarity value of 1, and hence \(m'\) has a similarity of -1.
memory, is 0. Since the probe is identical to \( m \) in Figure 6, the echo intensity should logically be 1, rather than 0, to reflect the probe’s single occurrence in memory.

Finally, it should be noted that a judicious choice of probes and traces can avoid the confounding effects of negative similarities and activations. In fact, for one set of simulations, Hintzman (1988) reported that the operative range of the activation function was roughly -0.1 to 1. This implies, however, that only subsets of probes’ and traces’ possible range of values are subject to experiment.

3.2 The inconsistent treatment of the symbolic zero

In Minerva, the feature value of zero is defined to mean that a feature is irrelevant, unknown or was forgotten. In contrast to the other two feature values, -1 and 1, this indicates that the feature value 0 is not treated as a numerical value. This symbolic rather than numerical definition of zero then should hold for all of the probing calculations.

The first probing calculation, the similarity operation, reinforces the interpretation that zero has a symbolic definition. That is, the similarity formula is constructed so that matching 0’s in a feature \( j \) of a probe and a trace have no effect on the resulting similarity value (because the zeros contribute 0 to the numerator and 0 to the denominator), whereas matching -1’s or 1’s have a zero or greater increase in the resulting similarity value (because they contribute 1 to the numerator and 1 to the denominator). Thus, the similarity formula was fashioned so that matching 0’s have a different effect on the resulting similarity value than matching -1’s or 1’s, which upholds the symbolic definition of zero.
The first violation of the symbolic definition of zero occurs during the probing calculations when the traces are scaled by their activations to define their contribution to the echo. This calculation treats the symbolic zero as a numeric zero, overlooking the fact that the result of multiplying the symbolic parameter zero with a numerical value (i.e. the activation) is undefined and not the same as multiplying the numerical definition of zero by another numerical value.

As well, further calculations violate the symbolic definition by defaulting to the numerical definition of zero instead. In particular, zero’s numerical definition is used when the echo content is “normalized” as part of the intertrace resonance algorithm. Normalizing involves finding the largest absolute value in the echo content and dividing all echo contents by this number. Feature values of zero in the traces remain zero in the traces’ contributions to the echo which then act to dilute all features of the echo to which they made a contribution, with the exception of the feature that has the largest absolute value. In all but this case, the feature value of zero behaves as the numerical value 0. For example, let us say that the values of feature j were forgotten, and are therefore zero in all traces in memory except for one which had a jth feature value of 1. Assuming also that the largest absolute value in the echo is 10, the value of feature j in the echo therefore becomes 1/10. Thus, for all of the traces for which feature j was zero valued, their zeros acted to dilute the resulting value in the echo. In other words, the zeros acted as their numerical values and pulled the result toward zero.

Thus, although Minerva begins with a symbolic definition of zero, meaning irrelevant, unknown or forgotten, by the end of the calculations it has in effect treated zero as having its numerical meaning, a feature value that is midway between -1 and 1.
3.3 Representing the real world with -1’s and 1’s

One practical issue in developing memories of experiences is that practical measurements of the real world lend themselves more naturally to number systems with greater dynamic range than simply the set \{-1, 0, 1\} that Minerva employs. In fact, because 0 represents irrelevant, unknown or forgotten information, meaningful data is actually represented with just -1’s and 1’s.

For practical applications, where a virtual agent is able to sense its virtual environment or a robot is able to sense its physical environment, it seems only natural that information perceived by the agent translates directly into features that can take on a range of values. For example, sensed features of the environment such as colour, size, and temperature could each be features of a trace. And the value of the feature as it is sensed by the agent or robot could translate directly into the feature value for the memory trace.

However, measurements of the real world do not generally lend themselves to a discrete representation such as the one that Minerva employs. Generally, it is not reasonable to categorize the size of all objects in the world as 1: large, or -1: small, to cite a very simple example. A more realistic solution that allows greater detail to be represented is to use a greater dynamic range, such as could be obtained by employing real-valued features in the range of -1 to 1 and to scale the raw measurements of the world appropriately. This approach would retain the spirit of Minerva’s representation but is much more useful in representing the features of an experience.

A second problem with confining feature values to the set \{-1, 0, 1\} is that although any discrete value can be represented using a number of these values, in a manner similar to the representation of discrete values with binary digits, not any operation on this binary-like representation is
equivalent to the operation on the corresponding discrete value. For example, ignoring the use of the feature value zero, which in Minerva does not designate an intermediate value between -1 and 1 anyway, if five digits were used to represent numbers from 0 to 31, two such numbers would differ from one another with any of the 63 values in the range of -31 to 31. But if two such representations are compared bit wise, as in Minerva, there are only 11 possible difference values that can be obtained. The 11 possible values do not map into the 63 differences in discrete representations in any consistent way. So, even if binary representations were used, the system could not use them directly to accomplish the kind of similarity measure that underlies Minerva.

3.4 Interpolating into an undefined spaced followed by re-probing

In the calculation of the echo and the process of re-probing using the echoes as probes until a stable echo is found, Minerva carries out a form of interpolation followed by an attempt to return the trace in memory that is most similar to the original probe. As Hintzman (1986, p. 416) notes, sometimes re-probing works and sometimes it does not.

“Three or four echo-probe conversions are usually sufficient to produce a virtually perfect copy of one of the category names that were originally stored. The final result is usually the correct name, but sometimes it is a perfect copy of one of the alternatives. In the latter case, the subsequent echoes drift away from the correct answer rather than toward it.”

Figure 7 illustrates a simple example in which a probe is between a cluster of memories and one outlying memory, and closest to the outlier. Given that the clustered memories are somewhat similar to the probe and the outlying memory’s similarity is strong, echoes would be likely to drift away from the outlier toward the cluster. In Figure 7, the first echo is near the probe and has been pulled toward the cluster due to the net effect of their similarities. As the subsequent echoes
Figure 7. A histogram of sample traces’ (five identical and one unique) and probes’ relative positions in space. Due to re-probing, \( p_2 \) to \( p_4 \), the consecutive echoes, \( e_1 \) to \( e_3 \), drift away from the trace that is closest to the initial probe, \( p_1 \), to settle on a more distant trace, \( e_3 \), that is part of a cluster of traces.

and therefore the probes, move away from the outlier and toward the cluster, the similarity of the outlier becomes progressively weaker allowing the echoes to drift away from it and thus the net similarity of the clustered memories becomes greater, pulling the echoes toward them. This combined weakening and strengthening of the similarities is what allows the probe to drift away from the correct answer.

Disregarding for the moment the instances in which re-probing fails, the result of the echo content calculation used in the determination of the echoes has its own problems. The echo content operation is a form of interpolation in that the sum of all traces, scaled by their respective activations, is divided by the largest absolute value in the vector sum. However, it is meaningless to interpolate into a real-valued space, which is usually the case when the defined space is \{-1, 0, 1\}. As well, although dividing the echo by its largest absolute value will scale the echo so that all values are in the range of -1 to 1, the resulting echo is not the most accurate interpolated point.
Chapter 4

A RADIAL BASIS MEMORY MODEL

The Radial Basis Memory Model of episodic memory presented here is based on Hintzman’s (1984, 1986, 1988) Minerva model of human recognition memory. Like Minerva, the RBMM is an implementable, global and multiple-trace memory model.

This is a description of the agent’s simple model of memory that is capable of learning through experience to solve simple tasks. Rather than memory being regarded simply as storage space, decision-making properties emerge from the memory model’s processes because they involve the memory of both experiences and actions. The agent is assumed to have a fixed set of possible perceptions, and a fixed set of possible actions. It is also assumed that at each point in the task environment, there is feedback available to the agent that indicates how desirable, in relation to its goals, its current situation is. With each action that the agent takes, it is able to construct a memory of the event, which consists of all three components: the perception, the action, and the reinforcement obtained from the action that enables the model to know what to do and when to do it. Each such event is stored in the agent’s memory as a separate trace. As the agent proceeds in the task environment, it is confronted with perceptual situations, and the agent must decide upon the actions it will take. In this chapter, the model’s data structures are described along with the model’s methodology.
4.1 Experiences and memory structures

Each experience is represented as an ordered vector of length \( n \). Every element of a vector is a component of the experience and is real-valued in the range of -1 to 1, inclusive. Components, that when taken together represent a higher-order aspect of the experience, are collectively referred to as a feature\(^2\) of the experience. To use an example from computer vision, colour might be a feature that is described by three components, red, green and blue, whose values are scaled into the range of -1 to 1. Of the many possible features, one that is always included in the experience is the action that followed the experience. The current experience, held in primary memory and referred to as the probe, does not yet have an associated action (and is therefore replaced with “nil” to indicate the absence of information) because the action will be determined by the results of probing memory. Past experiences are encoded to include their associated actions in secondary memory and are called memory traces or simply traces. This is illustrated in Figure 8.

In addition, every trace in secondary memory is linked to the trace that directly followed it in time. These links were not utilized in the current model but were put in place for future versions.

As well, when a trace is encoded in secondary memory, some of its components’ values may be forgotten and are replaced with “nil.” However, the procedures involved with forgotten components were not used in the current work and because they are untested, they are described in Appendix A rather than in this chapter.

\(^2\) The components correspond to Minerva’s “features” and the features referred to here are called “minivectors” in the Minerva-DM model of Dougherty, Gettys and Ogden (1999).
Figure 8. The current experience, called a *probe* when in primary memory, and the memories of past experiences, called *memory traces* when in secondary memory, are illustrated in terms of their *features* (the vertical lines dividing them are for display purposes only) which in turn are composed of varying numbers of *components*, whose values are real numbers in the range of -1 to 1, inclusive. The *probe* is missing values for its *action feature* (replaced with “nil”) because the *action* taken in the current situation has yet to be determined. There are two *actions* defined using two *components* of the *action feature* dedicated to each one.

The definition of the features

*Features* represent the characteristics of the environment that make up an experience. Their values may come from direct observations of the environment or from internally-generated values. Some *features* might always have directly observable values while others might always be estimated. Still others might be detectable in the environment in certain situations and not in other situations. In addition to *features* that characterize perceptions of the environment, *features*
may also represent internal states, such as hunger and health. Any number of features may be defined for use in the model.

Finally, to account for decision-making, there must be an action feature. Probing memory results in a decision about which action to take and once taken, the action is stored as a feature of the newly-formed trace. At this point in the development of the model, a given trace may only specify one action. The action feature has a particular format in which possibly numerous components of the action feature are dedicated to each action. Positive values in the first number of components indicate taking the first action, positive values in the second indicate taking the second action and so on for as many actions as have been defined. Only the set of components that correspond to the action taken may be positive. All others are zero-valued.

The purpose of using more than one component to represent a given action is so that when a trace is encoded in memory, and forgetting is applied to the trace, some component values of the action may be forgotten as well. As such, not only might the model not have perfect memory of a given experience, but it is possible to not have perfect memory of the corresponding action.

The relative distances within features

If three instances of a feature are equally different from one another, then the distances between each pair of them should be equal. Say, for example, the feature described John’s friends. Three instances are John’s three, very unique friends, Jane, Bill and Chris. A possible configuration of component values is Jane: (1, 0, 0), Bill: (0, 1, 0) and Chris: (0, 0, 1).

If, on the other hand, three instances are not independent and varied in some systematic way, then the distances between them ought to reflect the systematic differences between them. For
example, if a feature represented the colour of paint, and the three instances are red paint, purple paint and blue paint, then because red and blue paint combine to make purple paint, the distance between purple and the other two colours should be equal and the distance between red and blue should be greater than this. Component values could be Red: (1, 0), Blue: (0, 1) and Purple: (1, 1).

**The relative distances between features**

Consideration should be given to the relative weighting of all of the different features. The weight of a feature depends on the number of components that are allotted to it and the components’ usual values. The more components that make up a feature and the greater the variation is of the component values, the more relative weight a feature has. And the more weight a feature has, the larger is the role it plays during probing. Therefore, it is important to balance the weight of the various features in a way that seems appropriate for the task. This is also an issue in Minerva, one that Hintzman does not address.

### 4.2 A trace’s utility and characteristic radius parameters

Every trace in secondary memory has associated with it a utility parameter and a characteristic radius parameter. Thus, trace $i$ has a utility, $u_i$, and a characteristic radius, $\sigma_i$. A description of the characteristic radius is given first.

If a trace is viewed as being a point in $n$-dimensional space, such that the trace’s vector components also describe its coordinates in space, then the trace’s characteristic radius is a measure of a “sphere” of influence in a volume of space centered on and surrounding the trace. All points that exist inside the sphere of influence are considered to be similar to the trace, their
degrees of similarity depending on their distances from the center. The center represents a perfect match, and at large distances relative to the characteristic radius, points are considered not similar.

Just as the characteristic radius defines the extent of the trace’s sphere of influence, the utility defines the strength of that influence. The utility is a scalar value that ranges from 0 to 1, and reflects how successful that trace has been when applied in the past. For example, a value of 0, no strength, reflects the fact that this trace did not lead to successful outcomes in the past. On the other hand, a value of 1, full strength, implies that this trace led to successful outcomes in the past. When a trace is first encoded in memory, its utility is assigned a mid-range value of 0.5 because the outcome of applying the trace’s action is not yet known to be desirable or undesirable.

Thus, encoded with each trace is its own utility and characteristic radius. Different traces may have different values for these parameters. An example of this is given in the top graph of Figure 9 on page 40. The updating of these parameters’ values, because they vary with experience, and the effect that these parameters have on the choice of action are discussed in the sections that follow.

4.3 The model’s methodology

The model encompasses five procedures. At the onset of an experience in primary memory, the procedures are executed in this order: (1) updating utility values, (2) updating characteristic radius values, (3) probing memory, (4) recommending an action, and (5) encoding the experience
into secondary memory. Procedures (3) and (5) are based on Minerva with some modifications, and procedures (1), (2), and (4) are new.

The exact calls to these procedures are outlined in the pseudo code in Figure 10 on page 53. A detailed explanation of these procedures follows.

Because probing memory is naturally thought of as occurring when a representation of the current experience is formed in primary memory, for ease in understanding, this procedure is outlined first, followed by recommending an action and encoding an experience into secondary memory, and finally, updating utility values and updating characteristic radius values.

4.3.1 Procedure: Probing memory

Secondary memory is probed with each current experience as it occurs in primary memory. Very generally, the process involves comparing the probe to all traces in memory, which responds with a collective echo that is an interpolation of all past experiences that are found to be similar to the probe and that resulted in favorable experiences. The amplitude of the collective response is given by the echo intensity and can be viewed as an indication of the probe’s overall familiarity based on past experiences. Given the component values of the action feature of the echo, another procedure (discussed in section 4.3.2) recommends an action. Whether or not the results of probing or the model’s recommendation of an action are of immediate interest, byproducts of the probing calculations are necessary in order for reinforcement learning adjustments (discussed in sections 4.3.4 and 4.3.5) to be made during the subsequent experience. Therefore, if the model is to learn from its experiences then memory must be probed with every current experience.
Given a *probe* that contains information about the current experience and the collection of *memory traces* that make up *secondary memory*, the probing process involves the calculations that follow.

The calculation of each trace’s *distance* to the *probe* is the first step in determining each trace’s similarity to the *probe*. The *distance* function is calculated according to the Euclidean distance, $D$, between the *probe* vector, $\vec{p}$ and a trace, $\vec{t}$, excluding their *action features*.

$$ D = \| (\vec{p} - \vec{t}) \| $$

Although the term “distance” is used to aid in understanding, in practice, the square root in $D$ is not calculated since only $D^2$ is of interest, as an inspection of the following Gaussian radial basis function demonstrates. This shortcut saves execution time and preserves precision.

The extent to which a trace contributes to the *echo* depends on its *activation*, which is calculated using a Gaussian type of radial basis function. That is, the *activation*, $A_i$, of a trace $i$ is given by,

$$ A_i = u_i \cdot e^{-\frac{(D_i)^2}{(\sigma_i)^2}} $$

where $u_i$ is the trace’s *utility*, an indication of how successful the trace has been when applied in past situations (and the strength of the trace’s sphere of influence), $D_i$ is the Euclidean distance (as given above) between the *probe* and the trace, and $\sigma_i$ is the trace’s *characteristic radius*, a non-zero positive value that defines the width and therefore (because it does not define these independently) the steepness of the Gaussian function (and thus the extent of the trace’s sphere of
influence). The result of the activation function is truncated to zero when its value is less than $10^{-4}$. The truncation ensures that traces that have insignificant effects are not carried further in the calculations. This occurs when a trace’s distance from the probe is much larger than its characteristic radius. A zero activation value also occurs when the trace’s utility is zero. The upper bound is 1, and occurs when the utility is 1 and the distance is zero because the probe and trace are then identical.

As previously noted, a trace’s utility, which is a value ranging from 0 to 1, reflects how successful the trace has been when applied in the past. The purpose of including the utility parameter in the activation function is to scale the activation so that traces that led to successful outcomes in the past receive more activation and therefore contribute more toward the current echo. The method by which the utility is updated to reflect success and failure is discussed in detail in Section 4.3.4.

In addition to emphasizing the contribution of successful traces toward the echo, traces that are similar to the probe should influence the overall output of memory more than traces that have very little in common with the probe. The Gaussian function scales the activation received by a trace according to the ratio of its distance to the characteristic radius. For example, with a small characteristic radius, only traces very close to the probe receive appreciable activation and the amount of activation decays rapidly as distance increases. Conversely, with a large characteristic radius, the curve is wider and more gradual, and traces further away receive appreciable activation.
Figure 9. Four sample traces and a probe are viewed as points in 2-dimensional space such that their two components are their coordinates in that space. These points are plotted in the top graph which also illustrates the area of influence that surrounds each trace. The area of influence is defined by the trace’s characteristic radius. The gradation, which is to say the way in which a trace’s area of influence fades away, is according to a Gaussian function. The amplitude of a trace’s activation, which has been scaled by its utility, is given by the darkness of the trace’s area of influence. The probe does not have an area of influence but is simply a point that exists in the space and may or may not intersect traces’ areas of influence. The echo intensity is the sum over all activations, shown individually in the bottom graph.
Thus, the activation that a trace receives depends on its similarity to the probe as defined by the trace’s characteristic radius in the Gaussian function, as well as the trace’s success rate when applied to situations in the past which is given by the trace’s utility.

As in Minerva, secondary memory responds to a probe with the echo intensity, \( I \), such that,

\[
I = \sum_{i=1}^{m} A_i
\]

where \( A_i \) is the activation of trace \( i \). Unlike the situation in Minerva where the intensity can range from \(-m\) to \(m\), here intensity ranges from 0 to \(m\), where \( m \) is the number of traces in memory. As well, because the characteristic radius and the utility are included in the activation operation, the echo intensity reflects the familiarity of similar, successful experiences in memory. Therefore, a large intensity value indicates that similar experiences with desirable outcomes have been experienced many times before. A small intensity value indicates that the probe is a relatively novel experience in that few traces in memory match, or that the success rate of traces that do match is very low, or a combination of both. (If the intensity were required to reflect familiarity without bias for success rates, all traces’ utilities would have to assume a value of 1.)

Also in response to a probe, an echo is collectively formed from the contents of secondary memory. The echo vector has the same form as a trace and is a reflection of all memory traces that are similar to the probe and that resulted in favorable outcomes. (Again, if the echo were required to reflect similar traces in memory without bias for success rates, all traces’ utilities
would have to assume a value of 1.} The value of each component of the echo is an interpolation performed by the echo content function. That is, the echo content, $C_j$ of component $j$ is,

$$C_j = \frac{\sum_{i=1}^{m} A_i t_{i,j}}{I}$$

where $m$ is the number of traces in secondary memory, $A_i$ is the activation of trace $i$, $t_{i,j}$ is the $j$th component of trace $i$, and $I$ is the echo intensity.

In essence, each trace contributes itself, scaled by its activation, to the echo. The contributions are summed and each component of the resulting vector is divided by the total amount of activation of the traces that actively contributed to that component. A trace that does not actively contribute to a component of the echo has an activation value of 0 and has no effect on the resulting value of the component in the echo. For example, if a given trace’s $j$th component has the value 0.2 and the trace’s activation is 0.8, then given the activation value, the trace is clearly a strong match to the probe and has been quite successful when applied in the past. Thus, the trace makes a relevant contribution to the echo and so in the calculation of $C_j$, the trace contributes 0.16 to the numerator’s sum and 0.8 to the denominator, $I$. In contrast, a second trace that has a $j$th component value of 0.9 and the trace’s activation is 0, indicating that the trace is either dissimilar or was very unsuccessful in past applications, does not actively contribute to component $j$ of the echo. Specifically, it contributes 0 to the numerator’s sum and 0 to $I$. 

42
4.3.2 Procedure: Recommending an action

Based on the *echo contents* of the *action feature*, the model is able to recommend an *action* to take in the current situation.

As noted, in the format of the *action feature*, the first number of *components* are dedicated to an *action* A, the next to an *action* B, and so on such that all *components* of the *feature* are accounted for by an *action*. In the simplest case, every distinct *action* is represented by a single *component*.

The first step is to calculate the sum of all contributions toward each defined *action*. So for a particular *action*, the *echo contents* of all of the *components* assigned to the *action* are summed.

For example, let us say that an *action feature* consists of 16 *components* divided among four *actions*, A, B, C and D. The *echo contents* of the first four *components* are summed to give the total contribution toward carrying out *action* A, the next four toward *action* B, and so on. The result of carrying out this operation is that a single value corresponds to each of the possible *actions*. For a given *action*, its value reflects the extent to which previous situations have been successfully dealt with by taking that *action*.

At this point, there are a number of ways of deciding which *action* to recommend. One possibility is recommending the *action* with the largest corresponding value. Another, which is the decision method that was implemented, is a Monte Carlo method, in which all of the *action* values are turned into relative probabilities such that they sum to 1. Conceptually, the probabilities representing each *action* are then laid out as adjacent and contiguous ranges in the interval 0 to 1. A random number is generated in this range of 0 to 1, so that its value
automatically intersects the probability of one of the actions. This is then the action that the model recommends.

4.3.3 Procedure: Encoding an experience into secondary memory

The action feature of the probe is assigned the action that was taken following probing and the resulting vector is removed from primary memory and encoded into secondary memory as trace \( t_{m+1} \).

In addition, the newly encoded trace is assigned a utility value of 0.5, a compromise midway in the allowed range of 0 to 1, because at this point it is not yet known whether the outcome of the action just taken is desirable or undesirable. That is,

\[ u_{m+1} = 0.5 \]

The trace is also assigned a characteristic radius. If it is simply known that the newly-encoded trace describes a totally novel experience or if the trace had an echo intensity of zero (which indicates no familiarity and therefore a novel experience) when it was used as a probe, then the trace’s characteristic radius, \( \sigma_{m+1} \), is assigned the value of the free parameter, DefaultSigma, as follows,

\[ \sigma_{m+1} = \text{DefaultSigma} \]

If, on the other hand, the newly encoded trace had a non-zero echo intensity value then there is at least one trace already in memory that was considered similar to the current trace. Thus, the current trace is not viewed as a totally new experience and is assigned a value that reflects the
current state of the model’s ability to discriminate between situations that are similar to the newly-encoded trace. Accordingly, its characteristic radius is calculated as the weighted average of all traces’ characteristic radii. That is, the newly encoded trace has the characteristic radius, \( \sigma_{m+1} \), such that,

\[
\sigma_{m+1} = \frac{\sum_{i=1}^{m} A_i \sigma_i}{I}
\]

where \( m \) is the number of traces in memory, not including the newly-encoded trace, \( a_i \) and \( \sigma_i \) are the activation and characteristic radius values, respectively, of trace \( i \) in memory, and \( I \) is the echo intensity.

Lastly, the number of traces in secondary memory is incremented to reflect the new addition.

### 4.3.4 Procedure: Updating utility values

Thus far, each trace has been described as having its own utility value. As well, a trace’s utility appears in the calculation of its activation, where the role of the utility is to scale the amplitude of the Gaussian function according to how successful the trace has proved to be in the past.

So that utilities continue to accurately reflect their traces’ success rates, utilities are updated following every encoding of a trace into memory, when the next state of the world is perceived and feedback is available about whether the most recent action was successful or not.

Specifically, the utilities of the traces that recommended the action that was actually taken are updated according to a form of positive reinforcement if the action proved to be successful or positive punishment if unsuccessful.
If a given trace recommends the action that is taken and that results in a desirable situation, then that trace’s utility receives positive reinforcement according to the formula,

\[ u_i = u_i (1 + A_i \cdot UtilityScalar) \]

where \( u_i \) is the utility and \( A_i \) is the activation of a trace \( i \) that resulted in the correct recommendation, and \( UtilityScalar \) is a free parameter that is a constant, and whose value is between 0 (no updating) and 1. However, if the new \( u_i \) exceeds 1.0, then it is set to 1.0.

Conversely, if a given trace recommends the action that is taken and that results in an undesirable situation, then that trace’s utility receives positive punishment according to the formula,

\[ u_i = u_i (1 - A_i \cdot UtilityScalar) \]

If, however, the new \( u_i \) is less than 0.0, then it is set to 0.0.

So for example, if the most recent action was to look into a nearby bush and this led to the desirable experience of finding food, then the traces that suggested looking into the bush have their utilities incremented. All other traces’ utilities are not updated.

On the other hand, if for instance the most recent action was to look into a nearby bush and this led to the undesirable state of being stung by a bee, then the traces that suggested looking into the bush have their utilities decremented. All other traces’ utilities are not updated.
4.3.5 Procedure: Updating characteristic radius values

So far, each trace has been described as having its own characteristic radius, which appears in its activation function and determines the size of the trace’s sphere of influence.

The goodness, or otherwise, of the state of the world that results from an action being taken in the previous situation is a source of feedback to update the characteristic radii of the traces that recommended the action that was taken. If the feedback is positive, that is the action resulted in a desirable next state, then all characteristic radii remain unchanged. However, if the feedback is negative, meaning the action resulted in an undesirable situation, then the characteristic radii of the traces that recommended the action are decremented in the following fashion:

\[ \sigma_i = \sigma_i (1 - \text{SigmaScalar}) \]

where \( \sigma_i \) is the characteristic radius of trace \( i \) and \( \text{SigmaScalar} \) is a free parameter whose value is constant and ranges from 0 (no updating) to less than 1.

The effect of narrowing the characteristic radius of a trace that contributed to an unsuccessful outcome is to reduce its sphere of influence so that next time the same probe occurs, it will activate the trace less and therefore the trace’s contribution to the echo will be weaker. This reduces the likelihood of repeating the same action in the same situation.
Chapter 5

THE MODEL APPLIED TO A MAZE TRAVERSAL TASK

5.1 The maze traversal task

The Radial Basis Memory Model can be used to develop a maze traversal agent that can carry out a maze navigation task, and provide data that can be directly compared to human performance data. The maze traversal task is a good test for the RBMM in that it provides a perception-action cycle which coincides with the intent of the model to capture the role of memory in natural behavior. Browse and Gray (2006) experimented with human performance for learning virtual mazes through successive attempts, and concluded that the role of memory in the maze task does not appear to involve memorization. Rather, it appears to involve the type of natural, unintentional memory of events for which the model is intended. Thus, the memory model was applied to the task of traversing the mazes used in Browse and Gray’s (2006) experiment in order to determine if it can emulate the human performance by exhibiting the same average error rate for successive attempts at the mazes as the human subjects displayed.

5.2 Human data from Browse and Gray’s Experiment B

The navigational error results of the maze component of Browse and Gray’s (2006) Experiment B (reviewed in Section 2.3) indicate that all subjects improved over the course of the attempts and there was no significant difference in the mean navigational error between the Fixed Position Perspective and Head-Tracker Perspective conditions.
In the interest of having as much human data as possible and because there was no significant difference found between the two conditions, the data from the Fixed Position Perspective and Head-Tracker Perspective conditions was compiled into one. The memory model’s performance is compared to this set of human data.

5.3 The mazes

In the human experiment conducted by Browse and Gray (2006) and in the current experiment, subjects and our maze traversal agent completed six attempts at finding the goal for each maze traversed. In the human experiment, 10 subjects completed four mazes each. Twelve maze patterns (illustrated in Figure 3 and Figure 4) were used. In the current experiment, a single maze traversal agent completed these same twelve mazes upon which the human data are based. The mazes were used with different frequencies but proportional to the frequencies with which they were used in the human experiments.

All of the mazes referred to above have the following characteristics. All mazes contain a start position and a goal. The goal is six left/right decisions away from the start position. All left/right decisions are made at identical T-intersections in which one direction leads to a dead end and the other leads to another T-intersection, except for the final T-intersection which leads to a dead end and the goal. An equal number of left turns and right turns are required to reach the goal in every maze.

Although the mazes were schematically identical, the human experience and the agent’s experience in traversing these mazes differed in how distance and time were handled. The mazes in the present experiment were not identical distance-wise to the ones used in the human
experiment, but were identical in terms of the left/right decisions to the goal. For example, in the human studies, from the moment when a subject makes a correct decision at an intersection, time passes as the subject then traverses the intervening distance to the next intersection. For the agent, time is not a continuous entity but rather occurs in discrete increments. So if an agent is at an intersection at time $t$ and makes the correct choice of action, the next time of awareness occurs at time $t+1$ at which point the agent is at the next intersection.

5.4 The maze traversal agent and its memory

Because the RBMM does not have any sensors or actuators that allow it to interact directly with the maze environment, a “maze traversal agent” is employed that is outfitted with the RBMM as its memory. In the course of traversing a maze, the maze traversal agent determines whether its current situation is good or bad and passes this information to its RBMM to be used as feedback in the RBMM’s reinforcement procedures. The agent also passes its perceptions and internal estimations of its environment to its RBMM’s primary memory, which initiates the RBMM’s probing procedure, among others. The agent carries out the action recommended by its RBMM unless one of the agent’s built-in maze traversal rules overrides the choice of action. These simple built-in rules were put in place to carry out maze-related behaviours and do not include any specific maze traversal strategy that would enable the agent to solve the mazes on its own, that is, without its RBMM memory. Thus, the agent has no problem-solving algorithms of any kind other than the influence of its RBMM’s retrieved memory content. The performance of the agent, with the RBMM as its memory, will be compared to human performance for the same task and for the same mazes.
5.4.1 The maze traversal agent’s built-in rules

Because the performance of only the RBMM is of interest, seeing whether it matches human decision making for this task, other factors that do not rely on memory are removed by building them into the behaviour of the agent. These built-in rules are:

“When arriving at an intersection for the first time during the first attempt at a maze, randomly decide whether to turn left or right based on equal probabilities.”

The first attempt at a maze is considered different from subsequent attempts because the model has no prior knowledge of the maze upon which to base decisions. Therefore, the RBMM plays no role in this series of first decisions. Rather the agent decides with 0.5:0.5 probability whether to turn left:right.

“When arriving at an intersection from another intersection, not during the first attempt, consult memory for the decision to turn left or right.”

The available actions are to turn left or right and this decision is to be made by the RBMM. Backtracking is not an option because it was not an option in the human experiment either. In the human experiments, the path that a subject took through a maze during a particular attempt was rendered in real time as a dashed line on the floor. Therefore, subjects were able to see where they had just come from and thus could avoid wasteful backtracking. They were also able to avoid taking a wrong turn more than once at any intersection during a given attempt at the maze.

Similarly, the action of doing nothing is not an option for the agent because the human subjects were instructed to traverse a maze until arriving at the goal.
“At dead ends, backtrack.”

The only viable option at dead ends is to backtrack.

“Having returned to an intersection after hitting a dead end, turn in the opposite direction than the one taken that lead to the dead end.”

Because of the dotted line showing the human subjects where they had just been, the only path not yet taken at intersections to which they returned was obvious.

“At the goal, do nothing.”

In the human experiment, there was nothing further that subjects were required to do once they reached the goal other than wait until the next trial began.

5.4.2 Pseudo code for the agent’s and the RBMM’s method of maze traversal

A top level description of how the agent operates, incorporating its built-in rules, along with its RBMM’s procedures is given in the pseudo code in Figure 10. The regular text outlines directions for the maze traversal agent and the italic text outlines directions to be carried out by the model. The pseudo code specifies all 6 attempts at a maze.

During the first attempt at a maze, all of the subjects’ experiences are novel, therefore, the DefaultSigma value is used when the RBMM encodes each of its experiences in memory during the first attempt at a maze.
**Pseudo code**

$LTM = empty$

For attempts 1 to 6, do

Loop.

Perceive the current state of the world from external cues and internal cues.
Pass these perceptions to the RBMM’s primary memory in the form of a probe.
If perceive that at

the goal, then

This is a desirable situation.
*Update utility values and sigma values.*
*Probe with the current experience.*
Do nothing.
*Encode this experience in memory.*
Exit loop.

a dead end, then

This is an undesirable situation.
*Update utility values and sigma values.*
*Probe with the current experience.*
Execute the action: Backtrack.
*Encode this experience in memory.*

an intersection, then

If returned to current intersection from a dead end, then

This is a desirable situation.
*Update utility values and sigma values.*
*Probe with the current experience.*
Turn in the opposite direction than the one taken that lead to the dead end.
*Encode this experience in memory.*

Else if first attempt at the current maze, then

This is a desirable situation.
*Update utility values and sigma values.*
*Probe with the current experience.*
50:50 chance of turning left:right.
Generate random number to decide and turn accordingly.
*Encode this experience in memory.*

Otherwise, (arriving at this intersection for the first time during this attempt and this is not the first attempt at the current maze) then

This is a desirable situation.
*Update utility values and sigma values.*
*Probe with the current experience.*
*Recommend an action given the available options: left or right*
Take the recommended action.
*Encode this experience in memory.*

**Figure 10.** The pseudo code that specifies the agent’s behaviour (normal text) and the RBMM’s behaviour (italic text) for all six attempts at a maze.
5.5 The RBMM’s parameter values

During the process of testing the model to see how well it emulates the human data, some of the model’s parameter values remained constant and others were varied with each attempt at modeling the human data.

5.5.1 The constants

The parameters that remained constant throughout the RBMM’s modeling phase include the learning rate and the features of an experience.

The learning rate, $L$, or conversely the forgetting rate, $1 - L$, was constant throughout modeling. In fact, forgetting was not used, and therefore, $L = 1$ at all times.

Once defined, the features were also constant. As noted, the features that collectively make up an experience may be perceptions of the local environment (sensed or estimated) and internal states. The features used for the maze traversal task include percept, depth, dead reckoning and the requisite action feature. In some modeling attempts, dead reckoning was not used. Figure 11 lists all possible instances of these features in terms of their component values.

The only feature that is always directly detectible from the model’s immediate environment is the percept feature. In the mazes, there are three possible percepts: being at an Intersection (all of which are identical), being at a Dead end (all of which are identical), and being at the Goal. The actual traversal of hallways is removed for simplicity so that an action taken at a given percept results in the model’s immediate arrival at another percept. Thus, the perception of being in a hallway is not included.
Figure 11. From left to right, the four features used for the maze traversal task, the instances of each of the four features, and a meaningful description of what the instances represent. The component values and the number of components dedicated to each feature were chosen by consideration of the relative weighting between and within features.

Figure 12. The percepts at every location where an experience can occur in an example maze.
Figure 13. The depths 0 to 6 at every location where an experience can occur in an example maze.

*Depth* is included as a feature under the assumption that although there are no immediate cues, humans have a sense of how far into the maze they are and that keeping track of this helps them to know what actions are required when in the maze. Thus, the *depth* feature is the model’s estimation of how many steps into the maze it is. Believing that it is at the first intersection of the maze corresponds to a *depth* of 0. Believing that it is at the next step past the first intersection, whether it is a dead end or another intersection corresponds to *depth* 1. So more generally, with each additional step that the model takes that leads it deeper into the maze, *depth* increments. Conversely, with each step backward that the model makes, *depth* decrements. As mentioned, encoding errors in estimation, such as believing that it is four steps into the maze when it is three steps, is not modeled explicitly.
Figure 14. Dead reckoning at every location where an experience can occur in an example maze.

Dead reckoning is similarly included as a feature under the assumption that although there are no immediate cues, trying to keep track of the direction that one faces is a helpful strategy sometimes employed in way-finding. Therefore, dead reckoning is the model’s estimation of the direction it faces. The only possibilities in the maze are the four cardinal directions: North, South, West and East. Although there are no detectible cues to indicate North or any other direction anywhere in the maze, for reference purposes, the model assumes that when it begins a maze at the first intersection it faces North. Explicit errors in the estimation of dead reckoning, such as believing it is facing North when it is actually facing West, are not modeled.

The action feature, as previously mentioned, consists of four actions: Turn left, Turn right, Backtrack and Do nothing. These are the only four actions required in the maze and are the
Figure 15. This instance of an action feature represents turning right. Hence, all four of the components assigned to a right turn are 1’s and all of the other components are 0’s.

possible actions that the model recommends when an action is requested. An example of an action feature used during the testing phase of the model is depicted in Figure 15. Four components were assigned to each of the four actions. Therefore, the action feature in Figure 15 represents a right turn because the four components assigned to turning right are 1 and all of the other components are zero.

The relative distances within features

Features have been defined with their instances’ relative distances in mind.

The percept sub-vectors were designed so that the Euclidean distances between Intersection, Goal and Dead end are equal. Specifically, each is \( \sqrt{12} \) away from either of the others. This is because each percept is considered to be a concept independent of the others.

For the depth feature, each depth is \( \sqrt{4} \) away from an adjacent depth, and each successive depth is increasingly distant from the first. So for instance, depth 3 is \( \sqrt{4} \) away from depths 2 and 4, \( \sqrt{8} \) away from both depths 1 and 5, and is \( \sqrt{12} \) away from Depth 6. The maximum distance between
depths is $\sqrt{20}$ occurring between depth 0 and depth 6. The distances between the various depths were defined such that adjacent depths are separated by less distance than those that are not adjacent. This follows from considering the concept of being 2 steps into the maze to be more similar to being 1 or 3 steps into the maze than to being 5 steps into the maze. Without building in any explicit error in the estimation of depth, defining the distances between the depths in this way allows a greater chance of confusion between adjacent depths than non-adjacent ones.

Similar to depth, dead reckoning is designed such that adjacent directions are less distant than opposite directions. Specifically, adjacent directions are $\sqrt{4}$ away from each other and opposite directions are separated by $\sqrt{8}$.

The actions are equally distant, $\sqrt{8}$, from each other due to the structure of the action feature, in which four components are dedicated to each action.

The relative distances between features

Consideration was also given to the relative weighting between features. The more weight a feature has, the larger is its role during probing. It is therefore important that the features’ weights balance in a way that seems appropriate.

The consideration that went into defining the maze traversal features was that differences in percept should weigh more than small changes in depth and dead reckoning, and that small changes in depth and dead reckoning should be weighted about equally. That is, if the percept of a trace does not match the probe’s percept, this difference should outweigh the fact that the trace’s depth and dead reckoning match the probe’s.
5.5.2 The free parameters

Three free parameters were varied between modeling attempts in order to produce the different results. They are three of the RBMM’s global parameters: DefaultSigma, SigmaScalar and UtilityScalar. As noted, DefaultSigma is every novel trace’s initial characteristic radius value. During the first attempt at a maze, all of the subjects’ experiences are novel, therefore, the DefaultSigma value was used when the RBMM encodes each of its experiences in memory during the first attempt at a maze. SigmaScalar is the amount by which a trace’s characteristic radius is scaled down when the trace recommends an action that is taken and which turns out to be unsuccessful. UtilityScalar affects the amount by which a trace’s utility value is scaled up or down according to whether the trace recommends an action that turns out to be successful or unsuccessful respectively.

5.6 Results

The average human performance from Browse and Gray’s (2006) experiment and our model’s fit to that performance are shown in Figure 16. The average human performance reflects the total of 40 mazes that the 10 subjects completed. The model’s average performance is on the same 40 mazes, repeated 4 times. The model’s experiences consisted of the features percept and depth, and its free parameters were set at: DefaultSigma = 2.2, SigmaScalar = 0.3 and UtilityScalar = 0.4.

However, the average human performance is not representative of any individual subject’s average performance, which is given in Figure 17.
Figure 16. The average human performance, and the RBMM’s average performance using the features: percept and depth, and the free parameter values: $Default\Sigma = 2.2$, $SigmaScalar = 0.3$, and $UtilityScalar = 0.4$. The error bars represent the standard deviation of the subjects’ average performance (black bars) and the model’s average performance (grey bars).

In the second test, based on their error on the third attempt, subjects were divided into two groups: fast maze learners, and slow maze learners. Subjects 6 to 9 were considered fast maze learners because they all had less than 15% error by attempt 3. Subjects 1 to 5 and 10 were considered slow maze learners because they all had greater than 15% error by the third attempt.
Figure 17. Each subject’s average performance on 4 mazes, produced from the experimental data from Browse and Gray (2006).

It is apparent from Figure 18 that the average performance of one group is generally outside the standard deviation that describes the performance of the other group.

In modeling the fast maze learners, it was found that using the percept and depth features only was not adequate. It was hypothesized that perhaps the fast maze learners performed better than average because they used more complex strategies to traverse the mazes.
**Figure 18.** Fast versus slow maze learners’ average performance. The error bars represent the standard deviation of the fast learners’ average performance (black bars) and the slow learners’ average performance (grey bars).

When the *dead reckoning feature* was added, a decent match in performance to the fast maze learners was possible using $DefaultSigma = 2.4$, $SigmaScalar = 0.7$ and $UtilityScalar = 0.7$. The two most difficult points to model for this group were the errors at attempts 2 and 5. The error for the RBMM on attempt 2 tended to drop lower than that of the humans. Also, the average of the fast maze learners tended to perform extremely well on the fifth attempt, with low error and small standard deviation. As can be seen in Figure 19, the model’s performance by contrast was
Figure 19. The average of the fast maze learners’ performance and the RBMM’s average performance using the features: percept, depth and dead reckoning, and the free parameter values: DefaultSigma = 2.4, SigmaScalar = 0.7, and UtilityScalar = 0.7.

fairly linear from attempts 3 to 6, and did not improve as much on attempt 5 as the fast maze learners.

The performance of the slow maze learners was modeled, as illustrated in Figure 20, without difficulty using just the percept and depth features. The free parameter values of DefaultSigma = 2.6, SigmaScalar = 0.25 and UtilityScalar = 0.4, produced these results. Only on attempt 6 does the human performance improve while the RBMM’s performance remains more in line with that of attempts 4 and 5.
Figure 20. The average of the slow maze learners’ performance and the RBMM’s average performance using the features: percept and depth, and the free parameter values: DefaultSigma = 2.6, SigmaScalar = 0.25, and UtilityScalar = 0.4.
6.1 Conclusion

The motivation for the current work was to demonstrate that a model of human memory could support the completion of tasks that involve using memory in a natural, unintentional way. This follows from the notion that human episodic memory operates largely without the intervention of intentions to memorize and recall information. Thus, the goal of the current work was to develop a model of human memory that is capable of carrying out a simple maze traversal task such that its performance models human performance for the same task. The Radial Basis Memory Model that was developed did in fact exhibit performance very similar to that of humans for this maze traversal task.

The Radial Basis Memory Model’s design began with an existing model of human recognition memory, that is, Hintzman’s (1984, 1986, 1988) Minerva. This model was appealing as a starting point because it is an implementable, multiple-trace and global memory model. The most notable modification was the use of a Gaussian radial basis function in the calculation of the activation. This particular modification meant that two new parameters that vary with experience were defined for each memory trace. A trace’s utility is an indication of the success to which the trace was applied in past situations and the characteristic radius defines the area or volume surrounding the trace in which a given point is considered similar to some degree. The RBMM incorporates a mechanism that allows it to recommend an action given its perception of the
current state of the world and includes a reinforcement learning mechanism that incorporates feedback about the success of its actions so that the model can learn from its experiences.

In the maze traversal task used in an experiment by Browse and Gray (2006), subjects were required to navigate through mazes six times from a start location to a goal location. The mazes contained no landmarks and involved six decision points at identical T-intersections. Taking the wrong direction at a decision point led directly to a dead end and taking the correct direction led to the next intersection, or in the case of the last intersection, to the goal. Although individual subjects’ performance was quite varied, all subjects’ performance improved with successive attempts at the mazes. Because the human data for this maze traversal task already existed, the RBMM was applied to the same maze navigation task. However, because time occurs in discrete increments for the RBMM, it did not experience traversing hallways. Instead, the perception of being at an intersection and taking an action was followed directly by the outcome of the decision (such as being at another intersection) in the next increment of time.

Applying the model to the maze traversal task began with defining the model’s constant parameter values. The first of these is the learning rate which was set so that no forgetting occurred and all experiences were encoded into memory without any loss of information. Secondly, the features that make up an experience in the mazes were decided upon and their instances were defined. For example, percept was a feature of the experience and three instances of percept were defined: at an intersection, at a dead end and at the goal. Once these parameter values were set, they were not altered during the modeling process.
Modeling the human data required the manipulation of the model’s three free parameters: 

*DefaultSigma*, *SigmaScalar* and *UtilityScalar*. The first of these, *DefaultSigma*, defines a trace’s initial *characteristic radius* whenever the trace is considered to be a novel experience. The other two parameters are involved in the reinforcement learning mechanism. *SigmaScalar* scales the rate at which a trace’s *characteristic radius* decreases when the trace contributed to a decision to which the trace does not apply. And lastly, *UtilityScalar* scales the rate at which a trace learns the overall success of its *action* with each application of its *action* in later situations.

A maze traversal agent was created that was capable of sensing the maze environment and acting upon it. The agent had a few built-in rules that defined its abilities, but none of these rules enabled any strategies that would allow it to solve the mazes on its own. Rather, the agent was outfitted with the RBMM as its memory such that all relevant decisions in the maze came from the RBMM.

The agent was applied to the maze traversal task and its performance compared to the error rate from the human data. Values for the three free parameters used in the RBMM were found so that the RBMM completed the task in such a way that it modeled the average human data remarkably well. When the human data was grouped, based on error rate at the third attempt at the mazes, into fast maze learners and slow maze learners, the RBMM was able to model the average group performances as well. Thus, the RBMM is able to model individual differences in performances.

Because no forgetting occurred when the model encoded its experiences in memory, the model’s performance further suggests that error on the maze traversal task can be the sole result of interference by memories that are not applicable in the given situation. The amount of
interference depends on the traces’ characteristic radii. That is, traces with inappropriately large characteristic radii interfere in more decisions than traces with smaller characteristic radii.

More generally, the RBMM was built upon the Minerva memory model, which was designed with the intention of demonstrating that both episodic memory and memory for abstract concepts could be handled by a single model. Testing has not yet been done to determine how well the RBMM performs compared to Minerva in this respect. However, through its demonstrated success with the current maze traversal task, it has been shown to implement at least a simple stimulus-response form of procedural memory as well.

This model is expected to be reasonably scalable because many of the factors are limited only by the storage capacity and more importantly the processing power of the computer on which the model is run. These factors include the number of features and components that are required to make up an experience, the number of actions that are defined in the environment, and the number of memories stored in secondary memory. Thus, the RBMM is very flexible in terms of the variety and number of experiences that it allows. The model is also scalable in its application to tasks other than maze traversal. The current version of the model, however, is limited to carrying out a single task. In order to scale the model to handle a number of tasks concurrently, an additional mechanism would be required to coordinate a number of pairings of utility and characteristic radius per trace, a pair for each task.
6.2 Future directions

As the RBMM is in its initial stages of development, there is much testing still to be done and there are many suggested modifications and extensions that can be implemented to increase the model’s capabilities.

In terms of testing that ought to be done, two sets of tests take priority. Firstly, the model should be tested against Minerva’s performance and Hintzman’s human data. Secondly, it should be established through tests whether the model suffers from the same problems as Minerva in its inability to account for certain memory phenomena (as described in Section 2.1.3). Apart from these necessary tests, additional experiments in the following areas would be interesting:

1. Conduct more in depth testing to see whether the model committed the same proportions of errors at the same intersections in the maze as the humans did, and to see if it is possible to model a given subject’s performance. This may involve changing the features that were used, redefining their instances and possibly defining new features.

2. In the current testing, only the integer values of -1, 0, and 1 were used. However, the model is implemented to handle real-valued components in the range of -1 to 1, inclusive. Therefore, test the model with tasks that require real-valued components. Or, to take this a step further, rather than specifying the values of a feature’s set of instances as was done in the current work, test the model in an environment where the feature values are gathered directly from the agent sensing its environment.

As mentioned, the current model is a first version and there is much that can be done by way of modifications and additions. The following enumerates a few suggested improvements:
i. In the current work, a _trace’s characteristic radius_ value decreases when feedback indicates that the _trace_ recommended an _action_ that turned out to be unsuccessful. This has the effect of specializing, or being able to distinguish better between it and dissimilar situations. Conversely, it would be interesting to increase the _characteristic radius_ somehow to achieve the ability to generalize across similar situations.

ii. Make use of the time links between memories to pass _activation_ forward, much like semantic networks with spreading activation, so that there is a temporal, predictive mechanism.

iii. The _features_ that are made up of many _components_ influence the outcome of probing more than _features_ that consist of only a few _components_. Once these _features_ have been defined, their relative weighting does not change. However, the relative weighting of _features_ should be flexible and depend on the agent’s goals. As an example, an agent is in the situation that it has before it an apple and an umbrella. If its motivation is hunger, then it should act to pick up the apple. If its motivation is to find shelter, then it should pick up the umbrella. Thus, one context (hunger) of a particular situation (the presence of an apple and an umbrella) dictates that certain _features_ (presence of an apple) are relevant, however, the same situation in another context (exposed to the elements) requires that different _features_ (presence of an umbrella) are key. Therefore, the relative weighting of a given _trace’s features_ ought to be flexible in order to be useful in a variety of conditions.

iv. Forgetting was implemented in the coding of the RBMM using the same _learning rate_ parameter, _L_, as in Minerva but rather than having forgotten _components_ taking on the value 0, the value “nil” was used. This implementation, however, was not tested. Thus,
test the current implementation of forgetting and possibly alter the point in the process at which forgetting occurs and how it is implemented.

v. Currently, a trace’s sphere of influence is symmetric in all dimensions. It is, however, very probable that a trace’s influence might extend over a broader range of values in some dimensions and be restricted to a narrower range of values in other dimensions. Therefore, performance could possibly improve if the current symmetric radial basis function were changed to a non-symmetric function such that each dimension has its own defined characteristic radius.

vi. Rework the action feature so that a single component (rather than many components) represents a single action. This way, a trace can be thought of as being connected via a link, as in artificial neural networks, to one of the action output nodes. In addition, enable the model to allow multiple actions to be associated with a given trace.

vii. Currently, only a single task may be defined for the RBMM to complete. Scale this capability so that multiple and differing goals, such as finding food, avoiding monsters and taking shelter, may be simultaneously defined for the model. Having a unique utility and characteristic radius pairing for each goal will allow different actions to be recommended for a given situation according to the agent’s current goal.

viii. Try out other reinforcement schemes to update utility values. Currently, a form of positive reinforcement and positive punishment is used such that some traces’ utilities are updated and others are not. Updating all traces’ utilities, rather than a subset of them, might produce a greater range of learning curves to be modeled, particularly ones where learning occurs very quickly.
ix. The current model assumes that the feedback it receives about the state of the world is immediate feedback concerning only the most recent action taken. Implement a mechanism that allows delayed feedback about the success of the model’s current actions. Thus, the update of the utility and characteristic radius parameters would involve not just the current traces but also the traces involved in previous decisions. In the context of maze navigation, this capability would be useful in mazes where a wrong turn might lead to a number of other intersections before the mistake is discovered.

x. Increase the number of items that can be held in primary memory from a single item to seven, plus or minus two, items. One of these items would remain as the current experience and the others might be previous probes or perhaps echoes that were returned. Increasing the capacity of primary memory will benefit handling more than one task at once and might also benefit a delayed reinforcement mechanism.

xi. Allow the dynamic addition of features and components to the vector that makes up an experience. If a new feature is sensed, the sensed component values are stored from then on and memories that pre-dated the new sensing capability are nil-valued or set to an appropriate default value for the newly-added feature.
References


Appendix A

THE TREATMENT OF NIL

In the Radial Basis Memory Model, the component value “nil” represents the absence of information. In the probe, the action feature is nil-valued because the action has not been determined yet. As well, components of the probe may be nil-valued to indicate the absence of information about those components’ real values. In memory traces, components may be nil-valued to also mean that the real value was forgotten.

Similar to Minerva, when a trace is encoded into secondary memory, each of its component values is individually stored with a probability represented by the learning rate parameter, $L$. Thus, $L$ ranges from 0 to 1, such that when $L = 1$ (which was always its value in the current work), the trace is perfectly encoded in memory. If a component’s value fails to be encoded, then its value is stored as nil.

Now during probing, the distance, $D = \| (\bar{p} - \bar{t}) \|$ where $\bar{p}$ and $\bar{t}$ are the probe and trace vectors respectively, is calculated. If a nil value is encountered, for example, if the $j$th component of trace $i$, $t_{i,j}$, is nil, then the difference $(p_j - t_{i,j})$ between this component and the corresponding component $p_j$ of the probe is set to zero. Thus the distance between the two vectors is the same as if component $j$ had never been defined in the first place.

The activation is calculated as usual, as well as the echo intensity.
The *echo content*, \( C_j \), for *component* \( j \) is,

\[
C_j = \frac{\sum_{i=1}^{m} A_i t_{i,j}}{I_j}
\]

where \( A_i \) is *trace* \( i \)'s *activation* and \( I_j \) is the *intensity* of the \( j \)th *component* of the *echo*.

Again for a *trace* \( i \), if \( t_{i,j} \) is nil, then \( A_i t_{i,j} \) is set to zero. As well, \( C_j \) is nil if \( I_j = 0 \), where \( I_j \) is the *intensity* of *component* \( j \) of the *echo* and is given by,

\[
I_j = \sum_{i=1}^{m} \begin{cases} A_i & t_{i,j} \neq \text{nil} \\ 0 & t_{i,j} = \text{nil} \end{cases}
\]

If the nil value does not appear in any of the *traces* in *secondary memory* because \( L = 1 \) and therefore forgetting is not applied, then the *echo content*, \( C_j \), simplifies to its formula in Section 4.3.1.

A *trace* that does not actively contribute to a *component* of the *echo* either has an *activation* value of 0 or its *component* value is nil. Either way, these *traces* have no relevant information to contribute toward the *component*’s value and so the *echo content* calculations have been constructed so that these *traces* do not impact the value of the *component* in the *echo*. For example, if a *trace* has an *activation* value of 0, then in the calculation of \( C_j \), it contributes 0 to the numerator’s sum and 0 to \( I_j \) in the denominator. Similarly, if a second *trace* has a nil-valued \( j \)th *component*, and the *trace*’s activation is 0.6, then the *trace* contributes 0 to the numerator’s sum and 0 to \( I_j \) .
The three other procedures (recommending an *action*, and updating *utilities* and *characteristic radii*) do not require any modification to handle the addition of nil as a *component* value.