Computational Modeling of Neural Networks and Memory Simulation


Advisor: Minjoon Kouh
Assistant: Aaron Loether

Abstract

Computers and brains have many corresponding characteristics, such as the ability to use logical sequences, inputs, and outputs in order to store and recall information. In the following computer study, both Hebb’s theory and Hopfield’s model are applied to create a computer program that primitively imitates memory storage and retrieval in the human brain. This sheds light on the complexity of the brain and the basics of memory processing. The program is structured to make the computer memorize eighty-five images of NJGSS scholars and then recall them from noisy images. The computer was able to memorize and successfully recall up to six of the eighty-five images on average.

Introduction

Memory, the ability to actively recall past experiences, facts, and events, is a function within the nervous system. The brain, one of the main organs in the nervous system, allows for both the formation of detailed memories as well as the creation of consciousness. Memory is believed to be primarily processed in the hippocampus, a set of small horseshoe-shaped clumps centered in the brain.

Despite having thorough knowledge of the anatomy of the brain, the mechanisms for information and memory processing in the brain are currently poorly understood. Over time, researchers have developed several models to explain these processes. One such model, proposed by Donald Hebb in 1949, suggests that the formation of new synaptic connections leads to associative learning. This fundamental premise of Hebbian learning is the foundation of the Hopfield model, proposed by John Hopfield in 1982. This model embraces the concept of neural circuitry, such as inhibitory or excitatory signals. Neurons relay electrical signals via synapses, the gaps between the output axon terminals of one neuron and the input dendrites of another. These messages either propagate or inhibit action potentials in successive neurons. This model uses binary code to depict the connections between neurons. Points in matrices represent synaptic weights between neurons. Matrices are used to represent the electrical inputs, outputs, and synaptic weights. Synaptic weights represent the effects that synaptic outputs have on neighboring neurons. This computer model is a simplification of complex brain activity, which is not entirely binary.

In the brain, memories from experiences are stored as basic outlines for efficient recollection. When the memory is being recalled, the more specific and relevant information is recreated. Different levels of encoding, or the conversion of sensory stimuli into a retainable format, such as semantic, acoustic, and structural (semantic being the deepest and structural being the least deep), can influence how strongly the memory is stored. Several prominent scientists have spoken about the brain and its computational abilities. Jeff
Hawkins, a neuroscientist, approaches the topic of brain theory. Hawkins highlights that, despite the enormous amount of empirical data concerning brains and neural networks, there is no theory that can bring all of the data together. He stresses that there is an incorrect assumption impeding our comprehension of the brain: Intelligence is defined by behavior. He believes intelligence is defined by prediction, and not behavior, as you know what word is coming at the end of this sentence. Prediction is the source of intelligent behavior, and prediction finds its root in the neocortex. “Intelligence is making predictions about novel events,” he adds. To add to this hypothesis and create a thorough brain theory, Hawkins proposes including memory systems with recollection as a sequence of patterns that are auto-associatively recalled as predictions of future patterns with biological accuracy.

Another leading neuroscientist, Ed Boyden works in the field of optogenetics. Using a combination of lasers and genetic engineering, he implants brains with optic fibers, allowing him to activate special reflective proteins in specific neurons to see their connections. In addition to helping create detailed maps of brain circuitry, this process has been used to cure blindness in mice, and could point the way to cures for Parkinsons or Alzheimers, or even methods of connecting the brain to prosthetics. This neuronal mapping enlightens this project’s approach to the Hopfield Model. With simple programming, a computer can simulate the most unknown, interesting, and complex organ that we know of in the universe.

METHODS

Programming

Main Goal

The brain has the ability to take in complex information, store it in a network of cells, and recall it when presented with stimulus. The goal of this project is to model this phenomenon using computing techniques and further understand brain processes. A common exercise of human memory, the remembrance of human faces, was chosen as an appropriate experiment to test the efficiency of memory recall. The computer is given pictures of eighty-five Governor’s School in the Sciences scholars to “memorize”, which it accomplishes by changing the synaptic weights of its “synapses” using Hebbian learning. The program then recalls the memories based on a stimulus—in this case, a version of the original image “scrambled” by switching the state of random bits between black and white. The objective of the program is to understand the efficiency of memory recall. Information in the programming code was organized with matrices, and due to its efficiency with matrices and vectors, Matrix Laboratory (shortened to MATLAB) was chosen as the programming language of choice.

Hopfield Network and Hebbian Learning Rule

The two major principles of computational neural networks modeled in the memory-recall project are the Hopfield Network and the Hebbian Learning Rule. The purpose of the program is to compress and binarize an image (so it will be suitable for the computer program), memorize it, and then recall the picture at a later date using auto-associative memory. In short, this means that in a network of related neurons (connected by synaptic weights), a stimulus that activates a single neuron will activate all associated neurons. This program equivalent of associative memory compares two binary matrices and records the strengths of correlation in the W matrix.
The Hopfield Network is based on the Hebbian Learning Rule, which states that repetitive action potentials manipulate the plasticity of neurons. Its main precept is that neurons that “fire together, wire together.” For example, if neuron A continually activates neuron B, the synaptic connection between the neurons increases.

A final function of the Hopfield Network is to asynchronously update the states of different elements in the neural network. This means that a function processes changes in the neural network, as a function of time, based upon the synaptic weight matrix. Because the network is asynchronous, only one neuron is updated at a given time, congregating inputs from all the other neurons in the network. From these changes, the program eventually reaches a stable state and accomplishes its recall function.

The design of the program is inspired by the dynamic flow of memory. Neurons that recall faces, places, and past experiences translate initial stimuli into far clearer memories, deriving accuracy from correlations between related neurons. Just as the brain stores information and then draws pieces from that reservoir of data to complete memories, the synaptic weights attempt to unscramble images from the synaptic weights recorded previously. The entire structure of the program is thus analogous to a simple idea: a ball on a hill. On a bumpy landscape, the ball has the potential to roll into a variety of locations. When presented with a push (a stimulus), the ball is spurred to action, moving along until it hits a valley that contains its motion. This represents the stable state of the neuron or the final unscrambled output of the program. At this point, the neuron, after much interaction and interpretation, has translated its initial “spark” into a final output: a memory.

**Variables**

A neural network in its most basic form can be represented as a series of inputs and associated outputs. Inputs are composed of the signals emitted by each neuron in the neural network; in turn, the outputs are action potentials that occur at corresponding intervals. The variables in the Hopfield model translate into numbers that represent biochemical aspects of neural systems. For instance, synaptic weight represents the various factors that contribute to the strength of an action potential (i.e. the impact of the neurotransmitters that jump from the pre-synaptic to post-synaptic neuron).

Matrix $P$ is then used to create a two-dimensional matrix $W$ that contains the synaptic weights (level of correlation) between each neuron for each image. In keeping with the binary nature of the project, synaptic weights are either 1 or -1, with 1 representing that the two neurons “fire together” and are thus “wired together” as described by Hebbian Theory (Figure 1). The $W$ matrix (with the number of image patterns as the columns) sums the strength of the correlation between different neurons for each and every picture. From a Hebbian perspective, synaptic weights govern memory storage, and thus the matrix $W$ acts as a sort of “key” to recreate images even from incomplete inputs. When a net positive correlation is observed between two neurons, the activation of one neuron will lead to the activation of the other, and a chain reaction will continue through all connected neurons.
To recall the image, the same neurons that were on when the image was memorized had to “fire” again, and the neurons that were off originally had to be off. Thus, a two-dimensional matrix $Y$ was necessary to record the state of each neuron as a function of time. With each passing index of time, $t$, the neurons in $Y$ would change their state based on the connections made with each other as described by $W$. The equation describing this relationship is

$$Y(t) = W \ast Y(t - 1)$$

$Y$ thus evolves as a matrix that shows whether each neuron is on or off (the rows) at each time step.

To properly simulate the mechanisms of human memory, the image is scrambled before it can be recalled, so the first column of $Y$, which is the initial state of the neural network when it is trying to recall a memory, is scrambled. This initial state resembles the neural network when it initially receives stimulus (in this case, a poorly rendered image). A matrix $S$ is then created to hold the scrambled values for each picture. Any column in this matrix can then be converted into the vector $Y_0$, which becomes the initial state of the $Y$ matrix. This will eventually unscramble and recall the image. The column of $S$ chosen indicates the image that the person wants to recover.

Overall, the flow of the program is as follows (in terms of variables): the matrix of pictures, $\textbf{Y}_{\text{large}}$, is converted into a two dimensional matrix of binary values known as $\textbf{P}$; $\textbf{P}$ is then memorized by the computer, changing the synaptic weights of neurons as recorded in matrix $\textbf{W}$. Then comes the test of memory. $\textbf{P}$ is scrambled into matrix $\textbf{S}$, any column of which can be used as the initial state of matrix $\textbf{Y}$, which represents the evolution of the neural network over time. Finally, after the maximum time, $t$, passes, the end neural state will, by design, be identical to the initial, rendering the initial image.

The accuracy of the image recollection was calculated by a comparison between the desired output and the actual output. In this model, $\textbf{A}$ represents the accuracy, $\textbf{Y}$ represents the actual output, and $\textbf{Y}_0$ represents the desired output. Vector $\textbf{D}$ is defined as the absolute value of the difference in the outputs. Because $\textbf{Y}$ and $\textbf{Y}_0$ are binarized matrices, $\textbf{D}$ will demonstrate the discrepancies in the output such that the value 0 represents no difference in the elements and a 1 represents a difference in the elements. The accuracy function is a ratio.
between the discrepancies of elements (represented by the sum of the elements in the matrix \( D \) and the total number of elements).

Functions

To model the human brain with a computer system, the neural processes of the mind are interpreted as a series of functions in MATLAB. The functions are grouped into several overriding stages. The first stage is the preparation phase, the function \texttt{prepare\_P.m}, which serves to convert the images into a format that can be used in later functions within the MATLAB program. The first step in the preparation phase is to read the files into MATLAB. This task is accomplished using the function \texttt{imread} to transfer the images into the program. The next step is to re-size the images to dimensions suitable to the program. The resolution of the images is decreased, using three “for loops” to cycle thorough the dimensions of the three dimensional matrix, selecting only one pixel for every “p” number of pixels of the original. This process simplifies the original image into specified dimensions, which are more manageable for MATLAB, given its limited memory and processing capabilities. The final step in the preparation phase is the binarization of the images, transforming the three-dimensional matrix into a two dimensional matrix. We used the “mean” function to compute the mean values of the columns of pixels in the original image, where the columns represented individual images, and to generate a matrix of these values. We then used a “for loop” to compare the value of each individual pixel to the mean for the corresponding image, assigning the value of 0 to the pixel if it was below the mean, and the value of 1 if it was above the mean. This binarization created a matrix of ones and zeros, or \( P \), in the place of the images, which we could subsequently use in later stages of the program.

The second stage of the program is the memorization phase, function \texttt{autoConnectivity.m}, which simulates the storing of a memory. We input the matrix \( P \) from the first stage and generate a matrix representing the synaptic weights of the synapses connecting the neurons. \texttt{AutoConnectivity.m} creates a matrix of zeros, \( W \), of the same dimensions as \( P \). The program calculates the cross-correlations between two “neurons,” or values in the matrix, and using the code

\[
W = W + (2 \times P - 1) \times (2 \times P' - 1)
\]

Thus, if two neurons “fire” at the same time in a pattern, or column, of matrix \( P \), \texttt{autoConnectivity.m} will add 1 to the corresponding synaptic weight in the matrix \( W \). Conversely, if the two neurons do not fire together, the program will subtract 1, due to the negative correlation. This function results in a \( W \) matrix, which contains the correlations between the neurons in the \( P \) matrix, or synaptic weights, represents a pattern or a memory, which is defined by the synaptic weights.

The third phase of the program is the recall, simulated by the function \texttt{asynchUpdate.m}. The \( W \) matrix from \texttt{autoConnectivity} is then inputted into this function, along with a \( Y_0 \) matrix, composed of ones and zeros, which represents an attempt to recall a memory or pattern.

Then, using another “for loop,” the time interval allotted for the recall of the memory, the variable, \( t \), is run through. At each time step, one neuron updates its state, representing the “asynchronous parallel processing” emphasized in Hopfield’s work. The one neuron that is
randomly chosen to update uses the synaptic weight matrix $W$ and multiplies it by the previous state of the $Y$ matrix following the formula

$$Y(t) = W \ast Y(t - 1)$$

applying this to only one neuron. The process follows for “nTs” time steps until the network falls into a stable state, which means no new neurons are activated.

In order to judge the performance of the program, an accuracy function is created. A simple pixel to pixel comparison is not enough to convey the true strength of the recall because it gives too much emphasis to the white spaces that surround the image. In fact, unrecognizable images receive values as high as 85%. Considering that in recalling an image, the neurons that represent the black pixels are most important (these are the pixels that actually define the image), a new function $\text{accuracy}_3$ is developed which considers only the black pixels. $Q$ is a matrix that holds a one if both corresponding pixels of the output and target image are both black, a negative one if the two are opposite, and a zero if both are white. This matrix is summed and then divided by the number of black pixels that ever turned on. This gives a value between negative one (where all the black pixels are in the wrong place) and one (where all the pixels are in exactly the right place). See Figure 2 for examples of how distorted pictures with certain $A$ values are.

A more in-depth flow of the program can thus be demonstrated. A set of pictures is read by the program and then cropped, binarized, and stored in a two-dimensional matrix by the function $\text{prepare}_\ P$. The correlation between each neuron is then determined using the function $\text{autoConnectivity}$. Then, the scramble function toggles random bits given a percent of noise, toggles random bits to generate scrambled versions of each picture. Finally, recall is done by $\text{asynchUpdate}$, which asynchronously updates the state of neurons in the network using the correlation between them to try and create the original image, the success of which is monitored by the $\text{accuracy}_3$ function.

**DATA AND ANALYSIS**

Figure 2 – Above is a sample run of the memory modeling program. To the far left is the inputted image, in the center is the scrambled image, and to the right is the program’s recalled image.

Pictures of students were used as test images as to whether the neural network could memorize and recall then clearly. Figure 2 is a sample image that was processed with a .1 noise level. In this example, the output image had an accuracy of .7963.
To understand the efficiency of the memory model, the effects of noise and number of pictures in the neural network on accuracy were analyzed. Figure 3 illustrates the relationship between the number of pictures memorized by the neural network and the ability of the network to recall pictures that it had memorized. According to the trend, a small number of pictures memorized in one neural system generates a very high accuracy. Having to memorize more than six pictures causes the accuracy to reduce largely and fluctuate at a lower performance range. This graphical behavior represents the inability of a neural system to be overloaded. In a true brain system, different networks memorize different information. Therefore, since our program simulates only one system, it cannot handle memorizing more than a few images.

Also, the performance had a significant decrease after six pictures because orthogonality was lost with an increasing quantity of pictures memorized. This result was expected because too many similar images are not easily recallable. Similar images lead to very close stable states; the neural network may not be able to recognize the correct stable state for a picture.

Figure 3 – Above is a graph of number of pictures memorized by the neural system vs the accuracy of the system at recalling the images.
Figure 4 – Above are graphs of noise added to the pictures vs. the performance of the recalled images. Figure 4a shows the relationship between the two components. Figure 4b is a residual plot that portrays the difference between the input and output performance.

The second test that was done was one of scrambling added to the initial image versus the accuracy of the final recall picture, as shown in Figure 4a. The effect of noise on performance was analyzed to determine the efficiency of recall. As the noise added to the picture increased, the performance constantly decreased. This was simply because with more scrambling of the image, it is more difficult to recall it.

CONCLUSION

This model efficiently stores and recalls complex data by implementing functions in a similar fashion to that used in natural brain activities. Like the brain, the computer model has difficulty recalling accurate images of very similar pictures (i.e. nonorthogonal images like Governor’s School students’ photos). Images that have no obvious patterns in common are said to be orthogonal. Similar data points, such as facial structures, are easily confused and the “ball” of the memory has a difficult time settling in the correct stable state. Because there
are so many similar images in the set of the eighty-five scholars, the “recalled” information is simply a muddled face-like structure, with no defined features. A key aspect of the model is that it can recall information after it has been scrambled (like recalling a person’s face after a month has passed). But, the more scrambled, or “noisy”, the stimulus image is, the more difficulty the program has recalling it; therefore, the accuracy of the recalled image decreased. Consequently, the full recall capacity of the model is reached when using orthogonal images. In other words, when starting with random binary patterns and scrambling the image, the computer is able to precisely display the true image at an accuracy level of one hundred percent. Because faces have basic patterns in common, they are more difficult to accurately recall. This implementation of the Hopfield model, overall, has confirmed both the model’s successes and failures, illuminating how neuronal modeling can recreate and thereby be used to study the brain’s functions.

DISCUSSION

Weaknesses and Simplifications

As with any other model, this specific model of a neural network has simplified certain aspects of the brain and thus has a few weaknesses. One of the most obvious weaknesses is the binary nature of the program. Although action potentials fired by neurons are indeed “all-or-nothing” and can theoretically be well modeled with binary constructs, synaptic weights are not all-or-nothing, and in a real neural system, there are many gradations of synaptic weight, which can change the intensity of a response. The model overlooks this inherent and important dimension of the brain, and therefore only allows for a single, standard response.

The model also asserts that each neuron represents one pixel of the picture that is to be memorized and can only deal in black and white images. Although the neurons for sight do work somewhat in this way, the black-and-white spectrum restricts the input that the program can take. In reality, each neuron would represent a small idea that, when connected to other ideas, would evoke a memory. The limited memory of MATLAB also forced the compression of the images rather than their presentation at full resolution. Obviously, the human brain has far more capacity to build higher resolution images than MATLAB does, but the program still suffices to accomplish its purpose.

Several aspects of this program had to be simplified. First of all, this model recreates memory only as if it were a separate and isolated function of the brain. In reality, the systems of the brain responsible for memory are intricately connected with systems of sense, emotion, and motor function. For simplicity, only memory functions are present in this project. The model, in keeping with Hopfield’s proposed model, is based on neurons having only binary states, on or off (this represents the fact that action potentials are all-or-nothing). The model, however, neglects the idea that the frequency of action potentials is also a means of encoding information. Further differing from the brain, the model’s synaptic weights were also made binary (although the connections between two neurons for different representations were summed); in a real neural network, synaptic weights vary greatly with time. When new synapses are created, the synaptic weight is increased; when synapses are broken, the weight is decreased; when more neurotransmitter is released, the weight is increased; and various other conditions can cause changes in synaptic weights. Our model considered only “connected” and “disconnected”; a necessary approximation of the Hopfield Model due to our limited understanding of the mechanisms that would cause these analogue changes. The
inability of the program to memorize a great deal of images could be in part because of the above simplification. Our brains have an ability to store data in an analogue form, with synaptic weights having other options besides connected and disconnected, which would allow each neuron to have more states, which in turn allows for more pictures to be memorized. Each pixel in the program is represented by a neuron, and while this may be true for vision, in most cases, a neuron in the brain is usually associated with an idea or with a feature. Rather than just memorize and recall images, the brain has the ability to memorize concepts and ideas, even the ability to synthesize different forms of information. The Hopfield Model is far from this.

Potential for Computational Neural Systems in the Future

The very nature of modeling does imply some sort of simplifications in order to exemplify the behaviors of some system. A simplified model can nevertheless be an effective model, and this model is such an example; it demonstrates the basics of one possible mechanism by which human memory could work: neurons representing a simple idea are activated by a certain stimulus, and they in turn activate neurons that represent connected ideas, causing a cascade effect until the network settles into a stable position, which is a memory. From this simple model, more complex models can be implemented, taking into account the different parameters that were overlooked; for example, the synaptic weights could be assigned more realistic values that are based on actual physical phenomena, such as amount of neurotransmitter released, and could thus take on several values besides zero and one. Having models of memory and the brain in general has crucial implications for the world. It would be much easier to treat disorders of the brain if we had a comprehensive theory of the brain and could trace neural networks to the roots of the problem. Learning could also be made more efficient if the stimuli that lead to memory and to the synthesis of information were better understood. Teachers could modify lessons so that students’ brains encounter the most effective stimuli possible. Understanding neural networks also has great potential for advancing computing. The brain is still more advanced than any computer in existence today, and understanding the mechanics that make the brain such an advanced machine could help improve current computers, with consequences radiating through the technological world.

REFERENCES

APPENDIX

Appendix 1a:

Main

function M = main

D = [285 380];
N = 85;
C = 3.5;

P = prepare_P(N,D,C);
W = autoConnectivity(P);

S = scramble(P,x);

if img
    Y0 = S(:,img);
else
    img = ceil(N*rand);
    Y0 = S(:,img);
end

l = (size(P,1))^(.5);
w = l;

a = accuracy(S(:,img), P(:,img));
disp(a);

Y=synchUpdate(Y0,W,nTs);
Pimg = reshape(P(:,img),w,l);
subplot(1,3,1);
imagesc(Pimg); axis image;

Simg = reshape(S(:,img),w,l);
subplot(1,3,2);
imagesc(Simg); axis image;

subplot(1,3,3);
imgDisp(Y,l,w);
a = accuracy(Y(:,nTs), P(:,img));
disp(a);
a=accuracy(Y(:,nTs),Y(:,nTs-1));
disp(a);
Appendix 1b:  
**Prepare_P**

```matlab
function P = prepare_P(Npic,D,C)
cropsize=15;
p = C;
N = Npic;
All = zeros (D(1), D(2),N);

for x = 1:N
    filename = [num2str(x) '.jpg'];
    m = imread (filename);
    All(:,:,x) = m(:,:,1);
end
crop = ceil((D(2)-(D(1)-(2*cropsize)))/2);
All = All(cropsize:D(1)-cropsize, crop:D(2)-crop+1,:);
All_small = compress(All,p);

for k = 1:N
    img_tmp = binarize(All_small(:,:,k));
P(:,k) = img_tmp(:);
end
```

Appendix 1c:  
**Scramble**

```matlab
function P = scramble (P0, x)
    [nBit, nPicture] = size(P0);
P = P0;
record = P0;
changebits = ceil(nBit*x);

for col = 1:nPicture
    for b = 1:changebits
        r = ceil(nBit*rand);
        while P(r,col) ~= record(r,col)
            r = ceil(nBit*rand);
        end
        P(r,col) = P(r,col)==0;
    end
end
```
Appendix 1d:

**autoConnectivity**

function W = autoConnectivity (P)

[Nneuron,Npattern] = size(P);
W = zeros(Nneuron,Nneuron);

W = (2*P-1) * (2*P'-1);

MSK = (ones(Nneuron) - eye(Nneuron));
W = W .* MSK;

Appendix 1e:

**asynchUpdate**

function Y = asynchUpdate (Y0, W, nTs)

Nneuron = length(Y0);
Y=zeros(Nneuron,nTs);
Y(:,1) = Y0;
for t = 2:nTs
    randIdx = ceil(rand*Nneuron);
    q = W(randIdx,:)*Y(:,t-1);
    Y(:,t) = Y(:,t-1);
    Y(randIdx,t) = q>0;
end

Appendix 1f:

**Accuracy**

function A = accuracy (Y, Y0);
N=size(Y,1);
D=zeros(N,1);
for i=1:N
    D(i)=abs(Y(i)-Y0(i));
end
A=1-sum(D)/N;
end

Appendix 1g:

**ImgDisp**

function imgDisp(Y,L,W)

colormap gray

S = size(Y);
N = S(2);
Y1 = Y(:,N);
Y2 = reshape(Y1,L,W);
imagesc(Y2);
axis image;
colormap gray;
Appendix 1h:

Shown here are some sample values of the correlation/performance value given by the accuracy for images scrambled by varying degrees. Note that the correlations range from 1 (a perfect image) to about .7, .15, 0, -.7 and -1 (an inverted image).