ABSTRACT

JACOB, THEJU ISABELLE. A Neural Model for Straight Line Detection in the Human Visual System. (Under the direction of Wesley Snyder.)

Building a model for how the visual cortex identifies objects is a problem that has attracted much attention over the years. In this thesis, the philosophy of Hough Transform is utilized to explain how the visual cortex could possibly detect straight lines. Practical implementations of the Hough Transform construct accumulators in a transformed parameter space. In this work, a neural network that can learn to form accumulators for straight line detection in an unsupervised manner is outlined. Ideas from Hebbian Learning and Self Organizing maps are borrowed to build the network. Experiments involving human observers are discussed and the results obtained are shown to add further weight to the hypothesis.
A Neural Model for Straight Line Detection in the Human Visual System

by
Theju Isabelle Jacob

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2014

APPROVED BY:

_________________________  _________________________
Dror Baron                  John Franke

_________________________  _________________________
Edgar Lobaton               Cranos Williams

_________________________
Wesley Snyder
Chair of Advisory Committee
DEDICATION

To Mannathoor.
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Wesley Snyder, and all of my committee members - Dr. Dror Baron, Dr. John Franke, Dr. Edgar Lobaton, and Dr. Cranos Williams for their support and many useful suggestions. I thank Dr. Jing Feng and HeeSun Choi from the Department of Psychology for working with me throughout the design, set up, and execution of the human experiments. I thank Dr. Brian Hughes for his help in refining some of the more mathematical aspects of my work. Thank you Benjamin Riggan, Yongqiang Huang, and Sanjeev Khanna - my current and former lab members and friends, for many interesting discussions. I thank my family and friends for their support.
# TABLE OF CONTENTS

## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Chapter 1 Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Hough Transform</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2 The Visual Cortex</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Ganglion Cells, Simple Cells and Complex Cells</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Ocular Dominance, Orientation Columns and Receptive Fields</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Hough Transform Structure in the Cortex</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 3 Hebbian Learning and Self Organizing Networks</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Hebbian Learning</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Self Organizing Maps</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4 Architecture and Methods</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0.1 Network Architecture</td>
<td>27</td>
</tr>
<tr>
<td>4.0.2 Dynamic Network Equations</td>
<td>28</td>
</tr>
<tr>
<td>4.1 Results</td>
<td>32</td>
</tr>
<tr>
<td>4.1.1 Network Training</td>
<td>32</td>
</tr>
<tr>
<td>4.1.2 Performance Comparison</td>
<td>35</td>
</tr>
<tr>
<td>4.1.3 Extension to All Orientations</td>
<td>41</td>
</tr>
<tr>
<td>4.2 Analysis</td>
<td>48</td>
</tr>
<tr>
<td>4.2.1 Maximization of Mutual Information, Under Constraint.</td>
<td>49</td>
</tr>
<tr>
<td>4.2.2 Experimental Results</td>
<td>51</td>
</tr>
<tr>
<td>4.3 Discussion</td>
<td>53</td>
</tr>
<tr>
<td>4.3.1 Supervised Learning</td>
<td>55</td>
</tr>
<tr>
<td>4.3.2 Self Organizing Maps</td>
<td>57</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5 Human Experiments</th>
<th>59</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Experiment Set Up</td>
<td>60</td>
</tr>
<tr>
<td>5.2 Results</td>
<td>67</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6 Signal Separation</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Separation of Lines</td>
<td>73</td>
</tr>
<tr>
<td>6.2 Results and Discussion</td>
<td>75</td>
</tr>
<tr>
<td>6.2.1 Comparison with SOM</td>
<td>80</td>
</tr>
<tr>
<td>6.2.2 Comparison with Oja’s rule</td>
<td>82</td>
</tr>
</tbody>
</table>

| Chapter 7 Conclusion and Future Work | 83 |

iv
# LIST OF FIGURES

| Figure 1.1 | Parameters of a straight line in the Hough space. | 3 |
| Figure 1.2 | Straight lines in Hough space. (Image Source: Nvidia Website) | 4 |
| Figure 2.1 | Visual Pathway Structure from [20] | 7 |
| Figure 2.2 | Cells in Retina from [20] | 8 |
| Figure 2.3 | Distribution of Rods and Cones in Retina from [1] | 8 |
| Figure 2.4 | Visual areas from [43] | 9 |
| Figure 2.5 | Simple and Complex Cells Receptive Fields. | 10 |
| Figure 2.6 | From Hubel [26] - ‘A tilted line segment shining in the visual field of the left eye (shown to the right) may cause this hypothetical pattern of activation of a small area of striate cortex (shown to the left). The activation is confined to a small cortical area, which is long and narrow to reflect the shape of the line; within this area, it is confined to left ocular-dominance columns and to orientation columns representing a two o’clock - eight o’clock tilt.’ | 12 |
| Figure 2.7 | The first figure shows the drift of receptive fields across 3 mm in cat cortex. Only 4 or 5 receptive fields recorded in the first tenth of each millimeter is shown. The second figure shows the growth in receptive field sizes with increased eccentricity from the fovea, in a macaque monkey, from Hubel [26]. | 13 |
| Figure 2.8 | ‘Ice-Cube model’ of the cortex. The figure shows the division of the cortex into 2 sets of slabs, one for ocular dominance and one for orientation, from Hubel [26]. | 13 |
| Figure 3.1 | Structure of a neuron [7] | 16 |
| Figure 3.2 | Activity at a Synapse [7] | 17 |
| Figure 3.3 | Oja’s rule. Effective input at, say node 1, is now $x_1 - yw_1$. | 19 |
| Figure 3.4 | Anti-Hebbian Rule leads to decorrelation in output neurons: $\Delta w_{12} \propto -y_1y_2$. | 21 |
| Figure 4.1 | Simulated cells in the visual pathway | 24 |
| Figure 4.2 | Orientation sensitive cells at every pixel location in the image plane. Let $\theta_1$ correspond to the horizontal line. When two adjacent points along the horizontal direction are excited, their $\theta_1$ sensitive cells end up exciting the same accumulator cell. $\rho_1$ represents the distance of the horizontal line from the origin. | 25 |
| Figure 4.3 | Connections between pixels in the image plane and the cells in the accumulator. For clarity, only the connections which give rise to the two strongest responses in the accumulator array are shown. | 26 |
Figure 4.5 Winner computation example. Suppose we have an array of neurons. Let the outputs for neurons 1, 2, and 6 be 0.1, 0.7, and 0.75 respectively. All other neuron outputs are presumed 0. Conventional wisdom would declare neuron 6 as the winner. In our computation, the neuron outputs are multiplied by a Gaussian centered at the neuron locations and added together. The resulting maximum is now located at neuron 2. Neuron 2 is declared the winner.

Figure 4.6 Resulting network, after training

Figure 4.7 Full diagonal line.

Figure 4.8 Full diagonal line, different position.

Figure 4.9 Occluded diagonal line.

Figure 4.10 Horizontal line segment.

Figure 4.11 Network trained of diagonal line, presented with a clutter.

Figure 4.12 Separation of peaks example. When two line segments are aligned, only one peak is observed in the accumulator. As the line segments separate, the peaks become more distinct. In the case of addition of noise, more peaks of similar amplitude appears in the accumulator as the standard deviation of the added noise increases. As the peaks become more distinct, the difference between the highest peak in the accumulator and the second highest peak in the accumulator diminishes.

Figure 4.13 Variation in angle test images. The angle between the two line segments increases as we move from (a) to (d).

Figure 4.14 Variation in distance test images. The separation in pixels between the two lines increases from (a) to (d).

Figure 4.15 Variation in noise test images. The standard deviation $\sigma$ of the added noise increases from (a) to (d).

Figure 4.16 Decay of difference in peaks with difference in angle between two line segments for the 4 different implementations.

Figure 4.17 Decay of difference in peaks with difference in distance between two line segments for the 4 different implementations. The same values were obtained for Accumulator Array Implementation of the Hough Transform and the Conventional Hough Transform, and hence AT plot is omitted.

Figure 4.18 Decay of difference in peaks with addition of noise for the 4 different implementations. The same values were obtained for Accumulator Array Implementation of the Hough Transform and the Conventional Hough Transform, and hence AT plot is omitted.

Figure 4.19 Plots showing the point where the accumulator claimed 2 or more lines were present.
Chapter 1

Introduction

The human visual system is vastly superior to man made vision systems in its ability to identify objects. Variations in illumination, scale, and position often do not hinder a human observer from identifying a target object. Perception of depth and color further adds to the observer’s ability to distinguish between different targets of interest.

The primary area of the human brain concerned with vision is the visual cortex. Building a model for how the visual cortex identifies different objects is a problem that has attracted much attention over the years. Both top-down [3] and bottom-up [6, 28] approaches have been proposed towards this end. Top-down models start at a high-level representation of the incoming visual input, while bottom-up models start with simple features within the input and then move to more complex features.

Oriented lines are one of the first features detected by the visual cortex in a bottom-up approach, see chapter 2. Lines and edges are interesting from an object recognition point of view because they mark the boundaries of objects in images. In this thesis, we limit ourselves to detection of straight lines. We utilize the philosophy of the Hough Transform to explain how the visual cortex could possibly detect straight lines, under conditions of
discontinuity and noise. A neural model that can learn a Hough Transform-like structure in an unsupervised manner is the main takeaway from this work. Optical illusions like the Poggendorff illusion could potentially find an explanation in the framework of our model.

The main ideas addressed by this thesis can be stated as follows:

- **Hypothesis 1:** The human visual system may have a Hough Transform like structure for straight line detection in the primary visual cortex.

- **Hypothesis 2:** A neural network can learn such a Hough Transform like structure.

Hypothesis 1 is addressed in both chapter 2 and chapter 5. Chapter 2 outlines characteristics of the visual cortex that point to the presence of a Hough Transform structure in the cortex, while chapter 5 outlines human experiments that add support to such a computation in the cortex. Hypothesis 2 is addressed primarily in chapter 4.

Models for object recognition based on visual cortex characteristics like selectivity and scale/position invariance [59, 64, 63] and invariance to lighting conditions [40] have appeared in literature previously. Neural activity models that lead to visual cortex-like characteristics are also found in the literature. For example, learning a sparse code for images as a model for neural activity is discussed in [54, 17]. Predictive coding as an explanation for visual processing in the cortex is discussed in [58]. Information maximization as a goal of sensory coding is discussed in [5, 42, 41]. A self-organizing map architecture capable of extracting features similar to that extracted in the early stages of visual processing is discussed in [34].

The rest of this chapter is devoted to the details of the Hough Transform for straight lines. A breakdown of the organization of the remaining chapters of the thesis can be found in the concluding paragraph.
1.1 Hough Transform

The Hough Transform is a popular feature detection technique in computer vision applications [11]. The Hough domain, when used to detect straight lines, is characterized by two parameters: 1) the orientation of the line, and 2) the shortest distance from the origin to the line, Fig. 1.1a. The underlying parametric transformation from Cartesian coordinates is represented by the following equation:

\[ \rho = x \cos(\theta) + y \sin(\theta) \] (1.1)

Every point in the Cartesian plane \((x,y)\) gives rise to a sinusoid in the Hough domain Fig. 1.2. The sinusoids from points lying along a line intersect at a single point representing the \(\rho, \theta\) values of the line. A straight line in the Cartesian domain is thus represented by a single point in the Hough domain.

Practical implementation of the Hough Transform algorithm relies on an accumulator
array $\mathcal{A}(\rho, \theta)$ with all possible orientations and distances. The orientations usually vary from 0 to $\pi$. The distances vary from 0 to the maximum distance possible in the image plane, usually the length of the diagonal of the image. Once the accumulator is in place, we iterate through the pixels in the test image, and for pixels of interest, we compute the $\rho$ value for every value of $\theta$ and increment the corresponding points in the accumulator.

Every point lying along a line in the Cartesian domain contributes to a set of points in the accumulator, Fig. 1.1b. The parameters of the line are obtained by looking at the $\rho$ and $\theta$ that give the maximum value. An entire line can be thus deduced from a single point in the accumulator array. The transform is insensitive to clutter and partial occlusion, as we rely on a voting process to determine the parameters of the line.

It is well established from the experiments by Hubel and Wiesel (see Chapter 2) that the primary visual cortex contains cells which are sensitive to the orientation of the visual stimuli. It is further known that the orientation sensitive cells aggregate into columns, and that the sensitivity of the columns themselves varies in a sequential manner. We also know that the cells of the visual cortex are organized into retinotopic maps. That is, neighbouring areas in the visual fields map to neighbouring areas in the cortex itself [29, 30, 31]. All of these results point to the possible existence of a structure similar to
an accumulator in the cortex.

In this thesis, we examine the possibility of the existence of such an accumulator array in the cortex, with neurons or groups of neurons acting as accumulator cells. We address how the cells in the cortex could self organize into accumulator arrays in an unsupervised manner. Starting at the orientation sensitive layer of neurons, we show how the subsequent layers could self organize to detect the location of a given straight line. We borrow from Hebbian Learning and Oja’s rule to do this. Experiments involving human subjects were conducted to add support to our hypothesis.

We begin with a discussion on the relevant aspects of the Visual Cortex in Chapter 2. Chapter 3 reviews Hebbian Networks and Competitive Learning Algorithms. Our network and the results obtained are described in detail in Chapter 4. Chapter 5 outlines experiments involving human subjects and the results obtained. An additional network model capable of acting as a precursor to the network from Chapter 4 is outlined in Chapter 6. We conclude in the final chapter.
Chapter 2

The Visual Cortex

The human visual system enables us to process and respond to all kinds of visual stimuli. Various structures along the visual pathway, shown in Fig. 2.1, constitute the visual system. Starting at the retina, neural information travels through the optic nerve to reach the lateral geniculate nucleus (LGN). Information travels further from the LGN to reach the visual cortex, the area of the cerebral cortex responsible for processing of visual information.

The retina itself is a complex structure, with rods and cones acting as photodetectors, and layers of horizontal, bipolar, amacrine, and ganglion cells above them before optic nerve fibres are reached, Fig. 2.2. The fovea is located almost at the center of the retina, representing about $5^\circ$ of visual angle. The highest concentration of cone cells and a near absence of rod cells are found at the fovea, Fig. 2.3. The outer region surrounding the fovea is called perifovea.

The visual cortex is divided into several areas named V1, V2, V3, and V4, Fig. 2.4. In this study, we are primarily concerned with V1, also called the primary visual cortex or striate cortex. The layered or striped appearance of the visual cortex in cross section
caused it to have the name ‘striate cortex’. The striate cortex is divided into layers numbered from 1 through 6 based on cell density.

Next considered in this chapter is the architecture and functionality of the visual cortex as described by Hubel [26]. We start with the hierarchial structure of cells found in the visual pathway and cortex and a description of their functionality. We then examine how the cells are grouped across the cortex, and how their organization could imply an architecture suitable for Hough Transform computation.

### 2.1 Ganglion Cells, Simple Cells and Complex Cells

We know from [26, 27] that ganglion cells in the retina fall into two categories: on-center and off-center. On-center cells give high responses when light stimulus falls on the
Figure 2.2: Cells in Retina from [20]

Figure 2.3: Distribution of Rods and Cones in Retina from [1]
center of their receptive fields, and low responses when light falls outside the center of their receptive fields. The opposite is true for off-center cells. Their receptive fields hence approximate a second order derivative function. Both are found equally across the retina, and appear to be intermixed. The receptive fields of these cells are circularly symmetric, and the orientation of the stimulus does not matter. The receptive fields of cells in the LGN have the same structure as the retinal ganglion cells that feed into them. The cells in the first stage of the visual cortex respond like the cells in the LGN as well.

After we pass the earlier stages of the cortex, we come across the simple cells. Simple cells have regions of excitation and inhibition which are separated by a straight line or 2 parallel lines, Fig. 2.5a. The most commonly occurring type of simple cells have a narrow region of excitation flanked by larger regions of inhibition. Spatial summation effects of the excitation and inhibition regions are observed. That is, if the stimulus fills more of the excitation or the inhibition region, more of excitation or inhibition reaction is observed.
in the output of the cell.

Complex cells, which make up about three quarters of cells in the visual cortex, respond only to specifically oriented inputs. They may or may not respond to stationary input, but if they do respond to stationary input, they respond irrespective of where the input is located in the receptive field. That is, they do not appear to have the excitation and inhibition regions as were observed in cells in previous stages. Complex cells also have larger receptive fields than simple cells, Fig. 2.5b.

A typical orientation-selective cell does not respond at all when the line is oriented 90 degrees to the optimal. All orientations are about equally represented in the cortex.

Orientation sensitive cells can be modeled using Gabor filters [9]. Gabor filters are usually used for detection of edges in images. The following equations represent the operation of a Gabor filter, where $x$ and $y$ refers to the coordinates in the $x$-$y$ plane and $\theta$ is the orientation of interest.
\[ x' = x \cos(\theta) + y \sin(\theta) \]
\[ y' = -x \sin(\theta) + y \cos(\theta) \]
\[ g_f(x, y, \theta) = \exp(-\frac{(x')^2 + \gamma^2(y')^2}{2\sigma^2}) \cos\left(\frac{2\pi x'}{\lambda}\right) \] (2.1)

### 2.2 Ocular Dominance, Orientation Columns and Receptive Fields

On inserting a probe perpendicular to the cortical surface, Hubel and Wiesel found that cell after cell favored the same eye - that is, the left eye or right eye. On inserting a probe parallel to the surface, the dominance of the left eye or the right eye was found to alternate back and forth. The reason why this phenomena occurs is thought to be found in the way the geniculate fibers (which are monocular) terminate in the cortex. A complete cycle of right eye and left eye dominance is found once every millimeter.

Similar to ocular dominance, orientation preference was found to remain constant in vertical penetration of the cortex. On inserting the probe parallel to the surface of the cortex, the orientation shifted by 10 degrees clockwise or counterclockwise for every 0.05 mm advance. A complete 180 degree shift was observed once every millimeter. These observations allowed Hubel and Wiesel to define a local coordinate system on the visual cortex, Fig. 2.6. Ocular dominance and orientation groupings of cells appeared not to influence each other.

When a probe was inserted perpendicular to the striate cortex, it was found that the receptive fields of the cells were located at the same place with small random variations in position, and considerable overlap between fields. The term *aggregate* receptive field refers to the area occupied by the superimposed receptive fields in a given layer (layer
Figure 2.6: From Hubel [26] - ‘A tilted line segment shining in the visual field of the left eye (shown to the right) may cause this hypothetical pattern of activation of a small area of striate cortex (shown to the left). The activation is confined to a small cortical area, which is long and narrow to reflect the shape of the line; within this area, it is confined to left ocular-dominance columns and to orientation columns representing a two o’clock - eight o’clock tilt.’

3, 4 etc.) under some point on the cortex. In a given layer, the aggregate receptive field could vary from a few minutes in the foveal region to a few degrees in the far periphery. On moving the probe parallel to the cortical surface, a steady drift in position across the visual field was observed, Fig. 2.7. The rate of movement was observed to be constant relative to the size of the receptive fields. Approximately 2 mm of movement would take one entirely out of one part of the visual field, onto the next.

A 2 mm by 2 mm block is hence considered sufficient to cover a given part of the visual field, and is considered standalone unit, Fig. 2.8. The 2 mm by 2 mm dimension is as it pertains to layer 3 of area 17, one of several areas identified in the visual cortex. It could be 4 mm by 4 mm in layers 5 and 6 and 0.2 mm by 0.2 mm in layer 4C. This standalone unit is called the Ice-Cube model of the cortex.
Figure 2.7: The first figure shows the drift of receptive fields across 3 mm in cat cortex. Only 4 or 5 receptive fields recorded in the first tenth of each millimeter is shown. The second figure shows the growth in receptive field sizes with increased eccentricity from the fovea, in a macaque monkey, from Hubel [26].

Figure 2.8: ‘Ice-Cube model’ of the cortex. The figure shows the division of the cortex into 2 sets of slabs, one for ocular dominance and one for orientation, from Hubel [26].
2.3 Hough Transform Structure in the Cortex

Suppose $(\rho, \theta)$ parameterization is used. To characterize straight lines, we need 2 factors - 1) Orientation of the line, and 2) Position of the line in the visual field. In Hough Transform space, every curve belonging to a straight line passes through a single point whose co-ordinates give us the orientation and position of the line. If we use accumulators to implement the Hough Transform [66], we have an array with, say, orientation information in degrees along the columns and the position information in terms of distance from a chosen origin along the rows. On reviewing what we know of the visual cortex structure, we can speculate that the on-center/off-center cells act as units of addition/accumulation. When any such cell is stimulated by the presence of a point in the visual field, it conveys that information to the simple cell it is connected to, effectively doing a +1 to that particular simple cell. As more on-center cells connected to a simple cell is activated, we detect a line of a particular orientation.

The simple cells are our gradient detector units, as they are specific in the orientation they respond to. As a group of simple cells contribute to the complex cell, and the complex cell receptive fields cover only a certain portion of the visual field, we speculate that the complex cells act as the location detector for the line. That is, we can determine the location of the line by looking at the receptive field location of the complex cell that responds to that particular line.

The grouping of cells into orientation columns and the observed receptive field drift across the cortex adds weight to our accumulator array hypothesis. The orientation columns can indicate the $\theta$ of the line, and the receptive field positions can indicate the $\rho$ of the line, as was seen in the Hough Transform. We also know that the ocular dominance columns lie in a direction orthogonal to the orientation columns. Since we
need information from both eyes to accurately process the information from any part of
the visual field, it would fit in with our hypothesis of an accumulator array like structure
in the brain.
Chapter 3

Hebbian Learning and Self Organizing Networks

In this chapter, we briefly examine the structure and signalling operations of a neuron and learn how two neurons interact. We then look at the learning rules which have been proposed to model such interactions between groups of neurons. All images in this chapter are from [7] unless otherwise stated.

Figure 3.1: Structure of a neuron [7]
The structure of a neuron is shown in Fig. 3.1. The neuron consists of a cell body, dendrites, and an axon. The dendrites and the axon are structures arising from the cell body which are used to communicate with other neurons. Signalling happens in a neuron by variation of its membrane potential. The inside of a neuron cell membrane is about -70 mV relative to the outside. $[Na^+]$ and $[Cl^-]$ are higher outside, while $[K^+]$ is higher inside. The cell membrane is a lipid bilayer impermeable to these charged ions, but with embedded ionic channels to allow the ions to flow in or out. Ionic pumps along the membrane strive to maintain the equilibrium potential of -70 mV. For further details of their operations, refer [10, 67, 2].

Figure 3.2: Activity at a Synapse [7]
Interactions between neurons happen at the junction where axons from one neuron interact with dendrites or the cell body of another neuron. Such junctions are called synapses. Synapses can be excitatory or inhibitory. An incoming spike along the axon of a neuron, called ‘presynaptic cell’, causes the release of neurotransmitters into the synapse. The neurotransmitters bind to the ion channel receptors of the dendrites of the second neuron, called the ‘postsynaptic cell’. The ion channels in the second neuron then open, causing either depolarization or hyperpolarization, depending on whether the synapse was excitatory or inhibitory, Fig. 3.2.

The Hodgkin-Huxley equations are popularly used to mathematically describe the effect of an incoming spike on a neuron. Hodgkin and Huxley models the neuron as a RC circuit [67]. Other models like the Integrate and Fire model and its variations are also used to describe neuronal activity. The neuron models found in artificial neural network literature can be derived from the biological neuron models after making a number of simplifying assumptions.

### 3.1 Hebbian Learning

Neurons learn by means of synaptic plasticity. Synaptic plasticity is the ability of synapses to strengthen or weaken over time. A basic mechanism for modification of synaptic strengths was given by Hebb in his seminal work [24]. He postulated that if a presynaptic cell repeatedly and persistently stimulated a postsynaptic cell, it would lead to an increase the efficacy of the synapse between the two cells.

Hebbian learning as defined for linear neurons can be formalized as follows: Suppose we have a neuron with $N$ inputs. If $y$ represents the output of the neuron and $w_i$ represents the weight from input $x_i$ to the neuron:
Figure 3.3: Oja’s rule. Effective input at, say node 1, is now $x_1 - yw_1$.

$$y = \sum_i w_i x_i$$

During training, various input samples are presented to the neuron. If the output of the neuron is high when a particular input is on, the weight between that input and the neuron is strengthened. The following equation is used for weight update, where $\alpha$ is the learning parameter:

$$\Delta w_i = \alpha y x_i$$

In the absence of constraints in Hebbian learning, the weights tend to grow without bounds. Decay terms were hence introduced in the learning rule itself [44, 69, 47, 35, 33]. For a detailed treatment of the type of constraints in Hebbian learning, see [45].

Oja’s rule [47] for a single neuron introduces a decay term in the weight update rule, derived by normalizing the incoming weights to a neuron to have an Euclidean norm of unity. Oja’s rule ends up extracting the largest principal component of the data. The weight update rule now looks like:

$$\Delta w_i = \alpha (y x_i - y^2 w_i)$$
Oja’s *Subspace Algorithm* [48] extended the above rule to $N$ output neurons. The weights then do not necessarily converge to the actual Principal Components of the input space, but they do converge to an orthonormal basis of the Principal Component Space. The weight update rule for the weight $w_{ij}$ between output neuron $y_i$ and input neuron $x_j$ now looks like:

$$\Delta w_{ij} = \alpha(y_i x_j - y_i \sum_k w_{kj} y_k)$$

Asymmetry was introduced to the Subspace Algorithm in the form of a parameter $\theta$ in order to force the weights to converge to the Principal Components [49, 52, 53]. The weight update equation now looks like:

$$\Delta w_{ij} = \alpha(y_i x_j - \theta_i y_i \sum_k w_{kj} y_k)$$

With $\theta_1 < \theta_2 < \theta_3 \cdots$, the weight vectors will converge to the principal eigenvectors in the order of their eigenvalues. The value of $\theta_i$ should be close to 1 to ensure convergence.

Nonlinear extensions to the above rule and its analysis can be found in [51] and [50]. A derivation of the nonlinear PCA rule from a error minimization and feature extraction approach is given in [32].

Sanger developed the *Generalized Hebbian Algorithm* [61] which also found the Principal Components of the input space, but introduced asymmetry in the feedback term. The weight update equations are then:

$$\Delta w_{ij} = \alpha(y_i x_j - y_i \sum_{k=1}^{\text{i}} w_{kj} y_k)$$

Weights between neurons in the same layer are often modeled as having inhibitory properties. *Anti-Hebbian* learning as a method to decorrelate outputs from neighbouring neurons is discussed in [19, 17, 18]. It has been used in conjunction with Hebbian learning.
Figure 3.4: Anti-Hebbian Rule leads to decorrelation in output neurons: $\Delta w_{12} \propto -y_1 y_2$ for feature extraction [16]. The simplest weight update rule for the weight $w_{ij}$ between output neuron $y_i$ and input neuron $y_j$ would then be:

$$\Delta w_{ij} = -\alpha y_j y_i$$

### 3.2 Self Organizing Maps

*Competitive learning, or winner take all algorithms*, are another group of algorithms that model relationships between neurons in the same layer. They form a powerful tool for feature discovery and leads to self organization of networks [60, 44, 69, 35]. They rely on competition, cooperation and synaptic adaptation amongst the neurons in the network [23]. Consider Kohonen’s *Self Organizing Map* or SOM: The neuron declared as the winner $i^*$ in any iteration is the one whose weights $w_j$ resemble the input $x$ at that iteration. That is:

$$i^* = \arg\min_j (\|x - w_j\|)$$

A neighbourhood function which peaks at the location of the winner neuron is then defined. A Gaussian function is a good choice:
\[ h_j = \exp\left(-\frac{\|i^* - j\|^2}{2\sigma^2}\right) \]

When the weights associated with any particular neuron are updated, the neighbourhood function’s value at that particular neuron’s location is multiplied with the learning parameter:

\[ \Delta w_j = \alpha h_j (x - w_j) \]

At the end of the learning process, the weight space of the neurons resembles the input space.
Chapter 4

Architecture and Methods

The key ideas addressed by this thesis can be stated as follows:

- **Hypothesis 1**: The human visual system may have an accumulator-array structure for detection of straight lines in the primary visual cortex.

- **Hypothesis 2**: The accumulator-array structure can be learned.

Hypothesis 1 was addressed in Section 2.3. The proposed network of neurons that leads to an accumulator-like computation in the cortex is described later in this section. The architecture and the learning rules detailed next lead to $\rho$-$\theta$ parameterization of straight lines as described in Chapter 1.

The presence of orientation-sensitive simple cells and complex cells in the cortex has been examined in Chapter 2. Visual stimuli travelling from the retina through the lateral geniculate nucleus encounter these cells on reaching the cortex. Groups of orientation sensitive cells located close together, which respond maximally to stimuli of a particular orientation, are called *orientation columns*. Our interest lies in the orientation columns as well as the next layer of neurons which receive their inputs from the orientation columns
as shown in Fig. 4.1.

Suppose we are interested in 4 of all the possible orientations - \( \theta_1, \theta_2, \theta_3, \theta_4 \). When a point in the visual field is illuminated, all 4 of the orientation sensitive cells at the point receive stimulus. They feed into all of the corresponding cells in the accumulator array. When a line of a given orientation falls in the visual field, a collinear set of points gets excited. The cells in the corresponding orientation column responds strongly. As points along a line of given orientation have the same \( \rho \), they all excite a single accumulator cell very strongly. This can be represented by the following equation where \( A(\rho, \theta) \) is the accumulator cell, and \( N \) is the number of points along the line:

\[
A(\rho, \theta) = \sum_{i=1}^{N} \delta(\rho - (x_i \cos(\theta) + y_i \sin(\theta)))
\] (4.1)
Figure 4.2: Orientation sensitive cells at every pixel location in the image plane. Let $\theta_1$ correspond to the horizontal line. When two adjacent points along the horizontal direction are excited, their $\theta_1$ sensitive cells end up exciting the same accumulator cell. $\rho_1$ represents the distance of the horizontal line from the origin.
Figure 4.3: Connections between pixels in the image plane and the cells in the accumulator. For clarity, only the connections which give rise to the two strongest responses in the accumulator array are shown.

This is demonstrated in Fig. 4.2 for the case where $\theta_1$ corresponds to the horizontal line.

The case with multiple lines is demonstrated in Fig. 4.3. The $\theta_2$ receptors of the points lying along the red line feed into a single cell in the $\theta_2$ column. The $\theta_1$ receptors of the points lying along the blue line feed into a single cell in the $\theta_1$ column. The point at the junction of the two lines feed into both columns.

The network described clearly lead to an accumulator array. The important question is: can such a network be learned? We next show how the neurons that receive their inputs from the orientation sensitive cells could learn the position of the given line. That is, we are interested in learning the connections from the $\theta$ sensitive cells to the corresponding $\rho$ sensitive cells, as shown in previous figures. In the case of the stimuli consisting of straight lines, this combination of orientation and position amounts to an accumulator array. The accumulator array in turn leads to an efficient representation of the input space, and makes the identification of inputs possible even in presence of occlusion and
4.0.1 Network Architecture

The network of interest consists of two groups of neurons, both arranged in 2-dimensional lattices - let us call them the input array and the output array. An example is shown in Fig. 4.4. The input array neurons receive signals from the input image, and also from neighboring neurons within the same array. These cells represent the orientation-sensitive simple cells in the cortex.

The output array neurons receive signals from the input array as well as from other neurons within the same array. The weights from the input array to the output array correspond to the mapping from Cartesian space to Hough space as shown in Fig. 1.1b. Every output neuron is fully connected to a neighborhood of underlying neurons from the clutter.

Figure 4.4: Example Architecture
input array as well as to neighboring neurons in the same array. They represent what will be the position sensitive cells, after training. The neuron with the maximum output will indicate the location of the input in the visual field, and will act as the input to deeper arrays in the cortex. For example, after training, when a test image with a horizontal line at the second row is given as input to the network, the maximum in the output array will occur at the neuron at the center of the second row. Assuming the origin to lie in the center of the output array, the neuron with the maximum indicates the shortest distance from the origin, which is also $\rho$.

The intra-array connection strengths are assumed to be symmetric - that is, the strength of the connection going from neuron $a$ to $b$ within the same array is assumed to be the same as that going from neuron $b$ to $a$. No self feedback is assumed.

The intra-array neighborhood of a neuron is defined as the group of neurons within a specified distance, when the neurons are arranged in a 2-dimensional lattice. Similarly, for inter-array connections, the neighborhood is the group of neurons from the input array that connect to an output neuron, when the input and output lattices are placed center to center and an equal number of input neurons are assumed to connect to every output neurons.

We build a network as shown in Fig. 4.4 for every orientation under consideration. After training, the output array of any given network will indicate the position of a line of corresponding orientation. This leads to an accumulator array like structure.

### 4.0.2 Dynamic Network Equations

Oja’s rule [47] is used to modify the weights between the input and output array. Ideas borrowed from Kohonen [33, 36] and Foldiak [19, 17, 18] are used to modify weights
between neurons in the same array.

Let \( x \) denote the external input, \( y \) denote the input neuron array and \( z \) denote the output neuron array. The output at each array is computed as follows:

\[
y_i(t) = f\left(\sum_j w_{ij} y_j(t - \Delta t) + x_i\right)
\]

\[
z_i(t) = f\left(\sum_j \hat{w}_{ij} z_j(t - \Delta t) + \sum_j \bar{w}_{ij} y_i\right)
\] (4.2)

In (4.2), \( f \) refers to a smooth thresholding function like the sigmoid or the hyperbolic tangent. \( \Delta t \) indicates increments in time. The neuron outputs decay as follows, where \( \text{decay} < 1 \):

\[
y_i(t + \Delta t) = \text{decay} \times y_i(t)
\]

\[
z_i(t + \Delta t) = \text{decay} \times z_i(t)
\] (4.3)

The decay parameter was chosen empirically.

While updating weights, the aim is to strengthen the connections between neurons that fire together, while resulting in a stable network. For weights between the input and output arrays, Oja’s rule is used. For neurons in the same array, the weights increase if the outputs of the neurons correlate, but with a restraining factor, \( \beta \).

The weights between the input and output array are updated as follows, where \( \alpha \) is the learning parameter:

\[
\bar{w}_{ij} = \bar{w}_{ij} + \alpha z_i(t)(y_j(t) - z_i(t)\bar{w}_{ij})
\] (4.4)

The weights between the neurons in the output array are updated as follows, where \( \beta \) is a constant:
The weights between neurons in the input array are similarly updated. The learning parameter decreases with time in an exponential manner. That is, if $\alpha_o$ denotes the initial value of the learning parameter:

$$\alpha(t) = \alpha_o \exp\left(-\frac{t}{\tau}\right)$$  \hspace{1cm} (4.6)

The input array is assumed to have the same size as that of the incoming input signal. The output array has a reduced size along one dimension - if the array is of size $n \times m$, $m << n$. This is noticeable in Fig. 4.7-4.11. The reduced size for the output array is justified as follows: Suppose an output array the same size as that of the input array is used. After training with horizontal lines, for example, it can be found that only a few neurons in each row of the output array are needed to represent the input samples. We can hence do away with some of the columns in the output array, thereby reducing the dimension of the output array.

Such a two-array network was built for every orientation of interest, Fig. 4.6. Each network was trained separately. The training samples consist of lines of the chosen orientation, located at all possible locations in the input image. The first array learned to become orientation sensitive, by means of its intra-array connections, resulting in the $\theta$ value. The second array learned to denote the $\rho$ for the corresponding $\theta$ from the first array. Taken together, this results in the $\rho$-$\theta$ architecture described previously.

This network differs from previous works [33, 17, 21] in that neurons within the same array are not assumed to inhibit each other. Our premise is that of Hebb: if two neurons within the same array fire together repeatedly, the firing of one of them alone should
cause some excitation in the other.

Figure 4.5: Winner computation example. Suppose we have an array of neurons. Let the outputs for neurons 1, 2, and 6 be 0.1, 0.7, and 0.75 respectively. All other neuron outputs are presumed 0. Conventional wisdom would declare neuron 6 as the winner. In our computation, the neuron outputs are multiplied by a Gaussian centered at the neuron locations and added together. The resulting maximum is now located at neuron 2. Neuron 2 is declared the winner.

After training, the networks were tested using images of lines as well as line segments under various conditions. The peak point or the winning neuron in the output array was determined as follows: every neuron output is multiplied by a two dimensional Gaussian function, and are then summed together. The neuron located at the location of the maximum value is declared the winner, see Fig. 4.5. The winning index in the output array \( i^* \) is determined by:

\[
g_i(t) = \sum_j z_j(t) \tilde{r}_{ij}(t) \\
i^*(t) = \arg\max_i (g_i(t))
\]  

(4.7)
Figure 4.6: Resulting network, after training

The $\tilde{r}_{ij}$ term represents a 2 dimensional Gaussian neighborhood. This new winning neighborhood approach leads to more robust results when compared to the conventional approach of declaring the neuron with maximum output as winner.

4.1 Results

4.1.1 Network Training

The incoming weights to the output array neurons were initialized to small random numbers normalized to 1. The weights between neurons in the same array were initialized to 0. The starting value of the learning parameter $\alpha$ was 0.1 and the decay parameter was 0.5. These parameters were determined empirically. Every output array neuron received inputs from a 4x4 array of neighbouring input array neurons. The input space consisted of
images with lines of given orientation located at all possible positions, one line in any given input. The network was trained with the images chosen at random, one array at a time - that is, the input array training was completed before the output array was started. The number of iterations were to the order of about 5000 for every array. The choice of the number of iterations was driven by the rate of decay of the learning parameter, equation (4.6) - once the learning parameter reached a point where the neurons had little effect on each other, training was stopped.

Fig. 4.7 and Fig. 4.9 show the results obtained from a network trained with 135° lines. In Fig. 4.7, a diagonal line was given as the input. The same line was represented in the input array. We obtained the maximum in the input array at the centroid of the line, and
the maximum in the output array at the row corresponding to the line. In Fig. 4.9, the same diagonal line, but partially occluded, was given as the input. The input array filled up the occluded portion. The maximum in the input array is now shifted as the input line is incomplete. However, we continue to obtain the same maximum in the output array as the full line, because the output array managed to identify the full line from the segments. In Fig. 4.8, we see a shifted diagonal line. The input array values, as well as the positions of maximum values in the input array and output are shifted as well.

Fig. 4.10 shows the results from a network trained with horizontal line segments. When a particular line segment was presented, the input array filled up part of the full line. The maximum in the input array and the output array both correspond to the particular line segment.
Fig. 4.11 are the results obtained when a network trained with diagonal lines was presented with an image with clutter. The input layer spread out all of the line segments along the diagonal direction. As there were aligned diagonal line segments along the origin, the maximal output corresponded to a diagonal line along the origin.

### 4.1.2 Performance Comparison

Next compared are the results from 3 different implementations of the Hough Transform, and the neural network trained on straight lines. The 3 implementations of the Hough Transform are -

1. The Conventional Hough Transform (CT) [see Algorithm 1]
2. The Accumulator Array implementation of the Hough Transform (AT) [see Algorithm 2]

3. A Hough Transform implementation using Gabor Filters (GT) [see Algorithm 3]

CT and AT have been discussed already - the only difference between them is that, in AT, the connections from the image pixels to the accumulator array are precomputed and weighted in order to simulate the behaviour of neurons.

In GT, a Gabor filter for every $\theta$ is applied at every pixel. Equation (4.8) represents the operation of a Gabor filter, where $x$ and $y$ refers to the coordinates in the $x$-$y$ plane and $\theta$ is the orientation of interest, reproduced here from Chapter 2:
Algorithm 1 Conventional Hough Transform, for an Image $\mathcal{I}$ of size $\mathcal{M} \times \mathcal{N}$.

\begin{algorithm}
\begin{algorithmic}
\FOR{$\theta = 0$ \text{ to } $\pi$}
  \FOR{$x = 1$ \text{ to } $\mathcal{M}$}
    \FOR{$y = 1$ \text{ to } $\mathcal{N}$}
      $\rho = x \cos(\theta) + y \sin(\theta)$
      $A(\rho, \theta) = A(\rho, \theta) + \mathcal{I}(x,y)$
    \ENDFOR
  \ENDFOR
\ENDFOR
\end{algorithmic}
\end{algorithm}

\begin{equation}
\begin{align*}
x' &= x \cos(\theta) + y \sin(\theta) \\
y' &= -x \sin(\theta) + y \cos(\theta) \\
gf(x, y, \theta) &= \exp\left(-\frac{(x')^2 + \gamma^2(y')^2}{2\sigma^2}\right) \cos\left(\frac{2\pi x'}{\lambda}\right)
\end{align*}
\end{equation}

Figure 4.11: Network trained of diagonal line, presented with a clutter.
Algorithm 2 Accumulator Hough Transform, for an Image $\mathcal{I}$ of size $\mathcal{M} \times \mathcal{N}$.

\begin{algorithm}
\begin{algorithmic}
\FOR{$\theta_o = 0$ \textbf{to} $\pi$}
\FOR{$x = 1$ \textbf{to} $\mathcal{M}$}
\FOR{$y = 1$ \textbf{to} $\mathcal{N}$}
$\rho_o = x \cos(\theta_o) + y \sin(\theta_o)$
\FOR{$\rho_i = \rho_o - 3$ \textbf{to} $\rho_o + 3$}
\FOR{$\theta_i = \theta_o - 3$ \textbf{to} $\theta_o + 3$}
$\mathcal{A}(\rho_i, \theta_i) = \mathcal{A}(\rho_i, \theta_i) + \mathcal{I}(x,y) e^{-((\rho_i-\rho_o)^2+(\theta_i-\theta_o)^2)}$
\ENDFOR
\ENDFOR
\ENDFOR
\ENDFOR
\ENDFOR
\end{algorithmic}
\end{algorithm}

Algorithm 3 Gabor Filter Hough Transform, for an Image $\mathcal{I}$ of size $\mathcal{M} \times \mathcal{N}$.

\begin{algorithm}
\begin{algorithmic}
\FOR{$\theta = 0$ \textbf{to} $\pi$}
\FOR{$x = 1$ \textbf{to} $\mathcal{M}$}
\FOR{$y = 1$ \textbf{to} $\mathcal{N}$}
$\rho = x \cos(\theta) + y \sin(\theta)$
$\mathcal{A}(\rho, \theta) = \mathcal{A}(\rho, \theta) + \mathcal{I}(x,y) g_f(x,y,\theta)$
\ENDFOR
\ENDFOR
\ENDFOR
\end{algorithmic}
\end{algorithm}

The value from the Gabor filter is then used to increment the corresponding cell in the accumulator array. The following values were used in (4.8): $\sigma = 1, \gamma = 1, \lambda = 4$.

In all three implementations of the Hough Transform, sampling interval for $\theta$ is $15^\circ$, while for the neural network, it is $45^\circ$. All possible $\rho$ values are covered.

Three conditions are tested:

1) Difference in $\theta$
2) Difference in $\rho$, and
3) Presence of Noise.

The parameter being plotted is the difference between the highest peak and the second highest peak in the accumulator, as explained in Fig. 4.12. Sample test images are shown in Figs. 4.13, 4.14, and 4.15.
Figure 4.12: Separation of peaks example. When two line segments are aligned, only one peak is observed in the accumulator. As the line segments separate, the peaks become more distinct. In the case of addition of noise, more peaks of similar amplitude appear in the accumulator as the standard deviation of the added noise increases. As the peaks become more distinct, the difference between the highest peak in the accumulator and the second highest peak in the accumulator diminishes.

For testing the difference in $\theta$, we presented images with two line segments, which were collinear at first. The angle between the two line segments were gradually increased. The difference between the highest and the second highest value in the accumulator was plotted. The difference gradually decreased as the lines became more distinct. Fig. 4.16 show the results. The curves follow a similar trend in all cases.

For testing the difference in $\rho$, we presented images with two line segments, which were aligned at first. One of the line segments was then displaced. The difference between the highest and the second highest value in the accumulator was plotted. The difference gradually decreased as the lines became more distinct. Fig. 4.17 show the results.

For testing the performance in the presence of noise, additive white Gaussian noise was added to the test images with two aligned line segments. The standard deviation of
Figure 4.13: Variation in angle test images. The angle between the two line segments increases as we move from (a) to (d).

Figure 4.14: Variation in distance test images. The separation in pixels between the two lines increases from (a) to (d).
the noise was increased. As standard deviation was increased from zero, multiple peaks appeared in the accumulator array for all 4 implementations. The results are shown in Fig. 4.18.

Another question to ask is: when does the accumulator claim that there are 2 lines present, instead of 1? What is the breakpoint after which the second highest peak is at least 75% of the highest peak? Plots with these results are shown for CT and GT Fig. 5.2.

4.1.3 Extension to All Orientations

The architecture outlined can be extended to detect other orientations in two ways -

1. Build and train separate networks for lines of separate orientations, or

2. Estimate the “actual” orientation from the peak outputs of the 4 networks already trained.

The first approach is straightforward, but time consuming. In the second approach, the outputs of 4 networks already trained are considered. The networks were trained
Figure 4.16: Decay of difference in peaks with difference in angle between two line segments for the 4 different implementations.
Figure 4.17: Decay of difference in peaks with difference in distance between two line segments for the 4 different implementations. The same values were obtained for Accumulator Array Implementation of the Hough Transform and the Conventional Hough Transform, and hence AT plot is omitted.
Figure 4.18: Decay of difference in peaks with addition of noise for the 4 different implementations. The same values were obtained for Accumulator Array Implementation of the Hough Transform and the Conventional Hough Transform, and hence AT plot is omitted.
Figure 4.19: Plots showing the point where the accumulator claimed 2 or more lines were present.
separately with lines of the following orientations - 0°, 45°, 90°, and 135°. The outputs of these networks (after training), for each of the 4 orientations, are then considered.

As expected, each network responds maximally to the orientation with which it was trained, Fig. 4.20a. For example, after training, the 0° network responds maximally to a horizontal line, while the 90° responds minimally to it. A curve is then fitted to pass through the output values of each of the networks, for each of the orientations, using a standard interpolation algorithm. We used an inbuilt MATLAB interpolation algorithm using the Fast Fourier Transform (FFT). The points are transformed to the Fourier domain using FFT, and then transformed back to obtain more points.

Four curves are obtained after the interpolation process. These 4 curves are all centered, Fig. 4.20b and averaged, Fig. 4.20c, to obtain a single curve which captures the variation in outputs from each of the trained networks. Let this curve be called the “output” curve, \( f(\theta) \), with a peak at \( \theta = 90° \). The curve is then shifted to peak at \( \theta = 0° \).

A test line (of any orientation) is then presented to the 4 networks, and their outputs are considered. Once again, a curve is fitted through the 4 outputs. Let the resulting curve be \( g(\theta) \), where \( \theta \) varies from 0 to \( \pi \). The “output” curve is made to slide over the resulting curve (much like in a convolution), and the square of the error between both curves are computed. The angle corresponding to the location of minimum error is then declared as the angle of the test line:

\[
\phi^* = \arg\min_{\phi} \left( \sum_{\theta} (f(\theta) - g(\phi - \theta))^2 \right)
\]  

The approach outlined above gave us the correct angle in the all of the test lines (with orientations varying from 0 to \( \pi \)) considered.
Figure 4.20: Extension to all orientations by curve estimation.

(a) Curves obtained for the four angles.

(b) The four curves centered.

(c) Single curve after averaging.
4.2 Analysis

In this section, a mathematical analysis of the network in Fig. 4.4 is provided. As discussed before, connections are present between neurons in the same layer as well as between neurons in different layers. The connections between neurons in different layers are governed by Oja’s rule, as seen in equation (4.4). Oja’s rule has been analyzed in detail in literature previously, see Chapter 3.

A derivation of the equation (4.5), governing connections between neurons in the same layer is provided in this section. A linear version of the network is considered for ease of notation. Similar analysis can be extended to the nonlinear version. Information-theoretic constructs, similar to the one derived here, have been provided for the Anti-Hebb rule in [55, 38].
4.2.1 Maximization of Mutual Information, Under Constraint.

Consider the architecture in Fig. 4.21. Let $x$ be the vector of input samples, and $y$ be the vector of output samples, with real continuous values. If $V$ represents the feedback weights between the output neurons, we can write:

$$y = x + Vy$$  \hspace{1cm} (4.10)

It has to be noted that $V$ is symmetric and that $\text{Tr}(V) = 0$. That is, there is no self feedback in the output neurons and that the connections are symmetric. The above can be rewritten as follows, where $W$ represents a weight matrix between input and output:

$$y = (I - V)^{-1}x = Wx$$  \hspace{1cm} (4.11)

$W$ can be thought of as a mapping from $x \in \mathbb{R}^N$ to $y \in \mathbb{R}^N$, where $N$ is the dimension of the vector space.

Consider the case where the presented inputs have a multivariate Gaussian probability density function with zero mean. If $N$ is the dimension of $x$ and $\sum_x = \mathbb{E}(xx^T)$ denotes the covariance matrix of $x$, the we can write:

$$P(x) = \frac{1}{(2\pi)^{N/2}(|\sum_x |^{1/2})} \exp\left(-\frac{1}{2}x^T \sum_x^{-1} x\right)$$  \hspace{1cm} (4.12)

Now consider the output unit $y_1$, element of the vector $y$. We can write:

$$y_1 = x_1 + \sum_{i=2}^N v_{1i}y_i$$  \hspace{1cm} (4.13)

The mutual information between $y_1$ and another output, say $y_2$, can be written as:
\[ I(y_1, y_2) = \mathbb{E} \left( \ln \left( \frac{P(y_1, y_2)}{P(y_1)P(y_2)} \right) \right) \quad (4.14) \]

\( \mathbb{E} \) represents the expectation operator. With the given probability density function for the inputs, with column vector \( y_{12} = [y_1 \ y_2]^T \) and \( \sum_{y_1,y_2} \) the covariance submatrix for \( y_1 \) and \( y_2 \), we can write:

\[ P(y_1, y_2) = \frac{1}{2\pi \left( |\sum_{y_1,y_2}|^{1/2} \right)} \exp \left( -\frac{1}{2} y_{12}^T \sum_{y_1,y_2}^{-1} y_{12} \right) \quad (4.15) \]

\[ P(y_1) = \frac{1}{(2\pi \sigma_{y_1}^2)^{1/2}} \exp \left( -\frac{1}{2} y_1^2 \sigma_{y_1}^{-2} \right) \]

\[ P(y_2) = \frac{1}{(2\pi \sigma_{y_2}^2)^{1/2}} \exp \left( -\frac{1}{2} y_2^2 \sigma_{y_2}^{-2} \right) \]

Consider the case where all of the connecting weights to \( y_1 \) are constrained to have a norm of 1. That is, \( \sqrt{\sum_{i=2}^N v_{1i}^2} = 1 \). Using the principle of Lagrange multipliers, we can construct the following cost function:

\[ \mathcal{E} = I(y_1, y_2) - \lambda \left( \sqrt{\sum_{i=2}^N v_{1i}^2} - 1 \right) \quad (4.16) \]

Next, we differentiate the cost function with respect to the weight between \( y_1 \) and \( y_2 \), \( v_{12} \). We use the following results for the same:

\[ \frac{\partial y_1}{\partial v_{12}} = y_2, \quad \frac{\partial y_2}{\partial v_{12}} = y_1, \quad \frac{\partial \sigma_{y_1}^2}{\partial v_{12}} = 2\sigma_{y_1,y_2}^2, \quad \frac{\partial \sigma_{y_2}^2}{\partial v_{12}} = 2\sigma_{y_1,y_2}^2, \quad \frac{\partial \sigma_{y_1,y_2}^2}{\partial v_{12}} = \sigma_{y_1}^2 + \sigma_{y_2}^2 \quad (4.17) \]

In the above equation, \( \sigma_{y_1}^2 \) is the variance of \( y_1 \) and \( \sigma_{y_2}^2 \) is the variance of \( y_2 \) and \( \sigma_{y_1,y_2}^2 \) is the covariance of \( y_1 \) and \( y_2 \).

On differentiating the cost function, we obtain:
\[
\frac{\partial \mathcal{E}}{\partial v_{12}} = y_1 y_2 \left( \frac{1}{\sigma_{y_1}^2} + \frac{1}{\sigma_{y_2}^2} \right) + \frac{\sigma_{y_1,y_2}^2}{\sigma_{y_1}^2} \left( 1 - \frac{y_1^2}{\sigma_{y_1}^2} \right) + \frac{\sigma_{y_1,y_2}^2}{\sigma_{y_2}^2} \left( 1 - \frac{y_2^2}{\sigma_{y_2}^2} \right) - \lambda v_{12} \quad (4.18)
\]

For slowly varying weight \( v_{12} \), the average of the second and the third term vanishes. So we are left with the term proportional to \( y_1 y_2 \) and the term arising from the constraint. If we were to do a gradient ascent on the weight, we would end up with the weight update equation:

\[
\Delta v_{12} = \alpha (y_1 y_2 - \lambda v_{12}) \quad (4.19)
\]

This is the same as equation (4.5).

### 4.2.2 Experimental Results

A single layer neural network with the architecture and weight update equations as discussed in the previous section was trained. The training samples were the images of shapes shown in first column of Fig. 4.22. The starting value for \( \alpha \) was 0.1. The value for \( \lambda \) was 0.01. The images were given to the network in random order for 3000 iterations.

After training, incomplete test shapes, shown in the second column, were given as input to the network. The output of the network, after taking the absolute value and thresholding for display purposes, are shown in the third column. As can be noted, the network managed to successfully retrieve the incomplete portion of the presented shape. This is similar to Kohonen’s Novelty Filter, discussed in [33], but attempts to increase the mutual information between neurons, rather than decorrelate them.
Figure 4.22: The training images are the complete shapes in the first column. The second column shows the test images. The third column shows the retrieved images.
4.3 Discussion

We have seen how unsupervised neural networks can construct accumulators with the help of a few learning rules. Once the accumulator structure is in place, the network can rapidly identify any given line by its orientation and position. The network can identify the line even in the presence of occlusion and moderate amounts of noise, by looking at the winning cell which has the maximum output value in the presence of a given input.

The network is also comparable to the different implementations of the Hough Transform in terms of the ability to distinguish separate lines. When the angle between two line segments was increased, the difference between accumulator peaks gradually decreased to zero, thereby identifying the lines as separate - the curve of the graphs obtained were similar for the network and various Hough Transform implementations. The same trend was observed when the distance between two line segments was increased, as well as when noise was added to the test images.

The network can be further extended by including more arrays to enable it to identify curves or more complex shapes. This hierarchy of accumulator cells would result in having a cell or group of cells respond maximally to a frequently encountered input, leading to a sparse representation of input data. Experimental results supporting this idea for visual objects have appeared in literature [37, 57].

Log-Polar Transform:

An accumulator like structure could potentially explain the phenomenon of log polar transformation observed in the cortex [62, 8]. For this discussion, assume that the center of an input image acts as the origin, and that the origin corresponds to the fovea. Suppose there is a straight line in the input image. Every point along that line will increment a set of cells in the accumulator array, and a maximum is obtained in a single cell or set
of cells. The cell or group of cells with the maximum value would indicate the position, or the distance of the line from the origin, \( \rho \).

Now consider a particular point \( p \) in the input image. Let the point be at a distance \( d \) from the origin. Any line which passes through \( p \) will have a \( \rho \) which is less than or equal to \( d \), i.e., \( \rho \leq d \). Since \( p \) will contribute to the accumulator array every time a line passes through \( p \), the cells excited by point \( p \) will indicate lines with \( \rho \) from 0 to \( d \). That is, \( p \) should excite a total of \( d \) number of cells in the accumulator array. Since a point closer to the origin will have a \( d \) that is smaller than that of a point which is farther away from the origin, this in turn implies that a point closer to the origin will excite fewer cells when compared to a point farther away from the origin. Hence all points which lie at the same distance from the origin will excite the same number of cells in the accumulator. This is in agreement with the log polar transform idea.

**Pin-wheel Model:**

The ice-cube model of the cortex was described previously in Chapter 2. Another prevalent model of the cortex is the pin-wheel model [13]. In the pin-wheel model, ori-
entation sensitive cells are arranged in a pin-wheel like fashion around a center. While traversing the pin-wheel structure, either in clockwise or in anti-clockwise direction, the orientations to which the encountered cells are sensitive to, vary smoothly.

The network presented in this chapter fits into this construct as well. Consider the second array of the 4 trained networks. When overlapped with centers coinciding, they present a pin-wheel like structure, see Fig. 4.23. The orientations to which the cells are sensitive to vary smoothly in both clockwise and counter-clockwise direction in the structure.

Next, some of the other neural network implementations that were explored are outlined.

4.3.1 Supervised Learning

Supervised learning techniques were initially investigated for learning the accumulator array like structure. The supervised learning techniques tried were: 1) Backpropagation network 2) Radial Basis Function network.

1) Backpropagation Network: Consider the network in Fig. 4.24. The input image, \(N \times N\) in dimension, is fully connected to the output layer, \(N \times 1\) in dimension. The weight matrix is \(N \times N \times N\) in dimension. The inputs and expected outputs are defined as following: Suppose only horizontal lines are considered. Since the input image is \(N \times N\), there are a total of \(N\) distinct horizontal lines. For each of those \(N\) horizontal lines, a particular cell from the \(N \times 1\) output layer should turn on, while the rest of the cells remain in off position. For example, if there is a line in row 1 in the input image, cell 1 from the output layer would output 1, while the rest of the cells would output 0.

Results: The network was trained using horizontal lines at random positions. When
N=8, with 15000 iterations, the network learned to output the correct cell, for each of the horizontal lines. For N=16, with a learning parameter little higher than that for N=8, the network learned the correct output, with iterations close to 25000. For N=32, even with different learning parameters and iterations as high as 100,000, the network showed only very little reduction in its error, and the complexity made such type of learning infeasible. So the number of input samples were increased to include vertical, diagonal, and anti-diagonal lines. The network was trained to give zero output if any line other than a horizontal line appeared. The network learned the correct output for N=32 as well N=64, for iterations close to 50000.

2) Radial Basis Function Network: The network used is shown in Fig. 4.25. The Gaussian function was used as the radial basis function, with the input matrices with a single horizontal line, rearranged as a vector, as centers.

Results: Once again, the network was trained using horizontal lines. The network
learned faster than Backpropagation Network, and learned for higher dimensions with fewer input samples. For N=128, with just 15000 iterations, the network learned all of the outputs correctly.

4.3.2 Self Organizing Maps

An attempt was made to use self organizing map concepts, discussed in section 3, to obtain an accumulator-like structure. A single layer neural network, Nx1 in dimension, was trained using SOM principles. The input images, NxN in dimension, had horizontal lines, one horizontal line in any given input image. Every neuron in the network was fully connected to the input image.

After training, a single neuron responded maximally to any given horizontal line. But the position of the neuron responding maximally to a given line did not correspond to the position of the line, and the same neuron need not respond maximally if the network was trained all over a second time. This is because the weights are initialized to random values and there is no way of predicting which neuron will respond maximally to a given input.
line. Once a neuron responds maximally to a given line, it will be declared the winner, and during training, it is trained to respond to that particular line most strongly. The self organizing map was therefore discarded as a method for building an accumulator.
Chapter 5

Human Experiments

In this chapter, the psychophysical experiments done as part of this research are discussed. The relevant literature is reviewed and the experiment set up is described. This is followed by details of the experiments and the results obtained.

The detection of lines composed of dots, in the presence of random noise consisting of dots, was examined in [68, 65]. In [68], the effects of dot spacing, numerosity, and orientation were examined, and dot spacing was found to be the most important factor influencing the detection of the line. Smits et al. [65] defined a perceptual salience measure for the lines, and found that experimental results matched the value predicted by their model based on Gestalt principles. They also found reduced latencies when the noise elements were farther away from the line, and a preference for horizontal and vertical orientations in the observers.

Beck et al. [4] examined the detection of lines under various conditions - varying sizes, shapes, contrast, and alignment of constituent elements. They found that spatial density and contrast of the elements did not play a significant role in line segregation, while element elongatedness and principal axis orientation did. Detection of contours consisting
of Gabor patches are discussed in [15, 25, 22]. The parameters examined include the relative orientation of path elements, the inter-element distance, length of the contour, and jitter along the contour. They found that with increase in each of the before-mentioned parameters, percentage of correct detection of contours went down for the human observers. Hess and Dakin [25] discusses detection of contours for the periphery as well as fovea, and found that contour integration does not deteriorate significantly beyond 10°. A Bayesian model for human contour detection was discussed in [14].

Results from tracking of curves by human observers are outlined in [73, 72]. They were compared to that from an ideal observer and the human observers were found to be most effective at detecting curves which are straight.

The experiments outlined in this document can be placed in the larger context of visual search. Various parameters like orientation and color influencing detection of targets amongst detractors and their ties to neural mechanisms are discussed in [70, 12, 46].

5.1 Experiment SetUp

A basic outline of the experiment set up is as follows: the human observer is presented with a number of test images, each test image for an interval of a hundred milliseconds. The observer is then asked to answer a couple of questions based on what s/he has seen, as quickly as possible. Before every test image, the observer will be presented with a focus screen, informing the observer where to direct his or her gaze, Fig. 5.1.a. After every test image, a masking image will be shown, Fig. 5.1.b. The display screen is at a distance of 50 cm from the observer. The screen resolution is 1920×1080. The test images will have line segments that may or may not be aligned, with or without noise, lying along a smooth curve or along a straight line. Sample images are similar to Fig. 5.2 and Fig. 5.3.
Figure 5.1: Focus Screen and Mask Image.
(a) Variation in $\theta = 0$ and $\theta = 15$ degrees cases, for horizontal lines.

(b) Variation in $\rho = 0$ and $\rho = 9$ pixels cases, for vertical lines.

(c) Variation in $\sigma = 0$ and $\sigma = 0.6$ cases, for inclined lines.

Figure 5.2: Aligned, Non-Aligned, and Noise Examples.
The program for displaying the images and recording user data was created using QT [56].

Experiment Details:

Experiments 1 - 3: Line Detection in the Fovea

This set of experiments examine how detection of lines in the fovea is affected by orientation of the line segments (Experiments 1-3), misalignment in orientation (variation in $\theta$, Experiment 1), misalignment in distance (variation in $\rho$, Experiment 2), and the presence of noise (variation in noise $\sigma$, Experiment 3). The line segments always appeared within $4^\circ$ of the center of the fovea. The line segments were each 10 pixels long. The orientation of the line segments could be one of 4: $0^\circ$, $45^\circ$, $90^\circ$, and $135^\circ$.

Stimuli:

In Experiment 1, the misalignment between the line segments is in the orientation of the two line segments. The difference in $\theta$ could be one of five - $0^\circ$, $5^\circ$, $10^\circ$, $15^\circ$, and $20^\circ$, see Fig. 5.2.a. For the 4 orientations mentioned before, and 5 differences in $\theta$ for each orientation, 15 test images were generated - amounting to 300 images in total. The observer was asked to indicate whether the line segments were aligned or not, and if they were aligned, to indicate one of the four possible orientations.
In Experiment 2, the misalignment between the line segments is in the placement of the two line segments. The difference in $\rho$ in pixels could be one of five - 0, 3, 6, 9, 12, see Fig. 5.2.b. Once again, 15 test images were generated for 4 orientations and 5 differences in $\rho$ - resulting in 300 images in total. The observer was asked to indicate whether the line segments were aligned or not, and if they were aligned, to indicate one of four possible orientations.

In Experiment 3, the line segments are perfectly aligned in this case, but embedded in noise. The noise added was Gaussian, with 5 possible values of standard deviation, $\sigma$ - 0, 0.2, 0.4, 0.6, 0.8, see Fig. 5.2.c. 15 test images were generated for each of 4 orientations with 5 possible values of $\sigma$, resulting in 300 images. The observer was asked to indicate whether the line segments were detected or not, and if they were detected, to indicate one of four possible orientations.

Experiments 4 - 6: Line Detection in the Periphery

In these experiments, the line segments always appeared within 10° and 20° of the center of the fovea. The line segments were 40 pixels long. The orientation of the line segments could be one of 4 - 0°, 45°, 90°, and 135°. The conditions tested were the same as before: variation in $\theta$, variation in $\rho$, and variation in noise $\sigma$.

Experiment 7: Detection of Contours

In these experiments, lines as well as contours consisting of line segments [22] that are aligned or not aligned, are presented to the observers. The observers are asked if the line segments were aligned or not, and if they were aligned, whether they were aligned along a contour or along a line.

Participants: 15 university undergraduates (6 men, 9 women; age range 18 - 24 years) participated in Experiments 1-3 for course credit. All participants had normal to corrected vision. Each participant completed the 3 experiments in a single session.
15 university undergraduates (8 men, 7 women; age range 18 - 24 years) participated in Experiments 4-7 for course credit. All participants had normal to corrected vision. Each participant completed the 4 experiments in a single session.

Each participant completed a supervised practice session of 5 trials before the experiment. The user would indicate his/her response to questions by clicking on the corresponding option on screen. For example, to indicate that a line was aligned, (s)he would click on the radio button appearing next to the word “Aligned” appearing on screen.

**Test Images:** The lines of varying orientations were generated in this manner: we start out with vertical line. Each pixel in the vertical line is divided into a $9 \times 9$ array of subpixels. The pixel intensity is divided equally between all of the subpixels. Each subpixel is then rotated by the specified angle, to a different pixel location, Fig.5.4. The new pixel location’s intensity is incremented by the subpixel intensity value. This routine resulted in smooth lines, without the jagged edges in lines obtained by a simple setting of pixel values. For test images with noise, the lines were constructed first, and noise added everywhere except at the lines themselves. Then the minimum value obtained
from the noise generator was determined, which could be negative. The absolute value of the minimum pixel value is then added to all of the test image pixels, including the pixels belonging to the test line.

The key argument in this document, as stated in Hypothesis 1 in Chapter 4, is that the human visual system may have an accumulator-array like structure for detection of straight lines in the primary visual cortex. That is, straight line detection is a hard-wired process, as opposed to a process involving the higher areas of the visual cortex. The human experiments were expected to add support by the following means:

- **Hypothesis 1:** Peripheral and foveal responses to straight lines will match, in terms of latency and accuracy. If peripheral and foveal responses match, it would imply that straight line detection does not require the full attention of the observer, and hence is a process happening in the early stages of the visual cortex. It would add weight to the “hard-wired” idea. The experiment results substantiated this hypothesis.

- **Hypothesis 2:** The observers will detect lines much faster than contours, aligned or non-aligned. If straight lines are detected faster than contours, it would once again add weight to the before mentioned ideas. This is because contours give a weaker response than straight lines in an accumulator-array structure. The experiment results did not fully support this hypothesis - aligned contours and lines were detected with the same latency and accuracy. However, non-aligned lines were detected as misaligned more accurately, and with a shorter response time, than the non-aligned contours. The implications are discussed in the next section.

- **Hypothesis 3:** Better alignment of line segments in terms of $\rho$ and $\theta$ as well as reduced/low noise conditions should improve accuracy as well as
latency of human observers. If the responses of the human observers match that of the Hough Transform implementation/Neural Network under 1) various alignment conditions for line segments and 2) presence of noise, it would add further weight to our key hypothesis. The experiment results support this hypothesis.

5.2 Results

The results obtained from the fovea are shown in Fig. 5.5 for variation in $\theta$, $\rho$, and $\sigma$ of noise. The results from periphery are shown in Fig. 5.6.

As expected, in both fovea and periphery, the human subjects detected the lines as aligned with a higher probability when the variation in $\theta$ and $\rho$ values were low. In the periphery, the drop in probability as a function of $\rho$ is less abrupt than as a function of $\theta$. In presence of noise, as the noise increased, the subjects failed to detect the lines, no matter the orientation, in the fovea. The same behaviour was seen in the periphery as well.

The overall variation in the curves follow the same trend for both fovea and periphery and supports the idea that better alignment of line segments in terms of $\rho$ and $\theta$ as well as reduced noise conditions improves accuracy of human observers. No marked variation in latency was observed in these experiments.

The response times for the users for both fovea and periphery was recorded, and the average response time in both cases was found to match - Fig. 5.7. Combined with the previous results, it adds support to the hypothesis that peripheral and foveal responses match.

The human subject responses to the contour vs straight line experiment are shown in Fig. 5.8. The responses to aligned straight lines and contours matched in terms of accuracy
Figure 5.5: Plots showing the variation in detection of the lines by the human observers, for all 4 angles when $\theta$, $\rho$, and noise $\sigma$ were varied, in fovea.
Figure 5.6: Plots showing the variation in detection of the lines by the human observers, for all 4 angles when $\theta$, $\rho$, and noise $\sigma$ were varied, in periphery.
and response time. This contradicted our hypothesis that lines would be detected faster than contours. However, non-aligned straight lines were detected as misaligned more accurately, and with a shorter response time, than the non-aligned contour. Since the non-aligned straight lines had two alternate line segments aligned along a line, while the non-aligned contour had two alternate line segments aligned along a contour, both cases had comparable degrees of misalignment. It could be argued that humans are better at detecting lines as misaligned than contours in short intervals of time - given sufficient time to examine the test images, the subjects could have easily determined the misaligned contours easily. However, for the very short times the test images were displayed, the subjects incorrectly identified contours as aligned a lot more often than lines. This could still imply that straight line detection is a hard wired process, not needing the full attention of the observer. The matching responses for both contours and lines in perfectly aligned cases could be because the aligned line segments give rise to comparable responses while viewing contours as well as straight lines.

(a) User response time, Fovea
(b) User response time, Periphery

Figure 5.7: Comparison of fovea and periphery response times.
(a) No: of times out of 16, when each of the cases was detected as aligned.

(b) Reaction time when the user response was correct

(c) Reaction time when user response was incorrect

Figure 5.8: Results from the contour experiments.
Chapter 6

Signal Separation

This chapter presents a neural network capable of clustering inputs that are similar in a squared error sense. This network can be thought of as a precursor to the network outlined previously. In chapter 4, separate networks were trained with lines of separate orientations. The question that may be asked is: in the natural world, how do these orientations get separated before encountering a network similar to the one discussed? The network presented in this chapter may be a solution to this question - given lines of different orientations, this network separates out the different orientations into separate clusters.

Separation of data into clusters based on a measure of similarity is a well researched idea[71]. In vision problems, data takes the form of images and video. Since the best vision system known to mankind is the visual cortex, biologically inspired vision algorithms are becoming common place, as described in chapter 1. In this chapter, a biologically inspired, unsupervised learning algorithm capable of separating input images into clusters is presented.

One of the interesting aspects of the mammalian visual cortex is the organization of
orientation-sensitive cells into orientation columns, as seen in chapter 2. As one moves across the cortex, one observes that the orientation sensitivity changes gradually and smoothly. Hubel and Wiesel[26] used this observation in their ice cube model of the cortex. An unanswered question is: “how does this come about; what process organizes the cells so that the changes moving from one cell to the adjacent cell is smooth?”. While the algorithm described in this chapter does not explicitly answer this question, it may provide some interesting insight.

The outlined neural network, when presented with lines of different orientations, separates them in a topologically ordered manner, similar to the cortex. The novel aspect of this network is that it combines Oja’s rule with Kohonen’s Self-Organizing Maps principles in order to do so. The idea of a winning neighborhood as opposed to a single winner in a winner-take-all algorithm context, is also used. When tested on the MNIST[39] database, it was found that the network is capable of separating other inputs as well, based on their similarity.

The chapter is organized as follows: Section 1 outlines the architecture and the learning rules of our neural network. Section 2 contains the results and their discussion.

6.1 Separation of Lines

In this section, a network capable of separating lines of different orientations is outlined. That is, using unsupervised learning methods, different neurons learn to become sensitive to different orientations, even when trained using lines of several orientations. In effect, this network serves as a precursur to the network outlined in Chapter 4.

The network, shown in Fig. 6.1, has a single layer of neurons fully connected to the input. The neurons are arranged in a two dimensional lattice.
The output is computed as follows:

\[ y_i = f(\sum_j w_{ij}x_j) \]  \hspace{1cm} (6.1)

In (6.1), \( f \) refers to a smooth thresholding function like the sigmoid.

In the next step of the algorithm, we determine the winning neuron. The winner is decided based on the outputs, rather than the weights, associated with the neuron. The winner is decided not by looking at the single neuron with the maximum output, but by looking at the neighborhood which gives the maximum output, see Fig. 4.5. If, for example, \( i^* \) denotes the winning index in the output array:

\[ g_i = \sum_j y_j \tilde{r}_{ij} \]

\[ i^* = \arg\max_i (g_i) \]  \hspace{1cm} (6.2)

The \( \tilde{r}_{ij} \) term represents a 2 dimensional Gaussian neighborhood.
Once the winner is decided, a neighborhood function is defined for the lattice of neurons. The neighborhood function can thus be defined as:

\[ h_j = \exp\left(\frac{-|i^* - j|^2}{2(\sigma^*)^2}\right) \]  

(6.3)

As in Kohonen’s *Self-Organizing Maps* [35], page 87, the neighborhood shrinks over time.

While updating weights, a combination of Oja’s rule and Kohonen’s SOM is used. The weights are updated as follows, where \( \alpha \) is the learning parameter:

\[ w_{ij} = w_{ij} + \alpha h_i y_i (x_j - y_i w_{ij}) \]  

(6.4)

The learning parameter decreases with time in an exponential manner. That is, if \( \alpha_o \) denotes the initial value of the learning parameter:

\[ \alpha(t) = \alpha_o \exp\left(-\frac{t}{\tau}\right) \]  

(6.5)

In the above network, in place of inhibition between neurons in the same layer, we have competition. The neurons are thus forced to respond maximally to different inputs.

### 6.2 Results and Discussion

The incoming weights to each neuron were initialized to small random numbers and normalized to 1. The starting value for \( \alpha \) was 0.1, and the starting value for the neighborhood function \( \sigma \) was 10. 10000 training samples were presented, one at a time, randomly chosen from a set of 5000 samples.
With straight lines as inputs, the neurons learned to respond to a single orientation maximally from the various orientations with which they were presented. Sample training images and the weight vectors of the different neurons are shown in Fig. 6.2 and Fig. 6.3.

The result obtained after training with the MNIST [39] database is shown in Fig. 6.4. MNIST is a database of handwritten digits, normalized in size, with a training set of approximately 60,000 examples. Neighboring neurons learned to respond to the same number, as can be noted from the figure. Neuron neighborhoods lying next to each other learned to respond to numbers with similar appearance.

Neurons at the border between two neighborhoods responded to both both inputs, but the response was weaker than those of the neurons belonging to the neighborhoods themselves.

The variation in orientation across neurons in Fig. 6.2 is similar to variation in orientation columns observed in the visual cortex [27].

In Fig. 6.4, 500 distinct samples for each of the numbers 0-9 were used. Suppose only a single sample were used, and only numbers from 0-7 were considered. The numbers 0-7 were considered because we were interested in the number of distinct samples being a power of two, which in this case is 8. The resulting weights are shown in Fig. 6.5. As seen in the figure, the neurons self-organized themselves into clusters responding to particular numbers. After training, when any particular number was presented during testing, the center of the cluster corresponding to the number responded maximally.

For further analysis, consider the following: Let matrix $\bar{x}$ represent normalized input image “0” and $\bar{w}_i$, $i = 1 \ldots 28 \times 28$ represent the weight matrix associated with neuron $i$. Then we define error $e_i$:

$$e_i = \|\bar{x} - \bar{w}_i\|^2$$

(6.6)
Figure 6.2: Training Images and Resulting Weights, with 4 training lines.
Figure 6.3: Training Images and Resulting Weights, with 12 training lines.

(a) 12 lines

(b) Weights of 256 neurons, arranged in a lattice
Figure 6.4: Weights of $28 \times 28$ neurons, arranged in a lattice, after training with MNIST numbers.

Variation in error with iterations for a neuron within the zero cluster and for a neuron outside the zero cluster are shown in Fig. 6.6. As can be observed, a neuron within the cluster gets closer to the representative input with each passing iteration, while a neuron outside the cluster grows farther away from it.

We also examined how similar the weights were amongst themselves, after training. For neuron at position one, we computed the following Euclidean measure of dissimilarity:

$$e_i = \|\bar{w}_1 - \bar{w}_i\|^2 \quad (6.7)$$

The results are shown in Fig. 6.7.a. As can be seen, the dissimilarity increases with distance from the neuron. The shallow regions represent the cluster border, where two or
more numbers overlap. A similar squared error between the representative input for the first cluster and the rest of weights was computed. The surface plot obtained was similar to the one already obtained, as seen in Fig. 6.7.b.

![Figure 6.5: Weights of 28×28 neurons, arranged in a lattice, after training with single samples of MNIST numbers 0-7.](image)

6.2.1 Comparison with SOM

The outlined algorithm was compared to Kohonen’s SOM. The rate of convergence of the two algorithms were comparable - that is, the squared error within the cluster decreased to zero at approximately 4000 iterations in both implementations. However, our algorithm resulted in a mapping that better corresponded with the dissimilarities between inputs.
(a) Decrease in squared error within the cluster  (b) Increase in squared error outside the cluster

Figure 6.6: Plots showing variation of squared error with increased iterations for a neuron a) within the cluster b) outside the cluster.

(a) Dissimilarity of weights  (b) Squared error between input and weights

Figure 6.7: Plots showing a) variation of dissimilarity between weights b) squared error between input and weights.
That is, the SOM may place two dissimilar clusters next to each other, Fig.6.8, while our algorithm ensures that they are placed as far away as possible, Fig.6.7.a. None of the figures in Fig.6.7 or Fig.6.8 are monotonic. However, in Fig.6.7.a, the cluster most dissimilar to the neuron at the first location is placed diagonally opposite to the neuron, while in Fig.6.8, it is placed in a corner on the same side.

6.2.2 Comparison with Oja’s rule

Oja’s rule, by itself, extracts the first principal component of the data presented. Unless the relationship between neurons are modeled, all of the neuron weights would end up being the same. Oja’s subspace algorithm or the Anti-Hebbian rule, discussed previously, does model the relationship between output neurons - however, they do not produce topologically ordered clusters as is done by our network. The difference being, we use self-organizing map principles to model the relationship between neurons, thereby producing the clusters obtained.
Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis described a new neural framework for detection of straight lines, in a manner similar to that of the Hough Transform. Human experiments that add support to the idea of such a computation happening in the visual cortex were conducted, and the results described. Besides the architecture itself, the novel ideas introduced in this work include:

- a winning neighborhood as opposed to a single winner in a winner take-all algorithm.
- a new learning rule for intra-array neuron connections, including its analysis and stand-alone results.
- a new method for determining the orientation based on estimates from networks tuned to 4 values - 0°, 45°, 90°, and 135°.
- a new algorithm for unsupervised separation of inputs, described in Chapter 6.
7.2 Future Work

In the future, the current work can be extended to recognition of more complex objects. The principle of the Hough Transform would be utilized once again - the objects would be recognized as a result of a voting process, the parameters being voted on being the various features. The questions to be answered are, broadly speaking - 1) what features should be voted on to recognize the class of objects under consideration? 2) can this process be learned in an unsupervised manner? Starting with simple shapes like circles, one can move on to recognizing complex objects like pedestrians and buildings. Questions of accuracy, computation speed and performance under noisy conditions should be addressed.

The proposed work is interdisciplinary, with elements borrowed from computational neuroscience, neural networks, and computer vision. It has room for collaboration with external research groups, especially for experiments with human subjects/recording human visual cortex activity. Increased interest in how the brain works has resulted in research funding becoming available for various aspects of brain research by government agencies and private organizations; they could be a potential sponsor for the biologically inspired object recognition outlined above.
REFERENCES


