A Neural Network Model of the Search Behavior of Cataglyphis Bicolor

Jachin Rupe
Department of Computer Science
Augsburg College
rupe@augsburg.edu

Efoe Akolly
Department of Computer Science
Augsburg College
akolly@augsburg.edu

Abstract

This paper presents a neural network that learns to wander in search of food in a manner similar to the ant Cataglyphis Bicolor. The semi-random nature of this particular behavior in the ant's navigation system presents many challenges in designing a network and learning rule. The ant leaves its nest in a particular direction, then wanders about randomly with the bulk of its movements taking it farther from the nest. When the behavior is traced it shows a fairly detailed search of a large swath of ground that is wider than it would have been if the ant had kept to a straight line but still extends a large distance from the nest.

Introduction

Cataglyphis Bicolor

The ant called Cataglyphis Bicolor lives in the Sahara desert. It is named Bicolor (two colors) just for the fact that its thorax has an orange-red color and the rest of its body is black. Unlike many other ants, Cataglyphis Bicolor does not secrete pheromone trails in order to find its way home but instead uses the sun as a compass to keep track of a vector home. The home vector can be inaccurate, so it also uses landmark recognition. Typically, landmarks are trees, rocks, and shrubs. Cataglyphis Bicolor has been the object of studies over the years by Rudiger Wehner at the University of Zurich, in Switzerland [4].

Augsburg Research Opportunity for Undergraduates

The Robotic Navigation Project is a National Science Foundation funded Research Experience for Undergraduates (REU). The project is an eight-week program with focus on the areas of robotic navigation and cognitive science. Conceived and developed by Dr. Karen Sutherland and Dr. Charles Sheaffer, both of Augsburg College, the program gives computer science students a chance to work in a research environment on projects including (but not limited to):

- Investigating the use of landmarks for navigation by creatures such as Cataglyphis Bicolor, the desert ant.
- Analyzing the use of composite landmarks such as ridge-lines, valleys or peak formations for navigation and writing and testing algorithms to exploit these composite features.
- Developing learning algorithms, which would relate a library of shapes to landmarks to be used for navigation.

We worked under the direction of Dr. Charles Sheaffer on developing learning algorithms.

Independent Study

After the summer program finished I wanted to continue my study of neural networks and continue the work started during the summer of 2002. So under the direction of Dr. Charles Sheaffer this work is continuing as an independent study during the spring of 2003.

Goals

The primary goals of this project were to bring together the work on neural networks done by previous research groups at Augsburg College that were working on modeling particular parts of the Cataglyphis Bicolor's navigation system, build a simulator to test the various parts of the navigation model, and ultimately get them all to work together. To achieve these goals a few aspects of the ant's navigation that had not been modeled in a neural network would have to be built to fill in the gaps.

Entire Ant Navigation System

The basic algorithm for the ant's navigation system is as follows [2]:

- The Ant begins at the nest
  - it turns until it sees two or more objects at the same time
  - it takes a snapshot and compass heading of what it is looking at
- The Ant begins to wander
  - a timer starts... the ant has to return to the nest when the timer runs out
  - the ant tries to cover as much ground as far away from the nest as possible all the while keeping track of the compass heading that will take it back to the nest and the distance from the nest.
- The Ant returns home
  - the ant follows its compass heading back to the nest
  - when it thinks it is close enough to see the landmarks it took a snapshot of when it left the nest, it starts looking for those landmarks
  - once it sees them it tries to line up the landmarks exactly as they were before it started wandering
  - when the landmarks line up the ant should be back at the nest

Robot

Once the navigation system works in simulation the next step would be to put the navigation system on a robot. A good candidate for a robot would be a design that made use of the BrainStem[tm] that another group during the Summer of 2003 REU was working on building.

Neural Networks

Introduction

There are many approaches to the problem of artificial intelligence. Most approaches attempt to model the reasoning processes humans go through when they try to solve a problem. Artificial neural networks take a different approach, attempting to model the biological structures that brains are believed to use in order to solve problems.

Neural networks are also referred to as “connection models” or “neuromorphic systems”. I’m only going to use the term neural network for the entirety of the paper. I will also assume that when I refer to neural networks I will be talking about artificial neural networks.

History

The study of biological neural networks began in the late 1800s. Naturally no one could create neural networks to experiment on; they could only be observed and theorized about. With the invention of computers neural networking became one of the first artificial intelligence ideas that was experimented with. However it became apparent that the amount of processing power needed to solve all but the most simple neural networks was not available. Consequently neural networks fell out of favor and other artificial intelligence techniques were pursued.

However in the last 20 years, thanks to effects of Moore’s law, the artificial intelligence community has taken another look at neural networks. While progress has been slow, neural networks show a lot of potential.

The Big Idea

The topic of neural networks is cross-disciplinary, drawing off knowledge from biology (mostly neuroanatomy) and psychology in addition to mathematics and computer science.

Biologists and psychologists are interested because brains appear to use a system of neural networks to learn and adapt. Biological neural networks are far superior to anything that has been created to run on computers (most neural networks are implemented with software as opposed to hardware). Knowledge gleaned from biological and psychological studies on biological neural networks has proven very useful in creating artificial neural networks.

Capabilities

A human's brain has time and time again proved its superiority over all the attempts by artificial intelligence to solve many types of problems. Humans have shown a profound ability to learn and adjust to similar but different situations. Biological neural networks are also capable of functioning when part of the network is damaged.

Artificial neural networks, while nowhere near the capabilities of what they are trying to model, do show some promise. They have shown the ability to learn how to solve a problem, in some cases at the same rate as humans. They also have the ability for graceful degradation (continue functioning even when part of the network is damaged).

Applications

Neural networks have been used with success in several experiments and real world applications, some of which include pattern recognition (handwriting, images etc.), control (robots), and optimization problems.

How Neural Networks work

Neural networks have two basic components: nodes and links (figure 1). Nodes are organized into layers, these are organized into an input layer and interior and output layers. The input layers are where input is collected, the interior is where the decisions are made. The last interior layer is called the output layer which is where the final decision is rendered. Links represent relationships between different nodes.

Nodes

Nodes are analogous to neurons (figure 2) in biological neural networks (brains). Each node has an activation value that represents the level of stimulation for that particular node. Activation values are numbers between a high bound (maximum activation) and a low bound (no activation). These number are often stored as a floating-point number between 0 and 1.

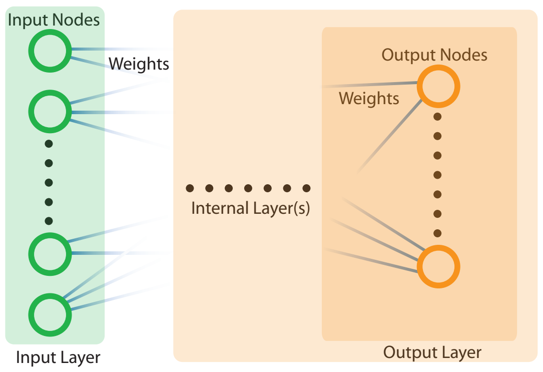

Figure 1: The Basic Design of a Simple Neural Network

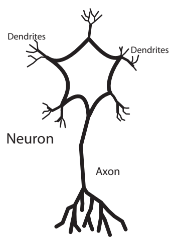

Figure 2: Neuron

Links

Links are the equivalent to dendrites and axons in biological neural networks. Each link connects two nodes (an origin and a target) and has a weight associated with it. The idea is exactly the same as the edge of a weighted directional graph. A higher weight means that the relationship between the two nodes that the link represents is stronger; a low weight means the relationship is weak. In practice it is often convenient to represent weights the same way as activation values for nodes (floating point numbers between 0.0 and 1.0 for example).

Layers

Input Layers

The nodes in the input layer function a little differently than the nodes in the rest of the network. They get their activation values from the inputs into the problem.

Here is an example of how the input to a neural network could work. Consider a network designed as part of an autopilot system to keep a plain flying level (figure 3). The input could be an angle reading from the horizon instrument.

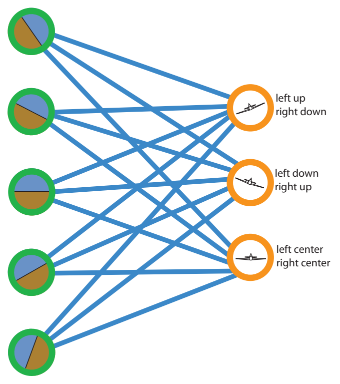

Figure 3: Level Flyer

Each node in the input layer would represent a range of angle values that could come from the horizon instrument. Every second the input layer would get the current reading from the horizon instrument, then the activation value for each node would be calculated based on their range (figure 4). The simplest way to do this is have each node represent a range of degrees and have no overlap. Then when the instrument returns a particular angle the node that is in that range gets an activation value of 1.0 and all the other nodes in the input layer get activation values of 0.0. Often the simplest approach is not the best. Using a more complex method can reduce the number of nodes needed in a neural network to solve the problem and allow it to learn more quickly. A tent function (figure 5) is an example of a simple method that can be used to calculate the activation for a particular input node. With a tent function each activation node needs a "center" (where the activation is its highest) and "width" which is used to calculate where the function intersects with the x-axis (where the activation becomes 0.0). If the tent function for a particular node evaluates at less than 0.0 then the input value is outside of its range so its activation is just 0.0.

Output and Internal Layers

Output and internal layers get their input values from the links they are the targets of. The function that decides the activation for a node in an internal or output layer is called the activation function. An example of a simple activation function is one that takes the activation values for each node that has a link to it and

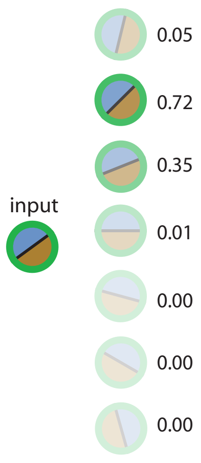

Figure 4: Input Layers

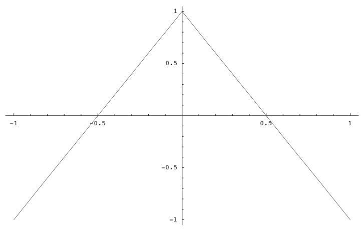

Figure 5: Tent Function

multiplies it with the weight of the link that connects them. The number that is the result of each multiplication is known as a signal. All the signals are added together and run though a function (figure 6) to calculate the final activation value for node.

An example of how this would work would be the output layer for the neural network designed to fly level. Each output node could represent a reaction by the plane's ailerons. Another example would be a neural network designed to recognize handwriting (figure 7). Each node in the output layer could represent a character that the network could recognize. The node with the highest activation value is the final result.

Learning Rule

Most practical neural networks learn via a learning rule. The way most learning rules work is that the network is given a set of problems to solve whose answer is

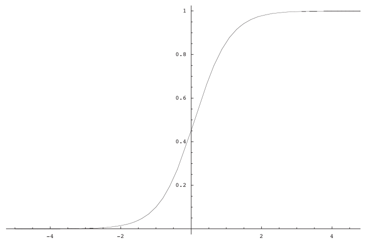

Figure 6: Sigmoid Function

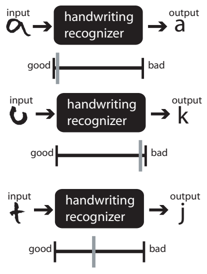

Figure 7: Examples of Learning Rules

already known. When the neural network returns each answer it is judged, at the very least as either as correct or false, but more often is given a score chosen from a wide range of grades. Then the activation values of the nodes are examined to see which parts of the networks were responsible for the decision. If the decision was a good one the relationships that were responsible for the decision are rewarded (strengthened) and if the decision was a bad one the relationships are punished (weakened).

Consider a neural network that is being trained to look at scanned handwriting and do handwriting recognition. The network could be given a letter at a time. When it correctly recognizes the letter the links responsible are rewarded, otherwise they are punished. The learning rule could also be made more sophisticated by taking into account that some letters look similar. For instance if a neural network recognized an “t” as a “j” it might rewarded a little; however if it recognized the “u” as a ”k” it would be severely punished.

Neural networks that require training should be designed and adjusted to minimize the amount of training necessary before the network can complete the task it is designed for.

The set of problems used to train a neural network is known as a training set. It is important that the training set have enough members that every link in the network has an opportunity to have its weight adjusted (probably several times). Consequently the members of the training set should be evenly distributed over the range of input values for the network.

One simple strategy that would work well for the example of the level flyer would be to put the airplane at a series of different angles. The network would be given each member of the training set. If the training set was large enough and had the full range of possible values, before long the network would be fully trained. In a simple scenario like this the easiest way to generate the training set is have a good pseudo-random number generator generate the training set.

Feed Forward Networks

Feed Forward networks are the simplest type of neural network. They get their name because all links connect a node in one layer to a node in the next layer (closer to the output layer).

An especially simple feed forward network is the linear associator. They are only useful when there is a linear association between the input and the output. More complex problems require networks with more complex topologies and learning rules.

Why we chose to use Neural Networks

We chose neural networks because neural networks fit well into possible applications of the complete navigation system. One of the advantages neural networks have is that they not only can learn how to solve a particular problem they can also adapt to solve slightly different problems.

There are many common scenarios where the ability is especially useful in robotics. One such scenario would be if a design flaw caused the right wheel on a robot to receive a little extra power causing the robot to tend to the left. A navigation system for the robot that used a neural network could learn to compensate for the design flaw and allow the robot to function normally.

The Wanderer

The specifications for The Wanderer came from the work done by Wehner and Srinivasan[5]. The basic idea is that the ant picks a direction to search for food once it leaves the nest (a target vector). Then it "wanders" in that direction, moving about semi-randomly. The final result of the ant's movements take it away from the nest and in the general direction of the heading that the ant picked when it left the nest, all the while not going over ground it has already searched. The figure 8 is from Wehner's website[4] which shows the wandering behavior we are interested in.

Our first thought was that this problem would be a very easy one, being that requirements for the ant's behavior are not very strict. However after working on the problem it became apparent that this was not the case. The most difficult aspect of the wandering behavior to mimic is its back and forth nature. The ant wanderers in almost a straight line until its home vector gets too far away from target vector. Then the ant mirrors its heading over the target vector and continues in wider and wider arcs until it decides to return home. This behavior is not yet fully simulated.

The aspects of the ant's wandering that we are able to mimic are the ant's movements taking it away from the nest and the seemingly random path it takes over a small area.

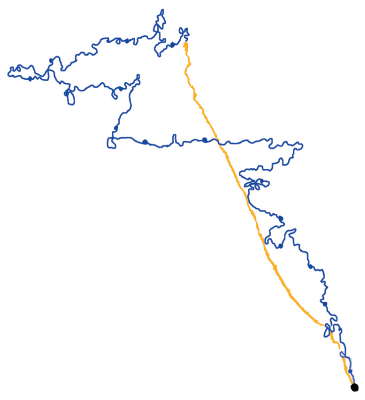

Figure 8: Search Behavior of Cataglyphis Bicolor

How The Wanderer works

The wanderer operates in a simulator (figure 9) that we during the Augsburg summer 2002 REU. The ant is trained for a period of time. Once that is complete, the ant is placed back at the nest and is set to wander.

The basic design of all the networks I have used has been the same. As a rule we tried to make the simplest neural network possible. The Wanderer had two layers:

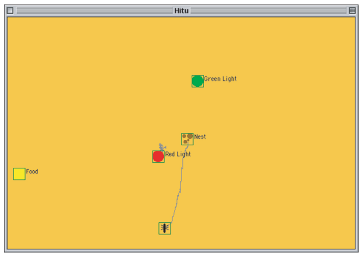

Figure 9: Simulator Screen Shot

an input layer and an output layer.

During the summer of 2002 we tried three different learning rules and corresponding network designs (with a few minor variations). The first two learning rules never worked due to bugs in the implementation of neural networks and shortcomings in their design. We believe that these learning rules may have potential, especially the second one but we have not had the opportunity to implement them properly.

Learning Rule One

The first learning rule (figure 10) used the direction the ant was facing as the input to the neural network and the output being a turn ranging from -180 to +180 (negative numbers being left turns and positive right). The learning rule would give positive feedback if the turn that the ant just made left it facing a heading within a set range of the heading it was supposed to wander in, and negative feedback for all other turns.

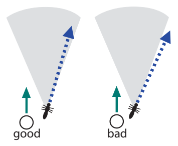

Figure 10: Learning Rule One

Learning Rule Two

The second learning rule (figure 11) had the same output as the first but instead measured the angle of the ant's home vector and used that as the input for the network. The network was given positive feedback if the home vector of the ant's current position was within a range of values around the heading in which the ant was supposed to be wandering and negative feedback otherwise.

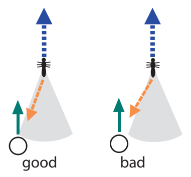

Figure 11: Learning Rule Two

Problems with the first two Learning Rules

The first of the problems with these designs is that the output layer is just generating turns, not headings. This would not have been a problem for the first rule if we had ignored all the training cases below (south) the nest because the learning rule takes the heading of the ant into account. This is an even bigger problem for second learning rule because it does not take the ant's heading into account, so the feedback from the learning rule is arbitrary.

The second problem with the first two learning rules is that they do not require the ant to actually get farther from the nest, they just look at the ant's position relative to the nest in terms of angles, without considering distance from the nest.

Learning Rule Three

For the third training rule (figure 12) the ant receives positive feedback if it is inside the window and moving away from the nest (in the proper direction), and negative feedback if it is moving toward the nest (or away from it in the improper direction). If the ant is outside the window, it is given positive feedback if it moves back toward the window and negative feedback otherwise.

This rule eventually trained the neural network to mimic most of the aspects of the wandering of the Cataglyphis Bicolor. However there were a few problems that had to be addressed before it worked. The first was that the ant moved around in perfectly straight lines. This problem was fixed by adding noise to the network by getting the output nodes to fire randomly[3].

The ant also tended to wander only towards the right or only towards the left. This is caused because the ant is being trained with random turns; the total number of

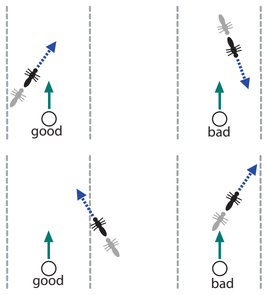

Figure 12: Learning Rule Three

right training turns and left training turns slowly diverges the longer the ant is trained. As a result the ant will favor one side or the other. One solution to this is to mirror training. So when the ant gets input for a particular decision the input is then mirrored to give the opposite heading the exact same feedback[3].

Future Research

In the future one could continue to work toward completing all the goals we initially set. The simulator still needs more work. Many of the features are not complete, not to mention all the features that could be added to make it a more useful application. Some of those would include more controls over the simulator via the graphic user interface, as well as more information about what exactly is going on in the simulation. Some sort of logging would also be useful to keep track of what sort of training sets have been used to train various neural networks.

The Wanderer still needs to be perfected to better mimic the wandering of the Cataglyphis Bicolor. Once that is complete it would be interesting to compare its effectiveness in different situations and compare it to other searching algorithms.

The BrainStem[tm] now seems to be working fairly well with all the necessary instruments (including a camera and compass) to replicate the ant's form of navigation[1]. Once all the aspects of the ant's navigation system have been built and are working together a robot could be build that navigates like the ant.

References

1. Bozonie, L. and Tuttle, C. (2002). Modeling the visual centering responce with the brainstem[tm] and cmucam[tm]. Technical report, Augsburg College: Department of Computer Science.

2. Cannon, K. and Riley, D. (2002). Java simulation and robot modeling of the cataglyphis bicolor. Technical report, Augsburg College: Department of Computer Science.

3. Sheaffer, C. M. (1994). *Adaptive Sensory/Motor Integration for an Autonomous Mobile Robot*. PhD thesis, University of Minnesota, Minneapolis, MN 55414.

4. Wehner, R. (2002). Home Page. http://www.neuroscience.unizh.ch/e/groups/wehner00.htm.

5. Wehner, R. and Srinivasan, M. (1981). Searching behavior of desert ants, genus cataglyphis. *Journal of Comparative Physiology*, 142:315–338.

Acknowledgements

I would like to acknowledge Dr. Charles Sheaffer for his instruction and guidance and Eric Johnson for his encouragement and help in preparing this paper.