Alex Likos

Final Paper 1

3/18/10

# Controlling a Sailboat with an Artificial Neural Network

## Introduction

Ever since I saw *The Matrix* I have been interested in the idea of machines with human capabilities, which is more formally known as true artificial intelligence. The most basic aspect of the human brain is it's ability to learn, so I feel that the first step towards a true AI is getting a computer to learn how to do a real world activity and be able to perform the task as well as a human.

Sailing is a perfect example of a real world activity that would be difficult for a computer to perform, partially because it is also difficult  for a human to perform. The task of sailing a boat, as with any real world problem, involves a great deal of information processing. Observing things like wind movement, current, and combining that information with the sail position  One could hard code every known situation that a sailor could encounter on the water, but the resulting program would be huge and inflexible, since one cannot fully account for every situation on the water. On top of which a human pilot has access to a far better system of sensory input and information processing than a computer does. Human sensory input gathers information from a longer range and with greater accuracy than computer sensors could ever have and the human brain can process that input with the equivalent power of 12 billion 60 Hz processing cores.

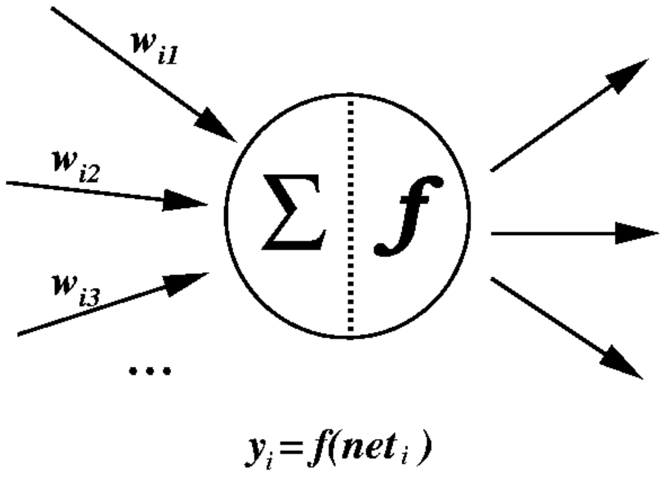
It is because of this that I am using an artificial neural network to tackle this task. Neural networks are good at processing information. After it is trained, a neural network can quickly analyze large amounts of data and generating correct output. However, as powerful as they are once they are trained, they can only perform the task for which they are trained and they can be trained incorrectly or incompletely. That is what makes them so interesting.

Because of the expense and time limitations of using a real boat in this project, I will train a back propagation neural network to pilot a boat using a sailing simulator.

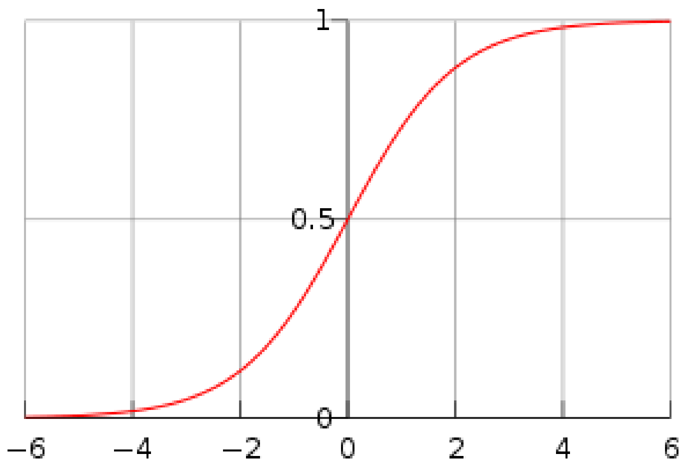
## Background

### The Artificial Neuron

The basic computational element in a neural network is the artificial neuron. Each neuron has connections for both input and output. Each connection has an associated weight, which can be modified in order to model synaptic learning. The neuron contains an activation function that takes the weighted sum of its inputs and calculates the output for the neuron. The most commonly used activation function is the sigmoid function, due to the fact that it tends to allow the neural network to reach global optima better than any other activation function



*Drawing 1: Representation of an artificial neuron*



*Drawing 2: Graph of a sigmoid function*

## The Artificial Neural Network

An artificial neural network (ANN) is a computational model that tries to simulate learning. The system is implemented as a network of interconnected processing elements that mimic the function of neurons in the brain. For that reason, these simple processing elements are also called neurons.

Neurons in the brain look like little balls with threads that branch off and connect to other neurons. They function as very simple processing units. An individual neuron takes an input signal, manipulates it in some way and send it out to the other neurons it is connected to, which in turn does the same thing. The artificial neurons in the ANN function in a very similar way. A basic ANN has a layer of input nodes, a layer of hidden nodes and a layer of output nodes. The input nodes contain data given to the neural network that will be manipulated to calculate the value(s) in the output.

The hidden layer contains nodes that function as neurons. Unlike input and output nodes which contain values, they contain an activation function. The functions take the combined values of all the incoming signals and allow a portion of the value at that node to pass through to the output nodes. If this function is a stepwise function, the activation function behaves more like an activation level. If the activation level of a hidden node is reached or exceeded, the value at that hidden node is sent to the output level. Otherwise the node does not activate and no signal is sent out from that node.

An important difference between real neurons and artificial neurons lies in the properties of the connections between the neurons. Connections between real neurons  are considered "loss-less" which means that the signal does not change from the time it is sent out of one neuron and received by another. Connections between artificial neurons are weighted so that the value changes as it is passed from node to node. These weighted connections are the property that allow the artificial neural network to learn.

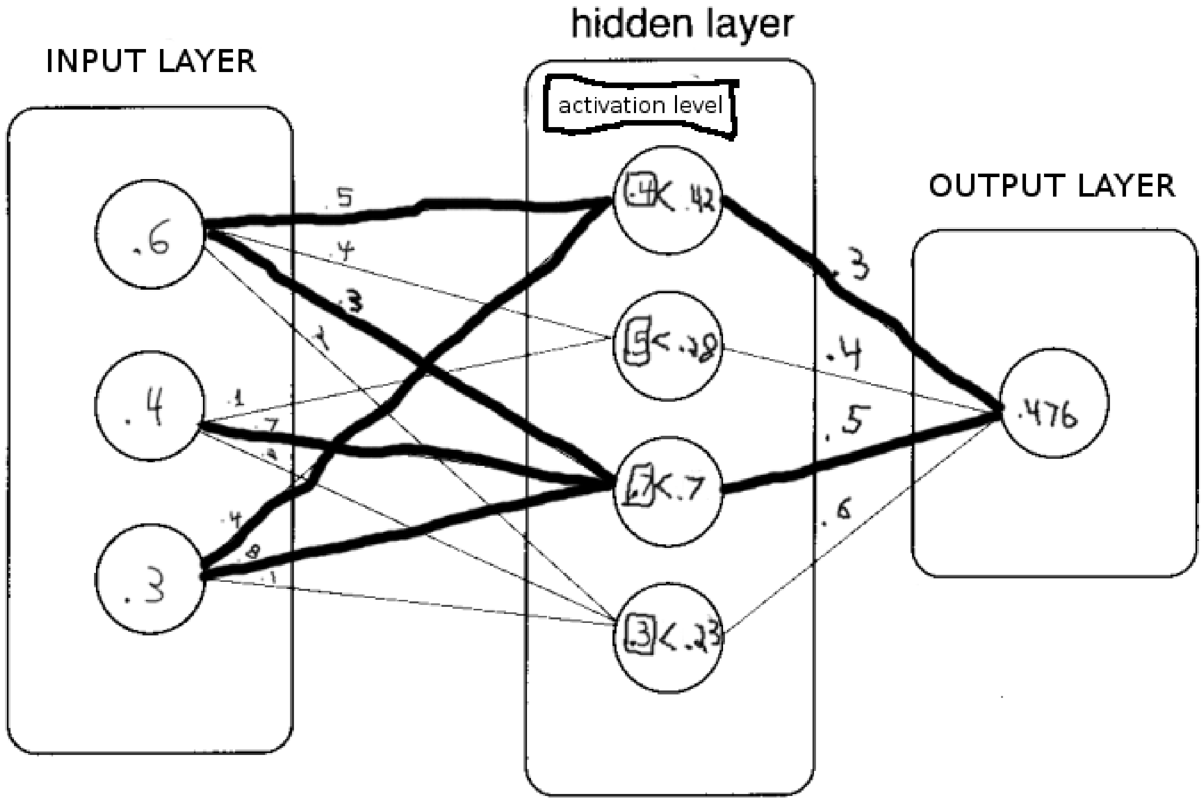


*Illustration 3: simple multi-layer, not-fully-connected neural network with input data propagated through the network and the calculated output. B*

*Breakdown of top hidden node value:* HN1=(.6*.5)+(.3*.4)=.42

TopInputNode=.6; weighted val=.3

BottomInputNode=.3; weighted val=.12

In the above example, the value given to the first (top) hidden node is the sum of the value in the first (top) input node multiplied by the weight of its connection to the first hidden node and the third (bottom) input node  multiplied by the weight of its connection to the first hidden node. The boxed value in each hidden node is the activation level. The only hidden nodes that are activated are the the first and third because the combined value of their inputs reaches or exceeds the activation level of those nodes. From there the output value is the sum of the value coming out of the activated hidden nodes multiplied by the weights of their connections to the output node.

## Training Humans vs. Training ANNs

The learning process for an artificial neural network learns  is very similar to the learning process for a human. Humans generally have a teacher when they first start learning a new skill. That teacher can be a book or another person. The teacher first explains what the student needs to observe about their surroundings in order to successfully perform the activity. They are then told what parts of their environment they have direct control over. After that, the student is given a rough idea of how to use the information they observe to decide how to best utilize their control over the environment.  Once the student has all this information, they must practice in order to become better. For a human, practice can be considered self-training because we are able to come up with new test situations, on the fly. This is where the human brain's ability to learn surpasses that of an artificial neural network.

Everything that was just mentioned about the human learning process has an analogue in the learning process of  an artificial neural network. The setup of the input and output nodes is analogous to a person being told what to observe and what they can control. The difference being that the training aspect requires more work on the part of the teacher. A neural network learns by observing rather than learning-by-doing. For this reason the teacher must know or be able to calculate the desired output for a set of input data. In order to learn, once the network has calculated what it “thinks” the output should be, the actual output must be compared to the desired output in order to calculate error. Some algorithm will use this error to adjust the weights in some way so as to minimize the error. This algorithm is the most important part of the learning process because the weight adjustment is what actually simulates learning in the neural network.

## Back-Propagation

There are many different algorithms to facilitate the learning process. The most commonly used algorithm is known as back-propagation or back-propagation of error. This algorithm compares the set of output values to a set desired value to calculate the error in the network. The error propagates backwards from the output nodes to the inner node, changing the connection weights so that the actual output will approach the desired output. The back-propagation algorithm falls under he category of supervised learning. Which means that the network is trained by a teacher that knows what the optimal output for the given set of input should be, which in this case is a large set of input and optimal output data. The process for back propagation learning goes as such:

1. Present a training sample to the neural network.
2. Compare the network's output to the desired output for that input. Calculate the error in each output neuron.
3. For each neuron, calculate what the output should have been, and a *scaling factor*, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
4. Adjust the weights of each neuron to lower the local error.
5. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
6. Repeat from step 3 on the neurons at the previous level, using each one's "blame" as its error.

This algorithm tries to find what nodes are contributing the most error to the output and reducing it by adjusting the weights.

## The ALVINN System

The ALVINN System[1] is a DARPA sponsored project to make an autonomous land vehicle controlled by an artificial neural network. This project has particular relevance to my own, because my goal at the end of this project is to have a neural network that can drive a real boat, despite the fact that I am not using a real sail boat to train the neural network.

### Josh Bongard's Work on Resilient Machines

The project regarding resilient machines[2] was my inspiration for using a simulator to train my artificial neural network. This project shows that it is possible to teach a neural network how to do a real world activity by training it in a realistic simulator.

## Project Details

### Overview

Artificial neural networks sometimes require prohibitively long training times and large training data sets to learn interesting tasks. As a result, few attempts have been made to apply artificial neural networks to complex real-world perception problems like sailing a boat or driving a car. The goal of this project is to teach an artificial neural network how to pilot a small sailboat using a back-propagation algorithm. However, trying to train the ANN with a real sailboat and a bunch of sensors will be both expensive and time consuming. After researching Josh Bongard's work on resilient machines, I have come to the conclusion that it is theoretically possible to train a neural network to perform a real life task with a simulator that closely models the real world. So, I will train the neural network using the sailing simulator, *Sail Simulator 5*, developed by Stentec Software. It has very accurate boat handling and environment physics, which makes it an ideal simulator for training an ANN that could pilot a sail boat in real life.

As previously mentioned, back-propagation neural networks need a teacher in order to learn. I will function as the teacher because I can use the navigational tools in the simulator to confirm that  I am sailing the boat in an optimal fashion. Which means that the set of desired output will be generated the simulator.

Picture 1:  *Screen shot from Sail Simulator 5. Sail boat flying a spinnaker*

Picture 2:  *Screen shot from Sail Simulator 5.*

Sail simulator 5 is able to send environmental data such as wind speed and direction to a hardware serial port. It outputs information using the *National Marine Electronics Association* standard format, that is used as input by most GPS navigation systems. To gather this information, I will use a COM port emulator to create a virtual COM port. The output from the game will be sent to the emulated COM port and I will capture that data with a COM port monitor that will write the data from the game to a text file. The text file will contain the all the necessary input data and the desired output data for the neural network and the neural network code will grab information from this file as necessary.

Once the network is trained, I need to be able to test it by allowing the network to sail the boat on its own . To do this, I am going to need be able to stream data from the simulator to my neural network and then get the output from the neural network into the simulator.

## The Function of the Neural Network

Since the network is performing the function of the skipper, it makes sense that it should be presented with the same input that a skipper observes while he sails. Namely

1. Boat facing(direction of movement)
2. Wind speed
3. Wind direction
4. How much the boat is heeling
5. How the skipper is distributing his weight to counteract the heeling of the boat
6. The location of a waypoint ( a desired location)

The output of the neural network should be the the factors of the environment that the skipper has direct control over. Namely:

1. The position of the sheets(the ropes that one uses to control the sails)
2. the position of the rudder

The network will also return a decision to tack(turn the boat across the wind while keeping the sheets in the same position). This is a technique that must be learned in order to sail upwind.

## Project Status

### Sail Simulator 5

Sail Simulator 5 was modified to output the information I need to give to the neural network. However, when a variable was in transit, for instance when the boat facing, sail position or rudder position was changing, the data became extremely noisy. This was extremely detrimental to the reliability of several important variables variable. Because of this, I was unable to use the simulator output as input for the neural network. However, using the simulator, I was able to take fairly accurate physical measurements of the necessary data. I took measurements of the optimal sail angle on the screen with a protractor and used the various navigational instruments in the simulator to gather other necessary information. With this data I made a database that generated the information in place of the simulator. It generates, wind speed and direction, boat speed and direction along with boat and waypoint gps coordinates and the sail and rudder position.

## The Neural Network

When I ran the information generated by my sail sim database through the neural network after 10000 training iterations my neural network output yields an error of  .02 in the rudder position and an error of .000005 in the sail position. After 20000 iterations rudder position error is  .004 and sail position error is .000002. After 80000 iterations, rudder position error is .002 and sail position error is .0000005. after 100000 iterations, rudder position error is .0002 and sail position error is .00000002.

## Feedback Test

I added a method into my sail sim data base that used my neural network to calculate rudder and sail position instead of using the database's native methods. The data generated in the feedback test did not deviate from the information generated by the training set by more than .0007 degrees on the rudder position and $1.2*10^{-7}$ degrees in sail position.

## Conclusion

The ultimate goal of this project is very similar to the goal of the ALVINN project: to create an artificial intelligence unit that can navigate a vehicle with no post-training human intervention and both projects yielded reasonably successful results.

The very low error in the output values of my neural network leads me to believe that the network has successfully learned how to sail and it does so in a relatively short amount of time. The ALVINN developers found that on average their network was able to take over control of the vehicle after 5 minutes of training, but they had a much larger training set and more hidden layers.

The main issues I ran into over the course of my project was caused by the sailing simulator. The original plan was to use the pre-modified simulator's capability to produce NMEA output and write an NMEA translator to get the data for the ANN. But, I later learned that the NMEA format is proprietary and access to a translator is expensive. The Stentec developers turned out to be generous enough to modify their code for me, but the data stream was too noisy to be useful. This became apparent too late into the project to ask stentec to fix the noise issue, which prompted the creation of the database. Since I am confident that my measurements were correct, I am confident that if I was to get the data link to work properly and used its output as input for the simulator, that the Neural network would have no trouble learning how to sail the boat in the simulator.

At the very least, this project explores some fairly new territory in the field of artificial neural networks. According to Dean Pomerleau, One of the developers of the ALVINN system, very few attempts have been made to use sensor input to an artificial neural network to perform a real world perception problem. Although my project does not deal specifically with the real world, it is using the same kind of information that is obtainable from digital sensors, and if Josh Bongard's resilient machines are any indication, a neural network that was trained using data generated by realistic simulator will perform just fine in the real world.

## References

1. The ALVINN Project Home Page: http://ftp.utcluj.ro/pub/docs/imaging/Autonomous_driving/Articole%20sortate/CThorpe/ALVINN%20Project%20Home%20Page.htm

2. **Efficient Training of Artificial Neural Networks for Autonomous Navigation**
Dean A. Pomerleau
School of *Computer Science, Carnegie Mellon University,*
*Pittsburgh, PA 15213 U S A*

*3. Resilient Machines Through Continuous Self-Modeling*
Josh Bongard, Victor Zykov, and Hod Lipson (17 November 2006)
*Science* **314** (5802), 1118. [DOI: 10.1126/science.1133687]
*{http://www.sciencemag.org/cgi/content/abstract/314/5802/1118}*

*4. Dayhoff, Judith E., DeLeo, James M.*
Artificial Neural Networks, Cancer Vol 91, 2001
CP: Copyright © 2001 American Cancer Society
Complexity Research Solutions, Inc., Silver Spring, Maryland; Clinical Center, National Institutes of Health, Bethesda, Maryland
1.