

SELF LEARNING SYSTEM FOR ROBOTS

www.technicalpapers.co.nr

Self Learning System for Robots

Introduction:

Motivation for the Paper:

The recent resurgence of interest in neural networks has its root in the recognition that the brain performs computations in a different manner than do conventional digital computers. Computers are extremely fast and precise at executing sequences of instruction that have been formulated for them. A human information processing system is composed of neurons switching at speeds about a million times slower than computer gates. Yet, humans are more efficient than computers at computationally complex tasks such as speech understanding. Moreover, not only humans, but even animals, can process visual information better than the fastest computer.

Statement of NEURO PILOT:

The NEURO PILOT is a simulator or software used to control the activities of a robot pilot based on Artificial Neural Network. This robot perceives the information given to it and then acts accordingly.

To make the pilot to do so it uses backpropogation algorithm (discussed further) and the winner take all group. This NEURO PILOT when implemented with its

hardware and the situation it may be effectively trained and tested as did in the software. The results were displayed in the following sections. This implementation is done in C language and the exe file is created for the verification of the software. The main aim of this project is to minimize the error value encountered at the output and to make the network to behave 100% accurate.

Artificial Intelligence – The backbone of Modern Era:

Artificial intelligence is the process of creating machines that can act in a manner that could be considered by humans to be intelligent. This could be exhibiting human characteristics, or much simpler behaviors such as the ability to survive in dynamic environments. To some, the result of this process is to gain a better understanding of ourselves. To others, it will be the base from which we engineer systems that act intelligently. In either case, AI has the potential to change our world like no other technology.

In its infancy, researches of AI over-promised and under-delivered. The development of intelligence systems in the early days was seen as a goal just in reach, though this never materialized. Today, the claims of AI are much more practical. AI has been divided in to branches, each with different goals and applications.

The problem with AI is that technologies that are researched under the umbrella of AI become common once they are introduced into mainstream products and become standard tools. For example, building a machine that could understand human speech was once considered an AI task.

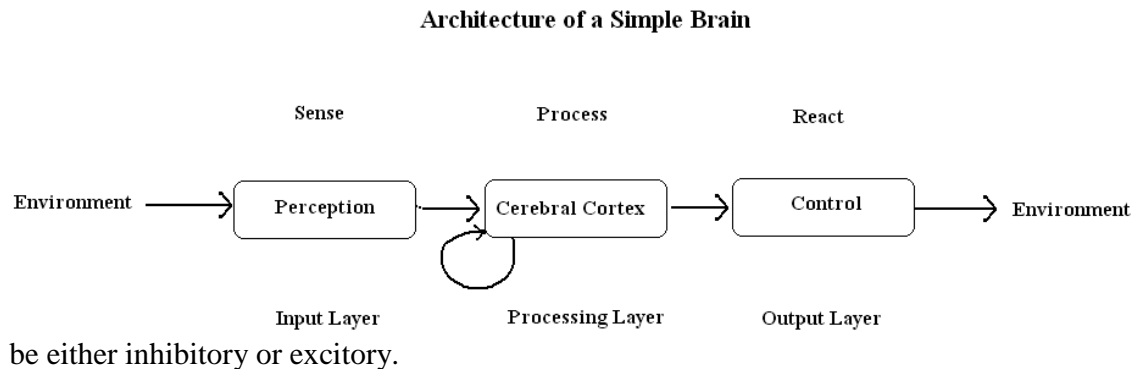
Now the process that includes technologies such as neural networks and Hidden Markov Models are commonplace. It's no longer considered AI. Rodney Brooks describes this as "the AI effect". Once an AI technology becomes utilized, it's no longer

AI. For this reason AI acronym has also being coined “Almost Implemented,” as once it’s done it’s no longer magic, it’s just common practice.

Such an idea was the motivation in creating an artificial neural robot as an pilot named NEURO PILOT.

The Biological Perspective of Neural Networks:

Neural networks are very simple implementations of local behavior observed within our own brains. The brain is composed of neurons, which are the individual processing elements neurons are connected by axons that end at the neuron in the synapse. The synapse is responsible for relaying a signal to the neuron. Synapse can

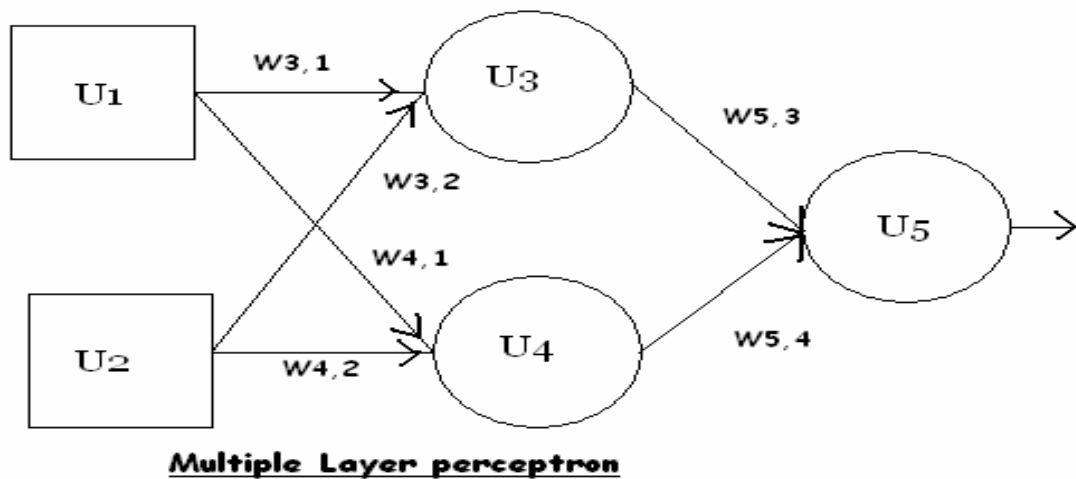


The human brain contains approximately 10^{11} neurons. Each neuron connects to approximately to 1000 other neurons, except in the cerebral cortex where the density of inter neurons connectivity is much higher. The structure of brain is highly cyclic (self-referential), but it can be thought of as having a layered architecture (as in above fig.).

In a very simplified model, the input layer to the network provides our sensory inputs from the environment, the middle layer, are the cerebral cortex, processes the inputs, and the output layer provides motor control back to the environment.

Artificial neural networks attempt to mimic the basic operation of brain. Information is passed between the neurons, and based upon the structure and synapse weights, a network behavior (or output mapping) is provided.

Multiple Layer Networks (A brief Introduction):



Multiple-layer networks permit more complex, non-linear relationships of input data to output results. From the above diagram, we can see that the multiple-layer network is made up of an input layer, an intermediate or hidden layer, and an output layer. The input layer simply represents the inputs to the network, and is not composed of cells (neurons) in the traditional sense. The naming of cells, chosen for this example, gives each cell a u_n identifier. The two input cells are named $[u_1, u_2]$, two hidden cells $[u_3, u_4]$, and one output cell $[u_5]$. Connections within the network is standardized as $w_{3,1}$, and represents the weighted connection between u_3 and u_1 .

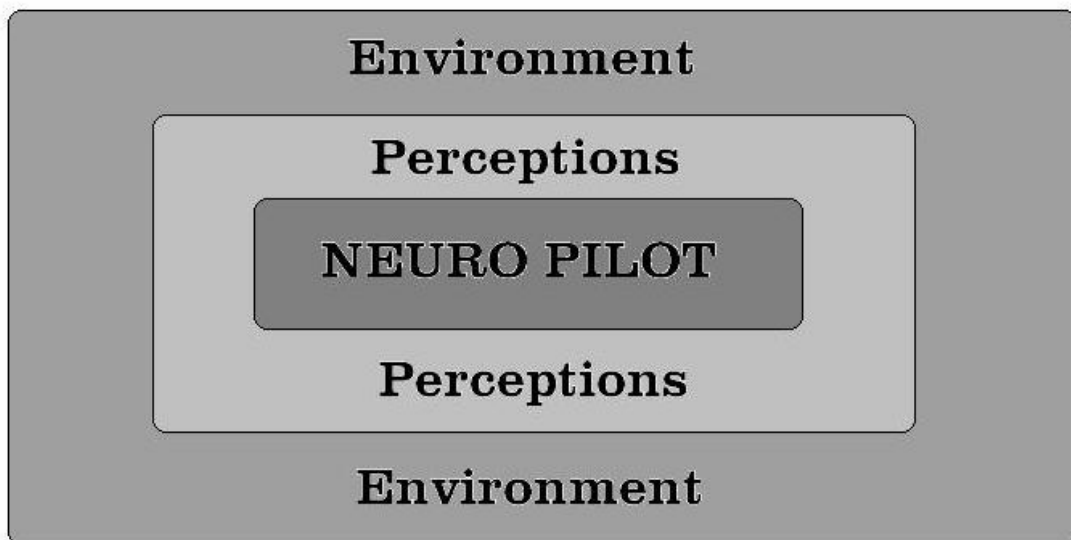
While input cells (u_1 and u_2) simply provide an input value to the network, hidden and output cells represent a function. The result of the sum of products is fed through a squashing function (typically a sigmoid), which results in the output of the cell.

In the further discussion of NEURO PILOT we would be using a multiple layer network since the robot gets information (input) from various source and can produce various outputs.

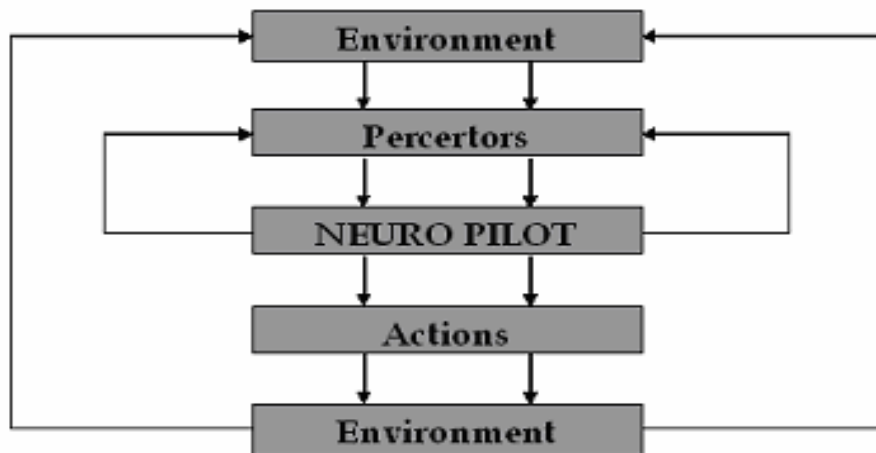
NEURO PILOT Architecture:

The NEURO PILOT Basic Structure:

This NEURO PILOT percept the information from the environment so that the basic block diagram of our NEURO PILOT looks as below.



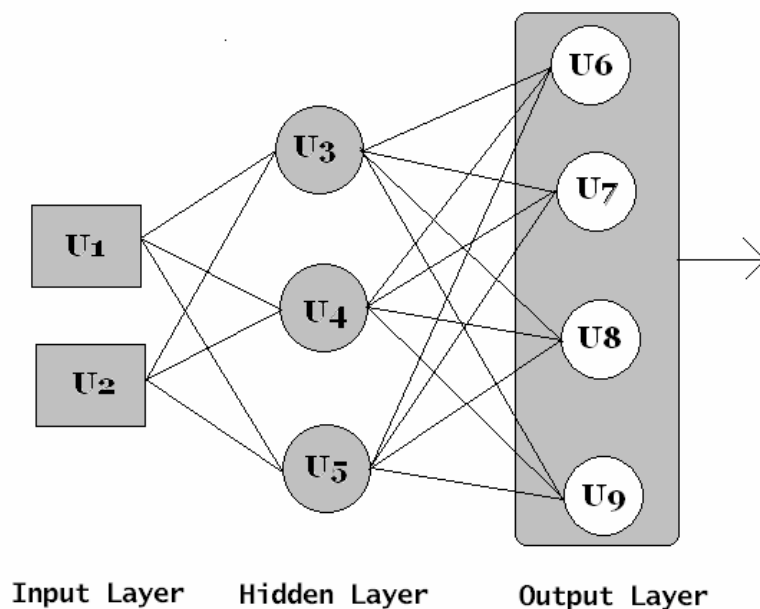
CONTROL FLOW



Winner-Take-All Group:

For the NEURO PILOT agent, we'll look at another architecture called the "winner-take-all" network. Winner-take-all architecture are useful where inputs must be segmented into one of several classes (see in following fig).

Winner-Take-All Group



In the winner-take-all networks, the output cell with the highest weighted sum in the “winner” of the group and is allowed to fire. In our applications, each cell represents a distinct behavior that is available to the robot in the airplane.

Example behaviors include inform to the superiors, fly, failed, information to passengers etc. The firing of a cell within the winner-take-all group causes the robot to perform the particular behavior. When the robot is allowed to perceive the environment, the process is repeated as described.

NEURO PILOT Architecture (in detail):

The network given below is the one used to test the architecture and the method for action-selection. The four points include the *weight* (0 if normal and 1 if overloaded), *Door* (0 if door is locked and 1 if door is unlocked), *Fuel* (0 if empty, 2 if full, 1 otherwise), *Air in the wheel* (0 if it is perfect and 1 if imperfect), *Seatbelts* (0 if all passengers and belted and 1 if any one of the passenger is un belted), *Signal* (0 if received from the control room and 1 if not received from the control room).

The outputs select the particular behavior that the robot will take. Action *Inform superiors* causes NEURO PILOT to inform to its superiors about the situation, *Instruct the passengers* causes the NEURO PILOT to give instructions to the passengers to do so, *Fly* causes the NEURO PILOT to take a drive, *Failed* causes the NEURO PILOT to abort the fly process.

The network employs the concept of backpropagation and the use of winner-take-all group of cells. Thus the complete network designed under a lot of criteria

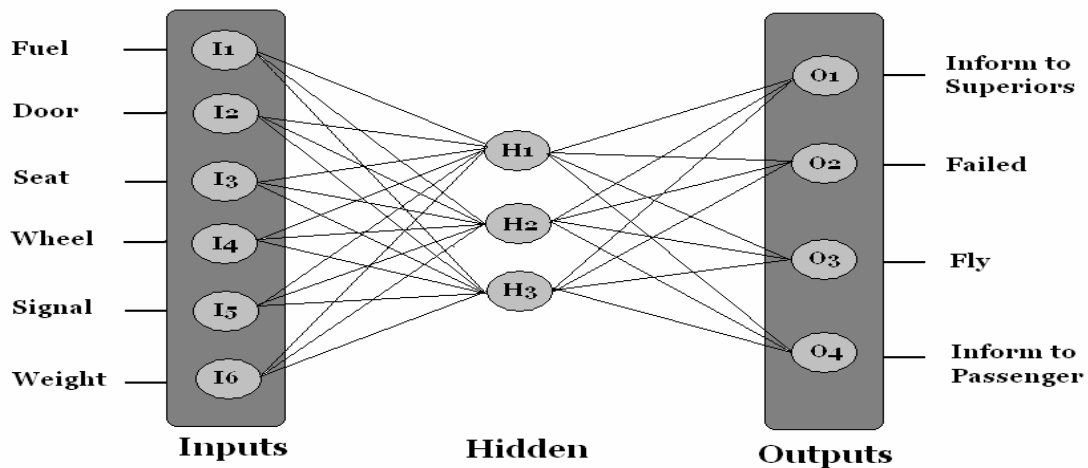
is described below with the mode of selection of various parameters that are very important during network design such as the no of hidden nodes etc.

Thus as a summary, the network employs six input nodes, three hidden nodes, and four output nodes. The network would work efficiently (100%) always but the time taken for training becomes another important factor. So it's again our duty to control the time and resource consumed by the program.

NB:

(In order to avoid the complexity in the implementation some of the input values are taken as 0 or 1)

The Complete Network:



The network contains three layers the *input* layer, *output* layer, *hidden* layer. The no of input nodes are six (weight of the flight, fuel present in the flight, present condition of the door, air present in the wheel, signal status from the control room, status of the passengers seat belts) and the number of output nodes are four (Informing to the higher superiors, instructing the passengers, make an drive and finally to abort the fly

process). The no of hidden nodes is three which is decided under a lot of criteria. Thus from the above figure its clear that the network is multi layered.

The inputs and the outputs may be varied according to their requirements. Here the six inputs and four outputs are decided as an example to demonstrate the Neuro pilot. The implementation is also done considering these inputs and outputs.

NB:

The no of hidden nodes is decided largely by trial-and-error. Three hidden cells could be trained for all presented examples with 100% accuracy. Decreasing the cells to two or one resulted in a network that did not successfully classify all the examples.

Backpropogation Algorithm:

The algorithm begins with the assignment of randomly generated weights for the multi-layer, feed-forward network. The following process is then repeated until the mean-squared error (MSE) of the output is sufficiently small:

- 1) Take a training example E with its associated correct response C.
- 2) Compute the forward propagation of E through the network (compute the weighted sums of the network, S_i , and the activations, u_i , for every cell).
- 3) Starting at the outputs, make a backward pass through the output and intermediate cells, computing the error values (given below.)

$$\begin{aligned} \delta_i &= (C_i - u_i) u_i (1 - u_i) && \text{For the output cells} \\ \delta_j &= (\sum_m w_{mj} \delta_m) u_j (1 - u_j) && \text{For all hidden cells} \end{aligned}$$

(Note that m denotes all cells connected to the hidden node, w is the given weight vector and u is the activation).

- 4) Finally, the weights within the network are updated as follows.

$$w_{ij}^* = w_{ij} + \eta \delta u_i \quad \text{For weights connecting hidden to output.}$$

$$w_{ij}^* = w_{ij} + \eta \delta_i u_i \quad \text{For Weights connecting hidden to input}$$

Where η represents the learning rate (or step size). This small value limits the change that may occur during each step.

5) The error value that is encountered at the output each time can found using the following formulae.

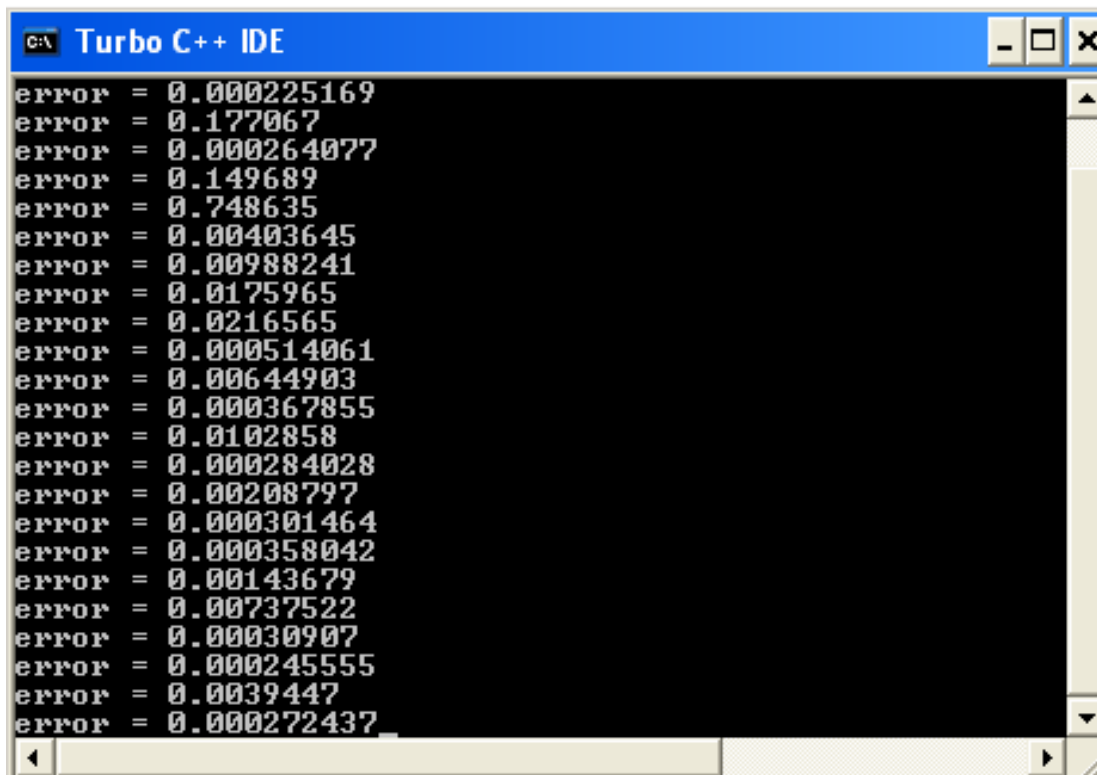
$$\text{Err} = 0.5 * (\text{O}_{\text{desired}} - \text{O}_{\text{actual}})^2$$

Where $\text{O}_{\text{desired}}$ is the desired output that the user needs, while O_{actual} is the actual output got through the network.

NEURO PILOT Training:

The NEURO PILOT within its environment is a static element of the pilot AI. The following discusses online learning of the NEUROPILOT within the environment. Training of NEURO PILOT consist of presenting training examples from a small set of desired behaviors, and then performing the backpropagation on the network given the desired result and the actual result. For example, if the NEURO PILOT is in possession of a fuel and all other inputs are perfect then, the desired behavior may be to fly. However, if the fuel is full, in possession of knife, and all other inputs are perfect other than the signal then, the correct behavior would be to inform to abort.

The sample screen of the training errors in the output for each time is given below.

A screenshot of the Turbo C++ IDE window. The title bar reads "C:\ Turbo C++ IDE". The main window area is black with white text displaying a list of 20 error values. Each line starts with "error =" followed by a decimal value. The values are: 0.000225169, 0.177067, 0.000264077, 0.149689, 0.748635, 0.00403645, 0.00988241, 0.0175965, 0.0216565, 0.000514061, 0.00644903, 0.000367855, 0.0102858, 0.000284028, 0.00208797, 0.000301464, 0.000358042, 0.00143679, 0.00737522, 0.00030907, 0.000245555, 0.0039447, and 0.000272437. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

```
error = 0.000225169
error = 0.177067
error = 0.000264077
error = 0.149689
error = 0.748635
error = 0.00403645
error = 0.00988241
error = 0.0175965
error = 0.0216565
error = 0.000514061
error = 0.00644903
error = 0.000367855
error = 0.0102858
error = 0.000284028
error = 0.00208797
error = 0.000301464
error = 0.000358042
error = 0.00143679
error = 0.00737522
error = 0.00030907
error = 0.000245555
error = 0.0039447
error = 0.000272437
```

NEURO PILOT learning:

When the NEURO PILOT is embedded in its environment (as an pilot), its training is complete and no further learning is possible. To include the ability for the NEURO PILOT to learn, portions of backpropagation could be included with the cell to adjust the weights based upon its environment. A very simple mechanism could adjust the weights of the NEURO PILOT based upon the last action made it. If the action led to negative consequences, such as the failure, the weights for this action given the current environment could be inhibited to make it less likely to occur in the future. Further all such AI pilots could learn these same lessons, in a form of Lamarckian evolution (whereby children inherit the traits of their parents, which would include lessons learned). After numerous decisions made, the Neuro pilot would become slowly better at avoiding negative situations.

NEURO PILOT Memory:

The feature of memory could also be created within the neural network by creating tapped-delay lines for each of the inputs (extending the input vector from one to two dimensions). Therefore, the prior input from the environment doesn't disappear, but becomes part of another input to the network. This method could also include feedback. So that the last actions could be fed back into the network providing the NEURO PILOT with action history. Additional internal feedback could include the NEURO PILOT to perceive the best output for the given inputs. These mechanisms further the generation of rich and believable action selection.

Conclusion:

Future Enhancements:

“Nothing is invented and perfected at the same time”

We are interested in implementing this NEURO PILOT in reality and test it under many circumstances. There are a lot other application of backpropogation to be used in the same NEURO PILOT such as the speech reorganization when a robot wants to inform to its superiors. Backpropogation can also used with this NEURO PILOT to implement the security system inside the flight (perceiving the terrorists) This NEURO PILOT is just the simulation of the robot pilot. As the above proverb goes this system can be perfected but takes its time.

Reference:

www.technicalpapers.co.nr