Explorations in Machine Learning, Sensory Augmentation and Fuzzy Set Theory, and their realization in Autonomous Driving Systems

Project Report Submitted to Mangalore University
for the award of degree of

Master of Science in Physics

By

Prajwal DSouza

Reg No : 169724037

Under the supervision of

Dr. K.M. Balakrishna

Professor in Physics

Department of Studies in Physics
Mangalore University, Mangalagangotri – 574199
Mangalore.

May 2018

Department of Studies in Physics
Mangalore University

Mangalagangothri - 574 199, India

Prof. K.M. Balakrishna
Professor, Department of Physics

Certificate

This is to certify that the project titled “**Explorations in Machine Learning, Sensory Augmentation and Fuzzy Set Theory, and their realization in Autonomous Driving Systems**” is carried out by **Mr. Prajwal DSouza**, under my guidance and regulation. The candidate is herewith permitted to submit this project to Mangalore University, Mangalagangothri, for the award of Master of Science in Physics.

The information furnished in this project report has not been submitted elsewhere for the award of any Degree, Diploma, Associateship, Fellowship or any other University or Institution.

Mangalagangothri
Date :   Project Guide

Examiners

1.   2.

Acknowledgements

I am indebted to my guide, Prof. K.M. Balakrishna for his constant support and guidance which helped us solve many challenges that I faced during the project.

I thank Mr. Shawn Ajay D’Souza, Department of Physics, St. Aloysius College, for helping with the setting up of the self-driving system.

I am eternally thankful to Mr. Vanamali C. Shastri, Research Scholar, Mangalore University for his constant support and guidance.

I am thankful to Ms. Akshatha Shanbhag and Ms.Fatima Shaguftha who helped us in making the tracks for training and testing the RC car.

I thank my aunt Mrs. Geetha Monterio for letting me borrow her son’s remote controlled toy car and then rip it apart for the work.

I thank Mr. Rohan Pinto for all his help during the project.

Finally, I thank my project partners Mr. Vinton Rebello, Mr. Nihal T J and Ms. Manjula J S for their valuable input and dedication towards the project.

(This page intentionally left blank)

Contents

Introduction

Artificial Intelligence (AI) as a field of study originated with the goal of imitating the human-like intelligence in machines, that is, to learn, “guess”, self-correct and in every way imitate a human intelligence. In computer science AI research is defined as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving". [a] Capabilities generally classified as AI as of 2017 include successfully understanding human speech, [b] competing at the highest level in strategic game systems (such as chess), **autonomous cars** etc. Many tools are used in AI, one that is of particular interest to us in this paper is that of **neural networks** (the meaning and concepts of which will be explained in detail later in this paper).

A typical AI perceives its environment and takes actions that maximize its chance of successfully achieving its goals. [a] An AI's intended goal function can be simple ("1 if the AI wins a certain game, 0 otherwise") or complex ("Do actions mathematically similar to the actions that got you rewards in the past"). Goals can be explicitly defined or can be induced. If the AI is programmed for "reinforcement learning", goals can be implicitly induced by rewarding some types of behavior and punishing others. The reader immediately realizes that the behavior of an AI is no different than that displayed by humans and other animals and in fact, AI finds the inspiration for its inception from the biological systems.

Sensory substitution, (discussed later in this paper) in its nascent stages aimed at supplementing one sensory modality with another. But, with the advancements in both the aforementioned fields of study they have come to be applied in various other fields, one of the applications being as a tool in aiding the study of human cognition. One of the consequences of studies in human cognition aided in parts by application of artificial intelligence (AI) and Sensory Substitution is a realization that human mind is “fundamentally not a logic engine but an analogy engine, a learning engine, a guessing engine, an aesthetics-driven engine and a self-correcting engine” [c]

To date, the concepts in artificial intelligence and sensory substitution, though studied separately, haven’t been studied in unison. I have, in this paper, therefore, proposed a model that merges the two. In this project, the data is sampled from the hidden layer of the neural network with the aim of training the human brain to identify the decision to be taken by the neural network.

Machine Learning

Introduction

There are tasks that computers do better than humans, and, there are tasks at which humans are better.

Computers are undoubtedly faster and accurate in instances that require calculations of enormous size or length or when one needs to execute a set of instructions in a particular sequence. So, given a task, the conventional procedure is to break the task into many parts, analyze each part individually, generate an instruction, and finally write these instructions in a program which the computer would then execute.

Large computations are undoubtedly hard for humans, but, there are certainly a few tasks that humans would do with a great ease---recognizing objects, faces, understanding a rhetoric, writing poetry etc. “Recognizing” or “making sense “of the image given below is an extremely tedious task computationally. The question that a researcher would ask is: given any image (which basically is an array of numbers representing pixel intensities), how would one decide if the given matrix of numbers represents an image of a dog or that of a cat? The question that immediately follows is: how then may one create a computer that can do these tasks besides the tasks performed by a conventional computer? The solution is definitely not in the conventional instruction-based methods!

[1]

$\longrightarrow$

```
116 115 115    ...    122 123 122
117 117 118    ...    123 121 123
117 117 117    ...    124 121 123

...
...

20  22  21     ...    237 237 239
20  22  19     ...    239 236 238
18  17  18     ...    237 238 238
```

Figure (1.1)

Writing instructions to “recognize” an image from the set of numbers representing pixel intensities is next to impossible for an image could have millions of pixels, and also, not all dog images would necessarily have to be the same[a].

[a] The dog could be in a different part of the image, it’s orientation in the image could be different, it could be of a different breed altogether.

There is something besides image recognition that humans are good at – The game of Chess!

Claude Shannon, the father of information theory, showed that the lower bound for game-tree complexity of chess to be 10^{120}, now known as the Shannon number [2]. This showed that writing a program using brute force method to analyse every possible move would be impossible. But, humans have intuition! Through experience a human knows that a certain move in chess would be more profitable than the others, which begs the question: how does one program intuition into a computer?

There are many tasks, for instance, telling if a particular email is a spam, also, identifying a tumor in an MRI that turn out to be tedious tasks by themselves. The difficulties faced by systems relying on hard-coded knowledge suggest that AI systems need the ability to acquire their own knowledge, by extracting patterns from raw data. This capability is known as **Machine Learning**. [3]

On Machine Learning

Two popular machine learning problems are **classification** and **regression**. Classification problems are those where the output is discrete quantities and regression is when the output is continuous. Predicting if an email is a spam is a classification problem, whereas, predicting the height of the person from an image is a regression problem as height is a continuous number.

In this project, we will be focusing on classification problems such as what should a car do when it faces an obstacle in front of it on the road?

In classification problems, machine learning involves a training session where the program explores a dataset, finds patterns in it using certain **classifiers**. There are many classifiers and picking the classifier depends on the type and size of the dataset to be analysed.

Machine learning can be also classified into two types based on the learning approaches.

Supervised learning involves a labelled dataset. There is some input that the user gives to the system, and its corresponding output is also specified by the user. This is like pointing out the image of a dog to a child and then telling it to call it a 'dog'. Of course, to train the system, we also show it thousands of images of cats and dogs, where each image is labelled with a class category such as ‘dog’ or a ‘cat’. This involves a lot of algorithms such as Support Vector Machines, Naive Bayes Classifiers and Decision Trees etc.

There is another form of supervised learning called the Reinforcement learning - a technique finding inspirations from psychology. This technique involves an agent and an environment. The agent acts, the actions of the agent lead to certain events in the environment, and, if the action is “right”, leads to a reward, else, the agent is given a negative reward (punishment). The agent then learns based on this reward-based system. [4]

Unsupervised learning involves an unlabeled dataset. There is only a dataset which hasn’t been labelled. The system has to learn to differentiate inputs and assign labels on its own, to find the hidden structure in the unlabeled data. This is like having a lot of images of dogs and cats and asking someone who has never seen a dog or a cat, to separate the images into certain categories. This involves algorithms such as K-Clustering.

Consider a binary classification problem.

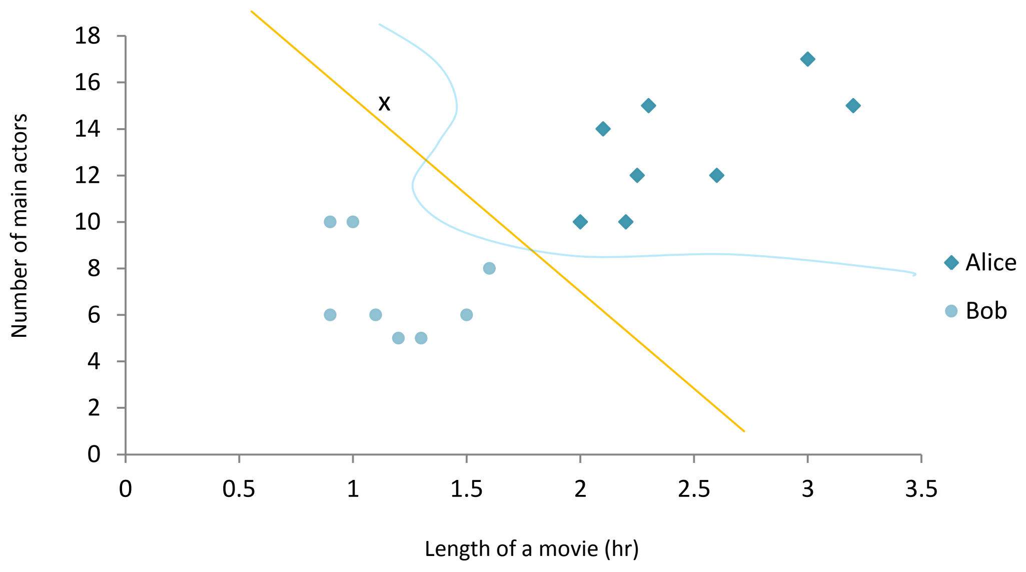


Figure (1.2) : A binary classification and the possible decision boundaries. Linear and Non-linear decision boundary.

The above is set of movies, Alice and Bob like. It’s a little evident that Alice likes longer movies compared to Bob and that Bob prefers movies with lesser actors. Now given a movie ‘x’, Who will like the movie?

The simplest approach would be drawing a line separating both the classes. Here each point is two dimensional therefore a line can separate them. This line is called the **decision boundary**. Since the movie 'x' falls on the right side of the decision boundary, the classifier will predict that Alice will like the movie. The decision boundary need not be linear (line or a hyper-plane in higher dimensional data). Consider a curved decision boundary in the above case. Here the movie 'x' will be classified as the one Bob would like. So, the decision boundary drawn will depend on the type of classifier used and its parameters.

Sometimes drawing a non-linear decision boundary becomes important.

Consider the XOR problem.

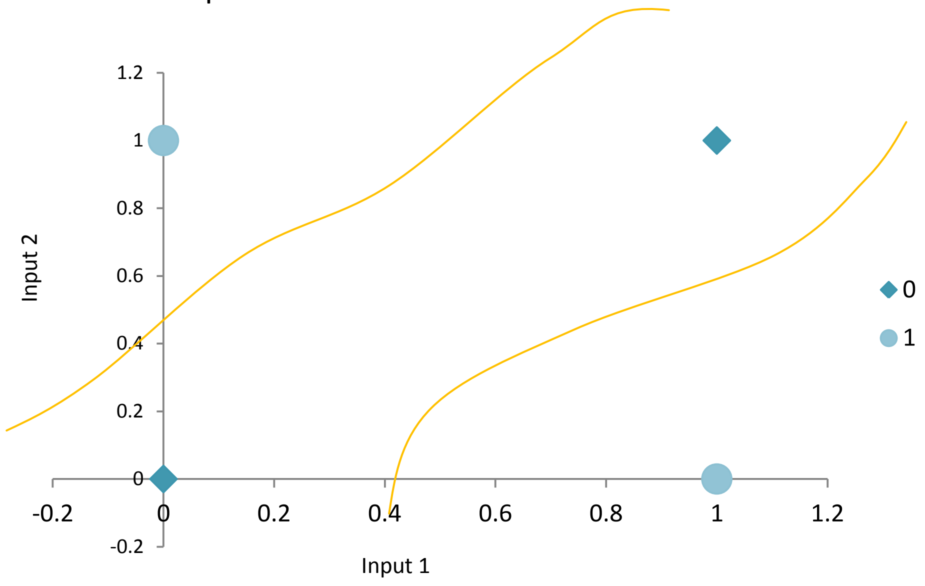


Figure (1.3) : XOR Classification decision boundary.

Single simple linear decision boundary isn't possible in this case. There are methods called Kernel methods used in SVM classifiers which can draw non linear boundaries. There are also a set of classifiers that do this well. **Neural Networks**, more precisely, **Artificial Neural Networks** (ANN) are classifiers which are known to handle non-linear classification boundaries.

Initially, **Neural Networks** were just simple single layered networks called **Perceptrons**. Perceptrons take some input vector, take a weighted sum of the input values and output a value based on which class the input belongs to.

In 1969, Minsky and Papert showed that basic perceptions (two layered neural network) could not learn the XOR circuit and also that neural networks needed a lot of computational power. This was because there weren't many hidden layers.

In 1975, Werbos's Backpropagation algorithm solved the XOR circuit using Neural network. Backpropagation will be discussed later in the paper.

Over time, we have seen many variants of simple multilayer neural networks. Recurrent Neural Networks (RNN) have shown promise in handling time series data such as sound etc. It has helped develop voice to text conversions which have made voice assistants possible.

Convolutional Neural Networks (CNN) have shown promise when it comes to classification of images. In fact, some CNN's have an accuracy greater than that of humans on image datasets. CNN's were inspired from the mammalian vision system. It involves certain layers which perform convolution on the input image.

For most part of this project, I will be focusing on multilayer feed forward neural networks.

Neural Networks

Architecture

Neural network architecture is as follows.

A **Neural Network** consists of many layers. The first layer is called the input layer where usually a high dimensional data is fed into as a vector. At the end, there is an output layer. In between, there are many hidden layers. (In the figure (2), there are two hidden layers).

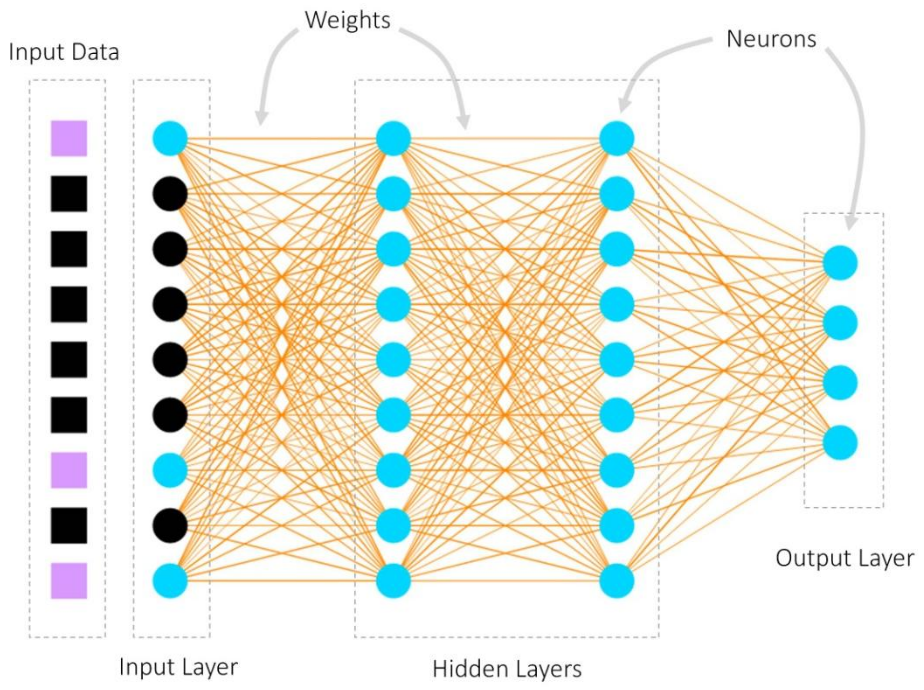


Figure (2.1) : The neural network architecture along with the input data.

Each layer consists of many **neurons**. A **neuron** is a non-linear unit. Each neuron in a given layer is connected to all the neurons in the previous layer. These connections are weighted, that is, the inputs from the previous layer are multiplied by corresponding weights. Each neuron has a value called its **activation**. Each Neuron takes the weighted sum of the neuron activations of the previous layer. This weighted sum then becomes the input to a non-linear function called the **activation function**, whose output is the activation of the current neuron.

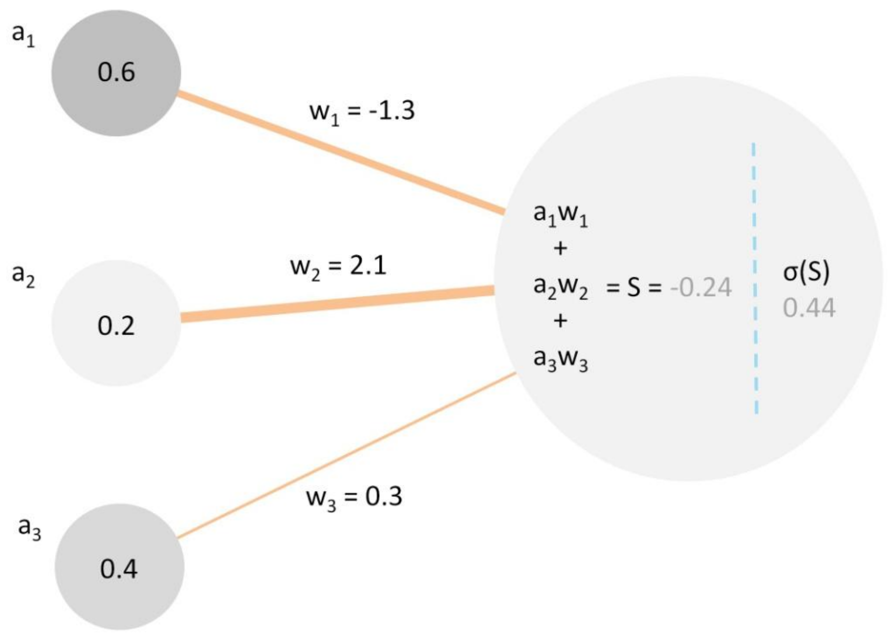
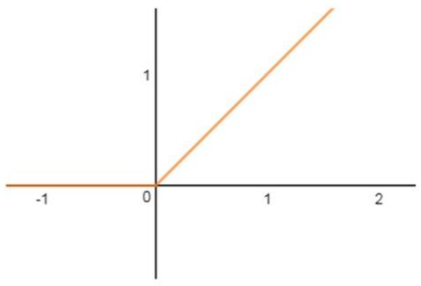


Figure (2.2) : Computations of a single neuron.

In the case depicted above, there are three neurons in the previous layer with activations a_1, a_2 and a_3. There is a single neuron in the next layer. This neuron is connected to the neurons in the previous layer via weights w_1, w_2 and w_3. The neuron then takes the weighted sum $S = a_1w_1 + a_2 w_2 + a_3w_3$. The sum S is input to a nonlinear activation function, here, the Sigmoid Function σ. The σ(S) gives the activation of this neuron.

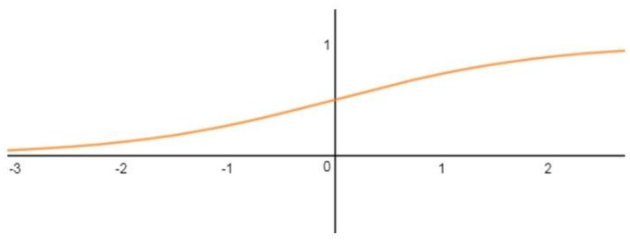
There are various notations used to represent neurons in various layers. But, the simplest are chosen here.

There are many Activation functions. Common ones include Sigmoid, ReLU (Rectified Linear Unit), Identity and Tanh functions. The activation functions make the system non-linear. Sigmoid is usually used in the output layer in case of classification problems. Its output is in between 0 and 1.



$$\operatorname{Re}LU(x) = \begin{cases} x & ; x \geq 0 \\ 0 & ; x < 0 \end{cases}$$

Rectified Linear Unit



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Sigmoid

Figure (2.3) : Graphs of popular activation functions. ReLU and Sigmoid functions

Each layer, except the output layer, will also have a Bias Neuron whose activation is usually set to 1.

Why hidden layers?

The hope is that given an input, once trained, the network will start extracting higher level features from the input which correspond to activations of particular neurons in the hidden layers. In case of a hand written digit, the hope is that the hidden layer of the neural network will learn high level features such as the arcs of a number 9 or number 5 which would hopefully correspond to activation of specific neurons in the hidden layer corresponding to the arcs in the image.

The input (usually a high dimensional vector) is given to the input layer, which then activates the hidden layers, finally leading to the activation of the output layer neurons. This gives us the output of the network for the given input. This process is called the forward pass.

Each weight and bias in a neural network is a parameter which can be varied to get desired outputs. In Machine learning, in most cases, we are trying to fit a function into a dataset. This function has many parameters. In case of a neural network there are thousands. So, training a network is about figuring out the right parameters (weights and biases) that give us the correct outputs.

Training a Neural Network

In Machine learning, we often define a loss function. Loss function is usually the difference between the expected output and the actual output.

If the output of the neural network is 0.8, when it must be 1, the difference 0.2 is the error which needs to be minimized. This forms the loss function whose minimum is to found. This is now an optimization problem.

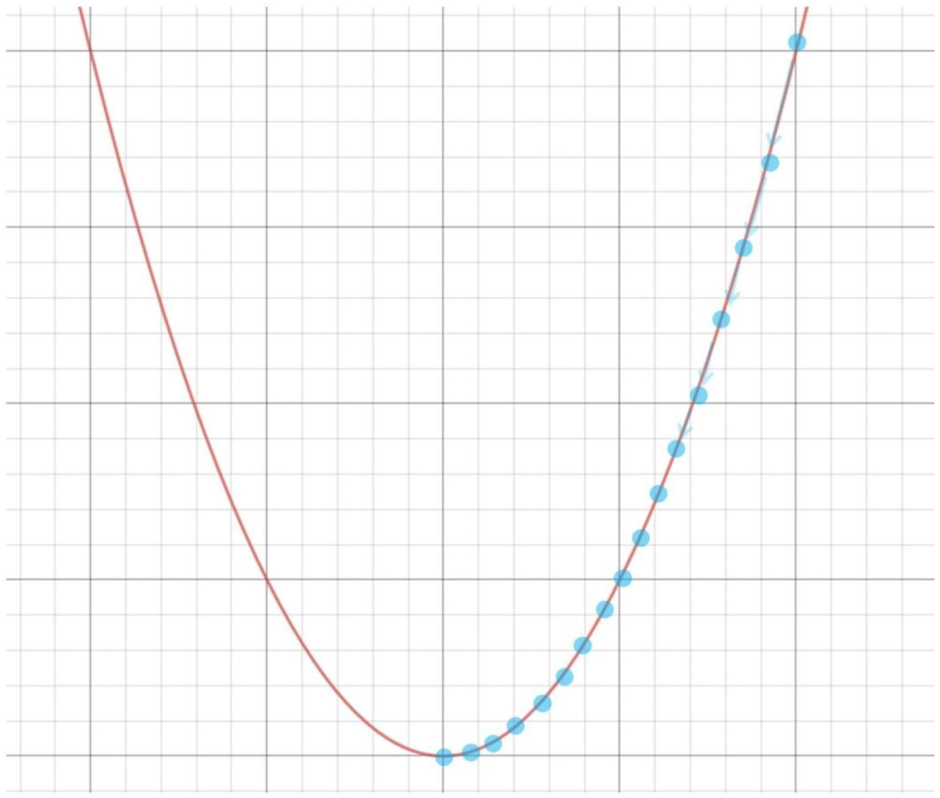


Figure (2.4) : Gradient descent approaching the minimum of the loss function.

The global minimum of the loss function has to be found.

There are many algorithms to train the classifiers. One of most basic ones is called the **gradient descent**.

In Gradient descent, we start at some value of the loss function, calculate the gradient of the loss function which then points us in the direction of the minimum.

Iteratively small steps are taken towards the minimum calculating the gradients at each of the points.

Calculating the gradient involves the calculation of the partial derivatives of the parameters of the classifier. In case of the neural network, it would be the weights and the biases. In case of a neural network this approach is called **backpropagation**.

Backpropagation involves a forward pass followed by a backward pass. On forward pass the neural network gives an output (actual) which is then compared with what the output (expected) must be. The difference in the actual output and the expected output is taken as the error. The error is then propagated backwards from the output layer into the hidden layers and the corresponding updates in the weights are made to correct the error. This correction leads to a step towards the minimum of the loss function.

Backpropagation stands for backward propagation of errors. Backpropagation was developed by Henry Kelly and Arthur Bryson in the 1960's. Gradient Descent need not lead to a global minimum which we desire. It might go to a local minimum and stay there. There have been many variants of backpropagation such as R-prop, Adam Optimizer etc, which can help escape a local minimum.

Other methods involve Genetic algorithms which help reach the global maximum.

Self-Driving Cars

Much of the technology seen in Waymo [*]- Google's autonomous car project, is already seen on the road. We're familiar with auto-braking systems, auto-parallel-parking systems that are direct consequences of proximity sensor technology. When the proximity sensor technology is combined with auto-steering used for auto-parking and the cruise-control technology, we basically get a loose framework for a self-driving car.

While it may seem like "self-driving" is a recent mantra, truth is quite from it. Automation had been tried in the year 1925 where the car driven was remote-controlled. The radio-controlled car could start its engine, shift its gears and sound its horns "as if a phantom hand were at the wheel".

Not much progress had been made until the early 1990s when Carnegie Mellon researcher Dean Pomerleau writes a PhD thesis, describing how neural networks could allow a self-driving vehicle to take in raw images from the road and output steering controls in real time. The same technique has been employed in our current project with a new twist.

The current Waymo car employs 8 sensors, the most prominent being a LIDAR (Light Detection and Ranging) device. LIDAR works much like radar, but instead of sending out radio waves it emits pulses of infrared light - aka lasers invisible to the human eye - and measures how long they take to come back after hitting nearby objects. It does this millions of times a second, then, compiles the results into a so-called point cloud, which works like a 3-D map of the world in real time - a map so detailed it can be used not just to spot objects but to identify them. Once it can identify objects, the car's computer can predict how they will behave, and thus how it should drive.

Self-driving cars have a lot of classification to do. Input could be the image of the road, the LIDAR data, the GPS data. Based on the input it has to take a correct action such as turning left, right, accelerating or braking.

* visit: https://waymo.com/ for more information on the topic.
Much of the information on this page has been borrowed from the following websites:
1. https://www.wired.com/story/lidar-self- driving-cars- luminar-video/
2. https://www.digitaltrends.com/cars/history-of- self-driving- cars-mileston
3. https://en.wikipedia.org/wiki/Autonomous_car

This is a lot of data, and the AI does decisions quickly. But, most important aspect is to make sense of why the AI took a particular decision on the road. If one uses neural networks in Self driving systems, then the decision may not be obvious. So, we think, that feeding the data that the AI processes to humans can be a way of understanding an AI’s decisions.

This is best done via **Sensory Augmentation**.

Data and Sensory Substitution

From an evolutionary perspective, whether you are a predator or prey, dying is something to be avoided. Predator constantly needs to find prey and the prey must figure out ways to avoid predators. Both needed sensory systems to do this. More data these systems collect the better. But more data needs a better system for analysis, a bigger brain. More data also means collecting only relevant data. Visual Cortex finding relevant patterns from the data transmitted from the eyes.

Data is everywhere, our sensory systems constantly pick it up, let the brain process most of it and finally make sense out of it. We understand the world, because we experience it via our senses. It's these experiences that lead to our vocabulary. Hence, augmenting more senses should help us understand the world better. Seeing (experiencing data) in Infrared region could change the way we recognize objects. Consider the approach of Wearable media (https://www.wearablemedia.studio/), a company that recently developed a vibro-tactile wearable that vibrates based on the asteroids that are in near earth orbit. It makes people feel part of a larger reality.

Experiencing more data is important. Adding more senses to humans via sensory substitution seems to be a way forward in this regard.

Sensory substitution is a change of the characteristics of one sensory modality into stimuli of another sensory modality. [5] The idea of sensory substitution was originally described by Paul Bach-y-Rita and colleagues in their seminal work", "Vision Substitution by Tactile Image Projection" which was in the development of visual substitution system for the blind and "as a means of studying the processing of afferent information in the central nervous system".[6] Sensory substitution can occur across sensory systems, such as touch-to-sight, or within a sensory system such as touch-to-touch. In one experiment, the touch sensory information via a glove containing artificial contact sensors was coupled to skin sensory receptors on the forehead of a person who had lost peripheral sensation from leprosy. After becoming accustomed to the device, the patient experienced the data generated in the glove as if they were originating in the fingertips, ignoring the sensations in the forehead [7a].

The most successful sensory substitution system to the present is Braille. Information usually acquired visually is, instead, acquired through the fingertips. A blind person using a cane is exhibiting another very successful simple sensory substitution system; the single point of contact with an object provides a great deal of practical information on location and identification [7b].

Sensory Augmentation

Human beings, through the faculty of five senses receive raw information about the world, interpret it and “experience” a certain reality. Could this “experience” be made richer? Could we “see” more than we can presently? Building upon the research conducted on sensory substitution, investigations into the possibility of augmenting the body's sensory apparatus are now beginning. The intention is to extend the body's ability to sense aspects of the environment that are not normally perceivable by the body in its natural state. [d]

Computers, as well as, human beings are limited in their own ways as discussed in the prequel of this paper. What if we could feed the data from any device (in our case, an autonomous vehicle), either in a raw form or in a processed form to a human being through the faculty of any one of the senses or more and the human being could process the data, interpret it and then feed it back to the system (after having converted to a form understandable by the machine through an intermediate device)? we’d have essentially solved the problem presented as unanswerable questions in the beginning of the paper!

One may inquire if the brain is capable of interpreting the data and producing a meaningful response to the same. It turns out that the human brain is quite capable of adapting to changes procured due to an addition of a sense or a loss of one. [8] [9] [10]

There have been successful attempts of a human-computer interfacing in literature [11]. So, this is not science fiction but something that can be achieved in the near future.

---

[d] Active work in this direction is being conducted by, among others, the e-sense project of the Open University and Edinburgh University, and the feelSpace project of the University of Osnabrück.
Link to feelSpace: http://feelspace.cogsci.uni-osnabrueck.de/

Experiment

Explore a simple self-driving system trained using a Neural Network

Since we wanted to study actions and possible sensory augmentation is self driving systems, we search for approaches to build simple self driving systems where AI implementation was possible. We came across a small project by David Singleton (http://blog.davidsingleton.org/nnrccar/).The project involved training a Remote Controlled Car (RC Car) to drive based on a video feed.

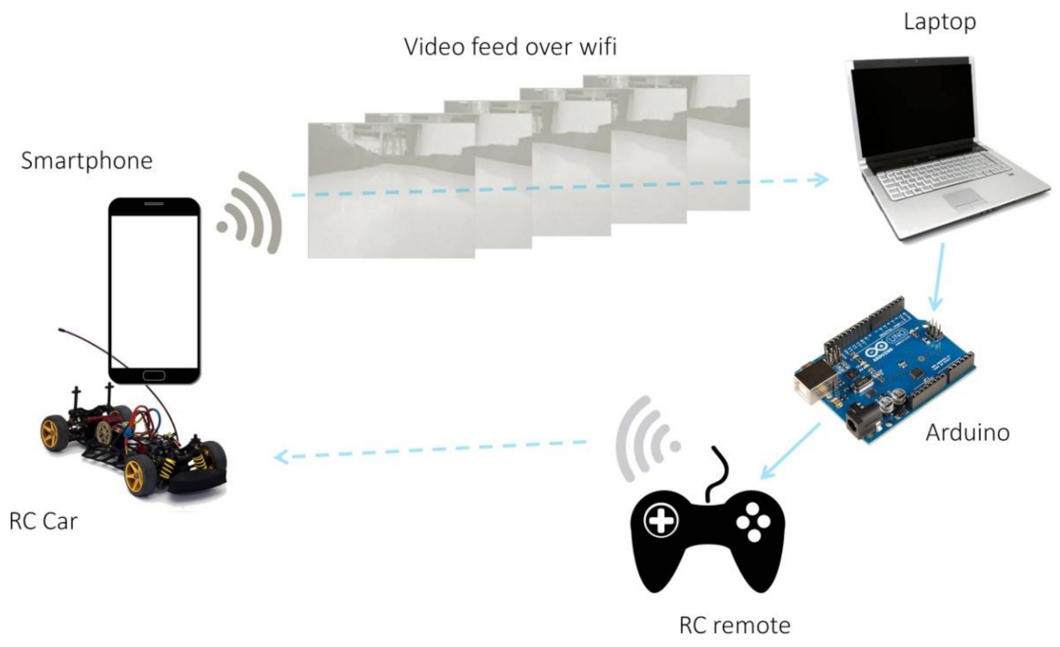


Figure (5.1) : The set-up of RC car control system.

Instrumentation

There were many software and hardware requirements to achieve the above.

Remote controlled car

We used a simple remote controlled car which came with a simple joystick like remote. The Remote controlled Car (RC Car) we used had a slight complication. Its front wheels weren't aligned well and hence would not go forward for a large distance. It would deviate from the path and turn left on forward instruction. This was party ignored as our interest was in the learning algorithm than the problems the car had maneuvering.

Arduino

Arduino consists of an Atmel 8-bit AVR microcontroller. It can be easily interfaced with python and communicate via the serial port. This would let the computer do the difficult work of processing images using python and then transmit actions to the arduino which would then drive the RC remote buttons.

Optocouplers

To drive each button on the RC remote, instead of using a direct connection from the arduino pin, there was an optocoupler used to assist the switching. This was much more simpler.

Opto-isolators or Opto-couplers, are made up of a light emitting device, and a light sensitive device, all wrapped up in one package, but with no electrical connection between the two, just a beam of light. The light emitter is nearly always an LED. The light sensitive device may be a photodiode, phototransistor, or more esoteric devices such as thyristors, TRIACs etc.

A lot of electronic equipment nowadays is using opt coupler in the circuit. An opt coupler or sometimes refer to as opt isolator allows two circuits to exchange signals yet remain electrically isolated. This is usually accomplished by using light to relay the signal. The standard opt coupler circuits design uses a LED shining on a phototransistor-usually it is an NPN transistor and not PNP. The signal is applied to the LED, which then shines on the transistor in the IC.

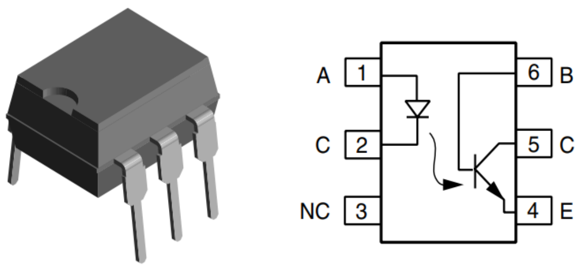


Figure (5.2) : MCT2E Opto-coupler. (http://projectpoint.in/datasheets/pdf/mct2e.pdf)

We used a MCT2E optocoupler IC. The MCT2E series of opto-coupler devices each consist of gallium arsenide infrared LED and a silicon NPN phototransistor. They are packaged in a 6-pin DIP package and available in wide-lead spacing.

When logic zero is given as input then the light doesn't fall on transistor so it doesn't conduct which gives logic one as output.

When logic one is given as input then light falls on transistor so that it conducts, that makes transistor switched ON and it forms short circuit this makes the output is logic zero as collector of transistor is connected to ground.

IP Camera

To transmit data wirelessly, we used a wonderful piece of software developed by Pavel Khlebovich, IP Webcam (https://play.google.com/store/apps/details?id=com.pas.webcam). It streams video from a phone camera via wifi at a particular IP address which can then be accessed by a computer by reading the video feed at that address.

We used a python code discussed at a StackOverflow thread (http://stackoverflow.com/questions/21702477/how-to-parse-mjpeg-http-stream-from-ip-camera)

This would read the video stream at the IP address and then with the help of OpenCV we decode it into a two dimensional array image data. The video stream was kept at the lowest resolution available. This had two advantages. The stream wouldn't have buffering issues over the wifi as the bit rate would be less. Also, the lower resolution data was good enough and low dimensional, making all the post processing tasks faster.

Python Libraries

There are many libraries used to handle processing. Some prominent ones include

OpenCV is a computer vision library available in python for processing images. It was used here for multiple purposes such as conversion of RGB image to grayscale image, to display the video feed etc.
(https://docs.opencv.org/3.4.1/d0/de3/tutorial_py_intro.html)

Sklearn is a machine learning library in python which has a lot of classifiers and can be used to implement neural networks.
(http://scikit-learn.org/)

PyBrain is a machine learning library that was finally chosen for its simplicity in implementation of the neural network. Our interest was the hidden layers of the neural network and this library offered simple access to them.
(http://pybrain.org/)

Other ML libraries were tried upon such as Keras, Tensorflow. But, the need for 64 bit system made us ignore Tensorflow.

Experimental Setup

The RC car remote buttons were connected to the optocouplers. The Optocouplers could be switched on using arduino. A resistor of 220 Ω was used between the input of the optocoupler and the arduino pin to limit the input current to the optocoupler.

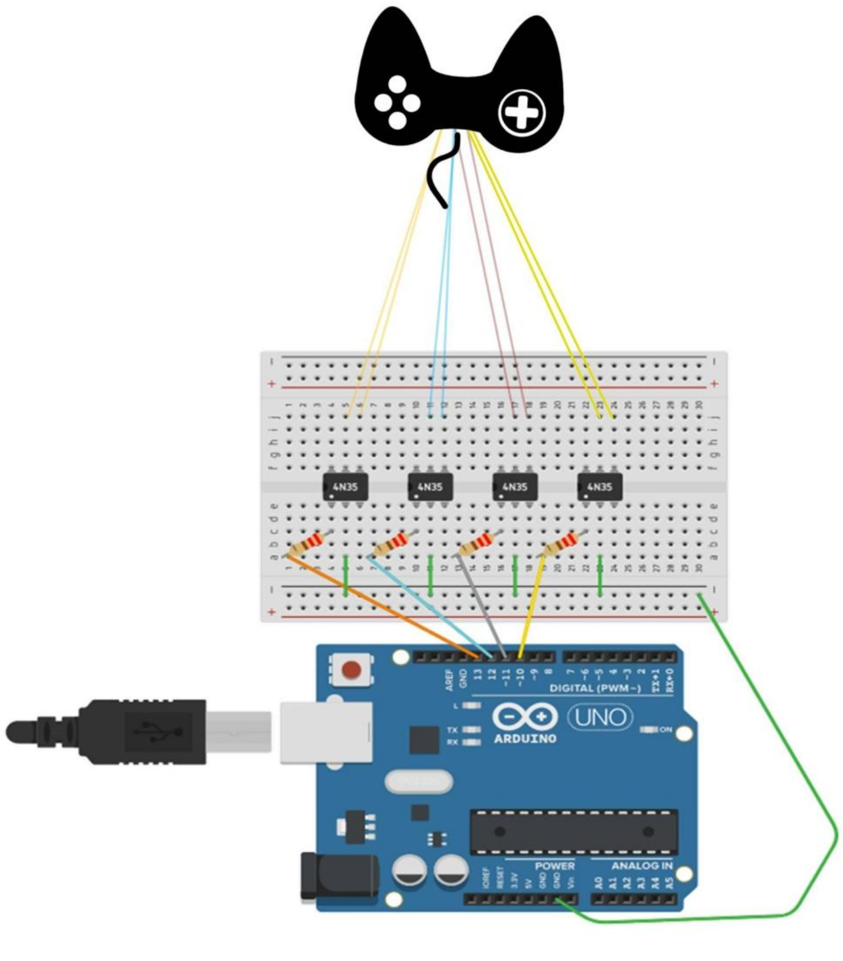


Figure (5.3) : Board connections. (Made using Tkintercad)

A python code is written to detect keypress on the computer keyboard. For this a python module 'keyboard' was used. On detection of particular set of keys, the python program would send a string over the serial connection that would indicate which of the arduino pins should go high. A forward arrow press on the keyboard would correspond to a digital write high on arduino pin 13. This would lead to switching of forward button on the RC remote due to the associated optocoupler.

The arduino code can be found at (https://github.com/prajwalsouza/Sensory-Substitution/blob/master/ProjectFiles/Connections.ino)

An Android phone was used which used the IP camera app mentioned earlier. The phone and the laptop would be connected to the same wifi network. The IP camera app would stream video at a specified address such as 192.168.1.33:8080 port. The video stream would then be read by a python program using OpenCV library. For the project, the video resolution was low as 176 x 144. The android phone was placed on the RC car with the help of a rubber band.

The Track and Recording Data

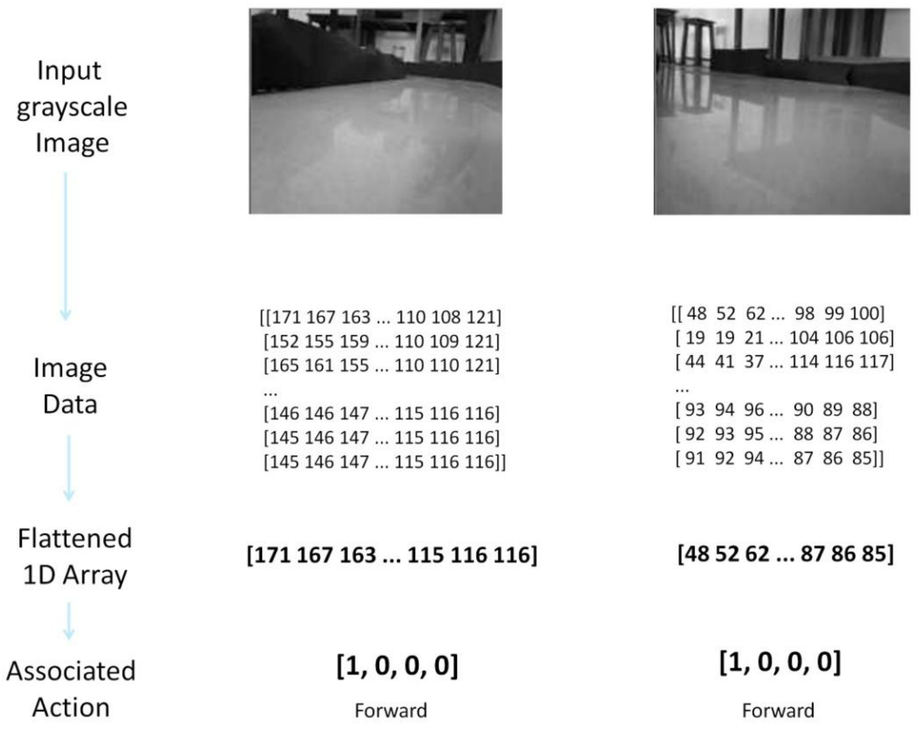
We initially created a simple track which had a white floor with black strips of chart paper which formed the foot path. But, later a variant was created where black chart paper strips were placed on the sides forming barrier which could be clearing visible.

This variant was created because it seemed to be the case that the track did not have clear distinction on what formed the road and the footpath from the view point of the phone camera on the RC car.

Once the track was built, we had to create a supervised dataset. This dataset would involve an input image and the corresponding action to be taken by the RC car. We would drive the RC car using the keyboard controls along the track. Whenever a key is pressed on the keyboard, the key press is recorded along with the corresponding video frame.

This would form our dataset.

Given an input image, output should be the actions we recorded.



```
[[171 167 163 ... 110 108 121]
 [152 155 159 ... 110 109 121]
 [165 161 155 ... 110 110 121]
 ...
 [146 146 147 ... 115 116 116]
 [145 146 147 ... 115 116 116]
 [145 146 147 ... 115 116 116]]
```

```
[[ 48  52  62 ...  98  99 100]
 [ 19  19  21 ... 104 106 106]
 [ 44  41  37 ... 114 116 117]
 ...
 [ 93  94  96 ...  90  89  88]
 [ 92  93  95 ...  88  87  86]
 [ 91  92  94 ...  87  86  85]]
```

[171 167 163 ... 115 116 116]

[48 52 62 ... 87 86 85]

[1, 0, 0, 0]

[1, 0, 0, 0]

Figure (5.4) : The data set. The grayscale image is processed and is associated with a particular action

Neural Network Architecture and Training

The neural network consisted of an input layer with 25345 neurons, a hidden layer with 65 neurons and an output layer with 4 neurons.

The input neurons correspond to the 176 x 144 image with 25345 pixels and a bias neuron. The output neurons correspond to the four possible actions the car can make, ‘forward’, ‘backward’, ‘left’ and ‘right’.

[1, 0, 0, 0] would correspond to forward action and [1, 0, 1, 0] would correspond to moving to the left.

Actions labels : [‘forward’, ‘backward’, ‘left’, ‘right’]

The Neural network was trained in Pybrain’s built-in Backpropagation Trainer.

Due to limited possible movements in the RC car used, the training was done over actions ‘Forward’, ‘Forward Right’ and ‘Backward’ actions. The data collected amounted to about 25 sessions, each session with around 75 frames, hence 75 data points.

The Datasets can be found at https://github.com/prajwalsouza/Sensory-Substitution/DataSet

All dataset were stored in python’s pickle format.

Testing the Network

The trained neural network was tested on the real track where it took actions based on the input images it received. The RC car would fail many times after training and succeed in completing the track sometimes. This was probably because of the limited training data. More hours of training and more actions should lead to better performance.

Reinforcement learning

As mentioned earlier, reinforcement learning involves an agent being trained in an environment using a reward based system. Training a real self driving car using reinforcement learning isn't sensible. We would have thousands of crashes and expecting the AI to learn after thousand road crashes (punishments) is expensive.

Although, this shouldn't stop us from trying reinforcement learning in simulated environments. We could train in a simulator and then transfer the learning to a real situation.

In this section, we will explore a reinforcement learning algorithm in a simulated environment to learn obstacle avoidance.

Q learning

There are many learning algorithms. Under Reinforcement learning, there is an algorithm is called Q Learning.

In Reinforcement learning, we have an agent, say, an ant, in an environment like a forest. The agent must take certain actions. Like move towards a food source etc. These actions may not lead to immediate rewards. Rewards could be in the distance future. After a set of actions, the agent gets a reward. This reward is associated with every action that led to it, but the association is lesser for actions that took place in the distant past. This association is recorded in a set of values called Q Values.

Now, if the agent arrives at the same state of the environment again, say, an ant arriving at the shelf which has the sugar container in the kitchen. There are many actions that the agent can take now, to go towards the sugar container, go towards the gas stove etc. Each of these actions has a Q Value. And since, taking the action towards the sugar container lead to a reward in the past, the Q value associated with the 'go towards the container' action will have a higher Q Value compared to other possible actions. The agent (here a wandering ant) will take the action that has the higher Q Value.

The calculation of the Q Value is the key to the Q Learning algorithm.

The Q Values are calculated using the Bellman Equation which gives an update rule for the Q Values.

$$\mathbf{Q\ (s_t\ ,\ a_t) = Q\ (s_t\ ,\ a_t) + \alpha\ [r + \gamma\ Q\ (s_t{}'\ ,\ a_t{}') - Q\ (s_t\ ,\ a_t)]}$$

where $Q(s_t\ ,\ a_t)$ is the Q value for the action a_t in the state s_t

r is the reward for the state s_t

α is the called the learning rate.

γ is the discount factor

$Q(s_t{}'\ ,\ a_t{}')$ is the highest Q value in the next state $s_t{}'$ associated with action $a_t{}'$

Experiment

Testing the Q learning Algorithm

We test the Q Learning algorithm using a simulation. The simulation was implemented using Javascript and HTML and hence is available online at (https://prajwalsouza.github.io/Experiments/Q%20Ball.html)

The simulation used a physics game javascript library called PhysicsJS (http://wellcaffeinated.net/PhysicsJS/)

The environment created for the agent had four walls the bounded its movement. In addition to the wall there were some other objects that were added. The agent was supposed to learn to avoid collisions with the objects and the walls.
The agent had short range sensors around it which would indicate distances from the object or the wall. Coming closer to a wall or object was given a negative reward (punishment) and if there was no object in the vicinity of the agent, it was given a zero reward.
So the agent would avoid getting negative rewards.

Each state of the environment was identified by how many sensors of the agent were lit. The firing of particular set of sensors corresponded to a particular state.

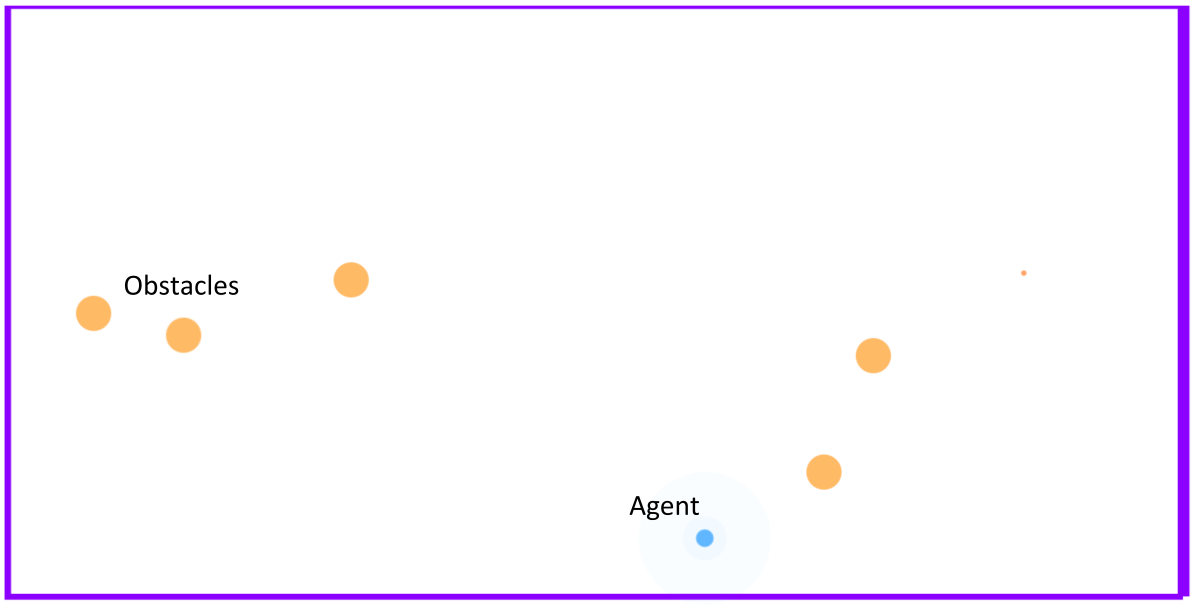


Figure (7.1) : The agent learns to avoid the walls and the obstacles put in the environment.

The agent learns to avoid the wall within a minute of training. It faced difficulties in learning to avoid other obstacles, but, it learnt to do so over time.

A similar simulation was done where an agent would train to explore a gird world. The agent had to avoid entering a red pit and go towards the green goal pit. After 100 episodes of training, the agent finds its way to the green goal. There could be multiple goals and pits in the environment.

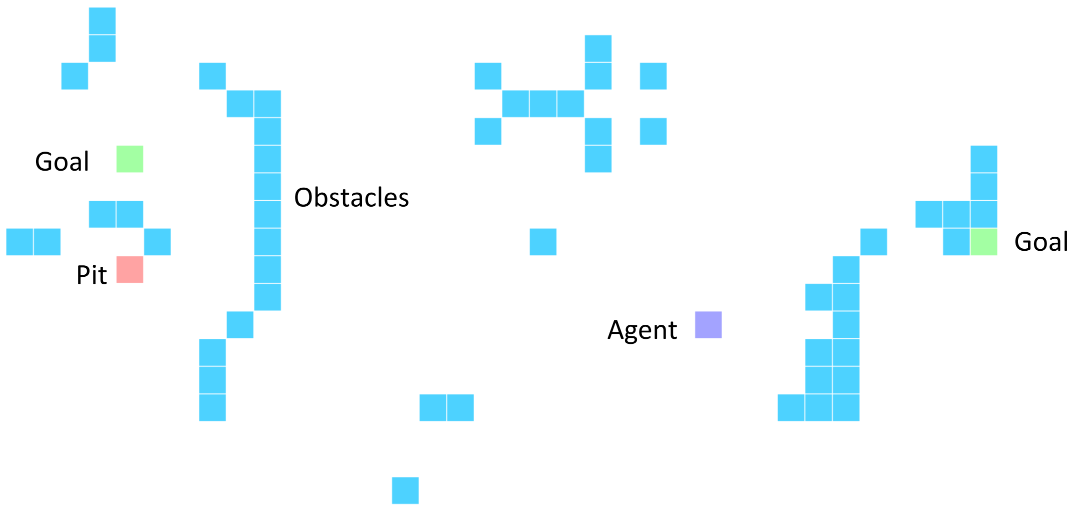


Figure (7.2) : The agent learns to avoid the pit and tries to reach the goal.

The simulation is available at
(https://prajwalsouza.github.io/Experiments/Q%20learning%20Grid%20World.html)

Experiment

Listening to the hidden layers in the neural network

Once I had a neural network that was trained to some extent, my interest was in the activations of the hidden layer. Take the hidden layer data and convert it into sound.

This was achieved using many python libraries. The hidden layer data here was a 65 dimensional vector. Human hearing is capable of distinguishing sounds well if they have a temporal resolution greater than 100ms. This would limit us to put only 10 sound variations in a second.

To convert 65-dimensional vector into a 10-dimensional vector, we will pick the 10 neurons that are activated most of the time by the network.

To achieve this, A machine learning algorithm named 'K-Clustering' was performed.

During the training of the network, all the 65 neuron activations would be recorded. The frequency of activation of a particular neuron would be recorded. Based on this frequency K centroids (in our case 10 centroids) would be assigned. This would let us pick the important neuron activations of the hidden layer.

Once K centroids were found, a forward pass would be performed. For each neuron activation a centroid would be assigned based on how near it was to a given centroid.

The centroid with the highest number of neurons assigned will have the highest value and accordingly other centroid will be assigned lower values.

We now have 10-dimensional vector which can be converted to sound.

This was achieved via frequency modulation.

The sound is then played to the human ears. Over time we hope, the human ears will be able to identify the patterns in the 10-dimensional vector.

For frequency modulation, the base frequency was taken to be 300Hz and the frequency would deviate to a maximum of ±100 Hz.

A Heuristic Fuzzy Set Theoretic Approach to the Design of Self Driving Car

Why “fuzzy”?

More often than not, in myriad of instances, we have observed that truth is not binary in nature (either 0 or 1, true or false, right or wrong). Thus, the reader is, I hope, convinced that the rules of logic as laid down by Aristotle and other ancient philosophers and in not so distant past by George Boole have to be modified for systems and devices that cannot be precisely described by mathematical models, or those that have significant uncertainties or contradictory conditions. The seminal paper titled “fuzzy sets” by Lofti Zadeh of the University of California at Berkeley published in the mid-1960s laid foundations to the kind of logic that I have spoken in the beginning of this section. Zadeh also formulated the notion of fuzzy control that allows a small set of 'intuitive rules' to be used in order to control the operation of electronic devices. In the 1980s fuzzy control became a huge industry in Japan and other countries where it was integrated into home appliances such as vacuum cleaners, microwave ovens and video cameras. Such appliances could adapt automatically to different conditions; for instance, a vacuum cleaner would apply more suction to an especially dirty area. One of the benefits of fuzzy control is that it can be easily implemented on a standard computer. Also, we have seen in the prequel that we need an enormous number of parameters to perform a basic function[a]. Could we by some alternate methods reduce such a load on the devices and yet get the desired results?

A Brief introduction to Fuzzy Sets

Let X be a set of points or objects and x be any element of X. Then we may write, in notations familiar to mathematicians, X={x}.

A fuzzy set A in X is characterized by a membership function $\mu_A(x)$ which associates with each point in X a real number in the interval [0,1] [3] (1.0)

The reader may assume the fuzzy set A to be the molecules of a cloud --- those that are in the middle of the cloud have a membership value nearly equal to 1 and those that are on the boundaries of the cloud have values for membership function closer to 0, that is, “belongingness to the cloud” of the molecules is greater at the centre of the cloud than at the boundaries.

As an example, one may feel 30°C to be warm while temperatures greater than 37°C to be hot (as 37°C is the temperature of the human body, which is well known), so one may associate a degree of hotness of 0.8 to a temperature of 30°C and a degree of hotness of 1 to all temperatures 37°C and above. The reader must note that the associated value of 0.8 is completely arbitrary and is decided depending on the nature and requirements of the systems or devices at hand and might even change if the scenario considered is totally different from the present one. *

Conventional logic theory, sometimes called 'crisp logic,' uses three fundamental logic-operations -- AND, OR and NOT--and returns either a 0 or 1. Similarly, traditional set theory, or 'crisp set theory,' assigns to objects either a membership or a non-membership in a set.

That is, either an element x of the universe **U** belongs to a set **A** or it does not, where, $\mathbf{A} \subseteq \mathrm{U}$. In fuzzy logic, the three operations AND, OR and NOT return a degree of membership that is a number between 0 and 1.

---

[x] One may consider the region of space located at a certain distance from the sun, such that all the points in this region are at a temperature of 30°C, measured using appropriate methods. Then one cannot associate a membership value of 0.8 to this region of space, the membership function now being a function of distance from the sun.

Some basic operations on Fuzzy Sets [4]

Some definitions:

Having defined fuzzy sets, we may say that a fuzzy set **A** is *empty* if and only if its membership function is 0, for all the elements of the universe **X.** (1.1)

Two fuzzy sets **A** and **B** are equal, that is **A = B**, if and only if their membership functions are equal for every element in the universe **X**
(written here forth, $\mu_A = \mu_B$). (1.2)

The complement of a fuzzy set **A** is denoted by A' and is defined
by

$$\mu_A' = 1 - \mu_A. \qquad (1.3)$$

A is contained in **B** if and only if $\mu_A \leq \mu_B$ (1.4)

The union of two fuzzy sets **A** and **B** with respective membership functions μ_A and μ_B is a fuzzy set **C**, written as **C = A U B**, whose membership function is related to those of **A** and **B** by

$$\mu_C = \mathrm{Max}\,[\,\mu_A, \mu_B\,] \qquad (1.5)$$

or, in abbreviated form,

$$\mu_C = \mu_A \vee \mu_B$$

The intersection of two fuzzy sets **A** and **B** with respective membership functions μ_A and μ_B is a fuzzy set **C**, written **C = A ∩ B,** whose membership function is related to those of **A** and **B** by

$$\mu_C = \mathrm{Min}\,[\,\mu_A, \mu_B\,] \qquad (1.6)$$

or, in abbreviated form,

$$\mu_C = \mu_A \wedge \mu_B$$

The reader has therefore come to the realization that the problems in artificial intelligence may find solutions or at least an alternate perception in the theory of fuzzy sets.

Fuzzification

Many a time, it so happens that we do not require precise data to perform a variety of tasks, like, if asked to fetch a “red” bag from the other room the doer of the task needn’t bother if the bag is fully “red”, a “reddish” one would do. (The term “reddish” immediately tells us that there is an associated degree of “redness” to this bag).

One may illustrate the concept of fuzzification through the following example.

Suppose that you are required to parallel-park your car, in which case, you must first align your car next to the one in front of your parking spot, following which, you need to angle your car back into the parking spot while slowly adjusting the steering wheel as you get to the end of the spot. You may need to align the car properly in your spot which may require to move back and forth while slowly adjusting for shallower and shallower angles using the steering wheel. One immediately realizes that the solution to this problem, if found using conventional computational methods, requires one to sort, store and process an enormous amount of data which might be expensive, both computationally and economically.

Self Driving Cars – A fuzzy set theoretic approach

In any self-driving vehicle, the primary feature must be that of obstacle avoidance, all else classifies as either a secondary feature or a luxury. Therefore, in this paper, I'd be restricting myself only to the problem of obstacle avoidance.

As we have discussed before, the input to the autonomous vehicle system is through the medium of a video feed. The question that immediately arises is this --- How may one process the data, sort it, and use it to produce a reasonable output y [m]?

The key to the puzzle is to look at the video data as a sequence of images run one after the other. A possible technique is to divide the image into 3 columns of equal size, where each column is further divided into 'n' sub columns of equal sizes and then compute the variation of average intensity of each sub-column with respect to the distance of the sub-column from the extreme left end of every column.

Written mathematically,

$$\mathbf{I_{avg} = I_{avg}(x_i)}$$

Where, $\mathbf{I_{avg}}$ is the variable representing the average intensity of the **i**th sub column at a distance $\mathbf{x_i}$ from the left-most end of any of the three columns.

The advantage of using fuzzy systems is that we need only an approximate variation of intensities, thus, eliminating the need for expensive, precise measuring devices. In fact, the task could be accomplished with extremely primitive sensory devices.

The membership functions, denoted, $\boldsymbol{\mu_{ij}}$, represents the average intensity of the **j**th sub column belonging to the **i**th column, where, **i , j** = 1,2,3 is written mathematically as follows:

$$\boldsymbol{\mu_{ij} = I_{ij}}$$

where**,** $\mathbf{i_{ij}}$ is the average intensity of the **j**th sub-column belonging to the **i**th column.

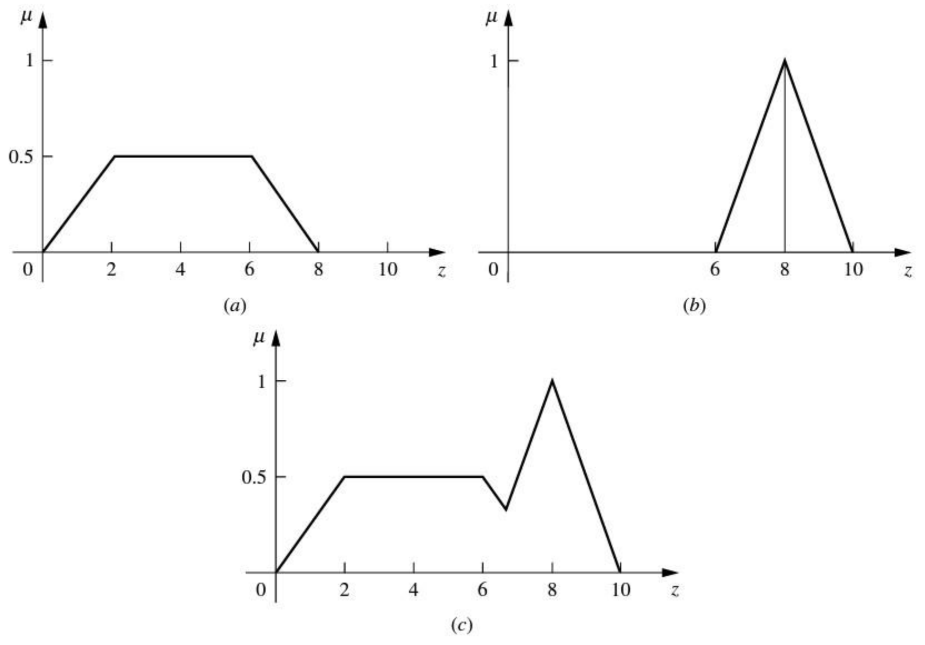
[m] y = [1,0,0,0] ; move forward
y = [0,1,0,0] ; move backward
y = [0,0,1,0] ; turn left
y = [0,0,0,1] ; turn right
y = [1,0,1,0] ; move left
y = [0,1,1,0] ; move backward left
y = [0,1,0,1] ; move backward right

This, when plotted against the index **j**, we obtain the membership function μ_i for the **i**th column.

We then take the union of the membership functions μ_i, , written mathematically,

$$\mu = \mu_1 \vee \mu_2 \vee \mu_3 \qquad \text{(Refer def. 1.5)}$$

since, the union operation involves the max-operator, **μ** thus obtained is graphically the outer envelope of the membership functions μ_1, μ_2, and μ_3.[6a] This has been illustrated in the images below.



[6b]

Typical fuzzy process output: (a) first part of fuzzy output; (b) second part of fuzzy output; (c) the union of outputs in (a) and (b)

Defuzzification

Defuzzification, as the term suggests, is the conversion of a fuzzy quantity to a precise quantity, just as fuzzification is the conversion of a precise quantity to a fuzzy one. [7]
So, how may one obtain a crisp value that can be understood by the machine from the fuzzy information as portrayed in the graph above?

Of all the methods available, we shall describe the method of centroids as it is the most appropriate one for the problem at hand, it is given by the expression:

$$x^* = \frac{\int \mu_i x_i dx_i}{\int \mu_i dx_i}$$ [8]

where x^* is the distance from the extreme left of the image where the "centroid" is located.

The output y (refer the footnote m) is then assigned as follows:

1. If the centroid is located in the middle column, then move forward.
   i.e., y = [1,0,0,0]
2. If the centroid is located in the left column, then move left and then forward.
   i.e., y = [1,0,1,0]
3. If the centroid is located in the right column, then move right and then forward.
   i.e., y = [1,0,0,1]

A neural network structure may be proposed as a means of performing above fuzzy logic inference [9], thus implementing the above theoretical discussion in practice.

Concluding remarks

Q learning algorithm for obstacle avoidance was written and in the simulation of the same the “agent” successfully learnt to avoid the “walls”. A self-driving car was also built as a demonstration of the learning process (machine learning) and an attempt of sonification of the hidden layer data of the neural network was made. The car used in the demonstration had exhibited learning but was limited by maneuverability and lack of training time, thus, failed to show adaptability in new environments. In the final chapter, an introduction to fuzzy logic was made and a new insight in the field of self-driving cars was provided.

Bibliography

a) Russell & Norvig (2003) (who prefer the term "rational agent") and write "The whole-agent view is now widely accepted in the field" (Russell & Norvig 2003, p. 55).

b) Russell, Stuart J.; Norvig, Peter (2009). Artificial Intelligence: A Modern Approach (3rd ed.). Upper Saddle River, New Jersey: Prentice Hall. ISBN 0-13-604259-7..

c) “fundamentally not a logic engine but an analogy engine, a learning engine, a guessing engine, an aesthetics-driven engine and a self-correcting engine” – Douglas R. Hofstadter, p.no., xviii, “Goedel’s Proof”, 8 th edition

1) Image credit: (https://commons.wikimedia.org/wiki/File:Dog_(Canis_lupus_familiaris)_(1).jpg)
2) Claude Shannon, “Programming a Computer for playing Chess”, 1950 (http://dx.doi.org/10.1080/14786445008521796) (accessed: 28/04/2018)
3) Yoshua Bengio, Ian J. Goodfellow and Aaron Courville, “Deep Learning”, March 30, 2015.
4) inspired from Google DeepMind’s Atari Breakout game play which used a reinforcement learning approach. (https://www.youtube.com/watch?v=V1eYniJ0Rnk&vl=en)
5) Wikipedia website, https://en.wikipedia.org/wiki/Sensory_substitution, accessed(28/04/2018)
6) Bach-y-Rita P, Collins CC, Saunders F, White B, Scadden L (1969). "Vision substitution by tactile the image projection". Nature. 221: 963–964.
7) Paul Bach-y-Rita and Stephen W. Kercel, “Sensory substitution and the human-machine interface”, TRENDS in cognitive sciences, Elsevier, vol. 7 No.12, December 2003, pg no. 541-546
8) Paul Bach-y-Rita and Stephen W. Kercel, “Sensory substitution and the human-machine interface”, TRENDS in cognitive sciences, Elsevier, vol. 7 No.12, December 2003, pg no. 541-546
9) A.Jeffs and K.Warwick, “Methods of Sensory Augmentation through Electro-Tactile Stimulation of the Tongue”, Proceedings of the 2011 10th IEEE International Conference On Cybernetic Intelligent Systems, September 1-2, London, UK, pg no. 118-122
10) Van Boven, R. W.; Hamilton, R. H.; Kauffman, T.; Keenan, J. P.; Pascual-Leone, A. (2000-06-27). "Tactile spatial resolution in blind braille readers". Neurology. 54 (12): 2230–2236.
11) Goldreich, Daniel; Kanics, Ingrid M. (2003-04-15). "Tactile acuity is enhanced in blindness". The Journal of Neuroscience Journal of Neuroscience 15 April 2003, 23 (8) 3439-3445; DOI: https://doi.org/10.1523/JNEUROSCI.23-08-03439.2003

12) Goldreich, Daniel; Kanics, Ingrid M. (November 2006). "Performance of blind and sighted humans on a tactile grating detection task". Perception & Psychophysics, pg. no. 1363-1371