International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-4, November 2019

A Solution to Cartpole using Neural Networks and Tensorflow

Nishad Sandilya, P Vinoth Kumar

Abstract: ***Machine learning is not quite a new topic for discussion these days. A lot of enthusiasts excel in this field. The problem just lies with the beginners who lack just the right amount of intuition in to step ahead in this field. This paper is all about finding a simple enough solution to this issue through an example problem Cart-Pole an Open AI Gym's classic Machine Learning algorithm benchmarking tool. The contents here will provide a perception to Machine Learning and will help beginners get familiar with the field quite a lot. Machine Learning techniques like Regression which further includes Linear and Logistic Regression, forming the basics of Neural Networks using familiar terms from Logistic regression would be mentioned here. Along with using TensorFlow, a Google's project initiative which is widely used today for computational efficiency would be all of the techniques used here to solve the trivial game Cart-Pole.***

Keywords: Artificial Intelligence, Cart-Pole using simple code, Deployment with TensorFlow, Machine Learning, Neural Networks, Python, TensorFlow

I. INTRODUCTION

Tom M. Mitchell coined the widely recognized formal definition of Machine Learning, "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.". Breaking this down further with an example, if in a situation an algorithm was to learn from the user's behavior when he flags a mail as spam in his inbox, the Experience 'E' here would be to observe the user's actions and the task 'T' will be to automatically mark the mails as spam based on the experiences. Here, the performance 'P' will be all the results of hits or misses of the actions or tasks 'T' embarked by the ML algorithm.

The goal of this paper will be to provide a clear intuition of all the components required to master a simple yet an effective benchmarking environment Cart-Pole. Further, to develop a solution to this problem using neural networks.

The entire paper is organized into sections marked by roman numbers indicating the key points and elements, followed by alphabetically marked sub markers explaining the sub topics.

A. The Cartpole System

Cart-Pole is a simple environment from the library Open AI Gym in python. The components in this game are just quite a few as well. To begin with, there is the main scenario having a horizontal floor with two rectangular boxes at the ends of the floor indicating the end points of the frame.

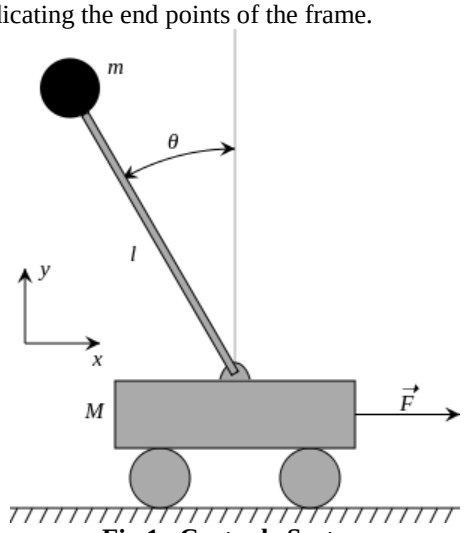


Fig 1. Cartpole System

A cart stands right in the middle which has a rectangular body with wheels on it, simple! basic 2D-Environment. There is a vertical rod right at the centre of the cart. The Primary goal of this game is to balance the rod in such a position that it does not incline more than 15 degrees from the mean point and that the cart even does not move to the extremities of the frame. An Ideal position for the cart to be in around the offsets from the centre of the frame. This would be the goal here with the algorithms.

B. Dynamics

The results depend on the dynamics of the system as well as the forces acting on it. The goal of the algorithm will be to balance the pole on the cart in such a way that the value of force set by the algorithm is adequate to counter the previous forces acting on the system which may de-stabilize the system. If this happens, the pole will move further away from the pivot and this will create an unacceptable angle between the mean pole position and its current position. This will destabilize the system. As we know, gravity's action is null with the pivot in its mean position, here the angle is 0 degree. However, if this angle increases, there will be forces acting on the cart.

Manuscript published on November 30, 2019.
* Correspondence Author
Nishad Sandilya*, Dept. of CSE, SRM University, Chennai, India. E-mail: theesaan@gmail.com
P Vinoth Kumar, Asst. Professor (Sr.G), Dept. of CSE, SRM University, Chennai, India. E-mail: vinothkumar5251@gmail.com

This force F will displace the cart in the opposite direction of the pole movement as shown in Fig 1.

The state of the system can be defined by the angular position of the pole and the linear position of the cart. We will not go deeper into this topic as we will be solving the problem using Python's Open AI Gym;
which has all the necessary actions to be applied to the environment. Still, for a little bit of intuition, the mass of the cart M is 711grams and the mass of the pendulum, m is 209 grams. g will be the acceleration due to gravity which is 9.8m/s^2. F will be positive or negative Newtons, and l will be the length of the pole which is .326m.

II. MODELS AND COMPONENTS USED IN PROBLEM SOLVING.

As mentioned earlier, this paper will proceed by giving a clear intuition and idea about the components and the working of the entire process, this is the first step. The primary components used in the making of the program/code are machine learning models. Under Machine Learning, neural networks are used to perform the calculations to find the best actions fit to keep the system stable. Other components include python's open AI gym library and regression as well as reinforcement training. All the calculations for the neural network are done using TensorFlow. We will talk about all of that later in this paper.

A. Linear Regression

Although we are not going to use this in our model a lot but only for predicting the number of moves or actions required to achieve the desired goal yet the idea of this is very important in order to properly understand what's going on with the later models to be seen here in this paper and also for a machine learning standpoint. Regression is a machine learning algorithm that is used to predict the output of continuous values for a given set of inputs. In other words, regression uses parameters and a prediction hypothesis that predicts the appropriate output of the given input. Below Fig 2 is a fig showing the prediction of housing prices based on the input feature given which is the size of the house.

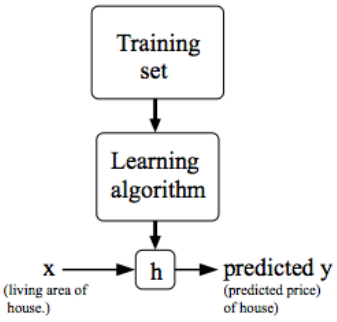


Fig 2. Basic model of regression(Houses' size Example)

Here, H is the hypothesis or the prediction model consisting of several prameters theta. Initially theta is randomized and advenced functions or algorithms such as gradient descent is used to find the best value of theta for which the hypothesis produces the best fit result for any given input. Formally the Hypothesis is given by the formula **:**

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Where, $h_\Theta(x)$ is the hypothesis and Θ_0 and Θ_1 are the parameters. The cost function is given by the mean square of the differences between the predicted values of the input and the given putput. The cost function is given by the formula:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Here, J is the cost and the goal of the gradient descent algorithm is to minimize th cost based on the tangent of the slope. Gradient Descent induces a learning rate *alpha* which impacts the value of the updated *theta's* iteratively, thus reducing down the *theta* values to the best fit required. The updated *theta* values deployed to the final model will provide us with the best prediction.

B. Logistic Regression

Logistic regression is very important to this context as the entire model of neural network which is discussed later is based on this regression model. Logistic regression is based of linear regression but the only difference is that it deal with discrete data and not continuously data. So, the outputs are discrete and are divided into classes if not binary outputs. What I mean by this is that in a given logistic regression problem, the outputs can be 0 or 1. That is the model pedicts yes or no. for example if data about temperature and humidity is given, the model can be used to predict whether there will be rain or not. i.e. 1 or 0. So, this model basically classifies an input. So, its also known as a classification model. Apart from binary predictions, there may be *k* different classes which the model might need to predict. Now the entire output is divided into *k* classes and output is predicted. Though there may be *k* classes, the prediction is based of the same 0 or 1 values. Let's see how it works**.**

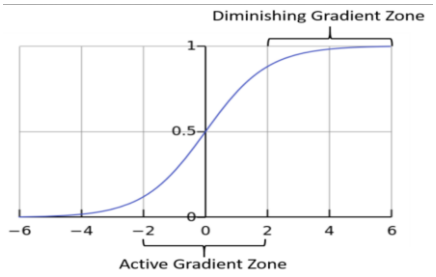
The Equations for Logistic regression uses a sigmoid function. The main goal of the sigmoid function is to contain the $h_\Theta(x)$ value or the prediction value between 0 and 1. When it does that, the final cost function's implementation becomes way too easier as the only thing needed to do then would be to just figure out whether the output is a 0 or a 1 classifier and use it to compare the prediction to the closeness of accuracy. We wont be looking into the cost function now as proper advanced optimization algorithms/modules are built into programming languages like pythoin and, Octave, R language, etc. which makes like easier. The general Prediction or so called hypothesis equation for logistic regression goes as such:

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Here, $h_\Theta(x)$ is again the same hypothesis that predicts the output based on the given *theta* parameters. Here, however, the sigmoid function is taken which contains the values of the pediction within 0 and 1 shown below.



$$\phi(z) = \frac{1}{1+e^{-z}}$$

Where Θ(z) is equal to g(z). Finally, the coust function is found out to find the deviation between the predicted output and the required output. Gradient descent algorithm or any other advanced algorithms are run on the model to find out the best fit theta values. We shall not see whats there in the gradient descent algorithm in much detail as doing so will take a lot of time and we already have quite a lot of effective advanced functions and algorithms in python libraries such as tflearn and scipy that we will be using later on to build the final model of the python bot.

C. Neural Networks

The idealogy behind neural networks is basically the human brain's functioning with the help of neurons and how they work in conjugation to learn a memory or sense an impulse. Human brains function so exclusively that even scientists till date are unware of many factors even when we have progressed so far in neuro-sciences. An artificial neuron mimics a human brain neuron. As a real neuron, an artificial neuron has input nodes, output nodes and hidden nodes or the main compututon node that mimics the dendrites, axon ends and the nucleus respectively. So, diagrammatically the neurons in an artificial nerral network can be represented as follows:

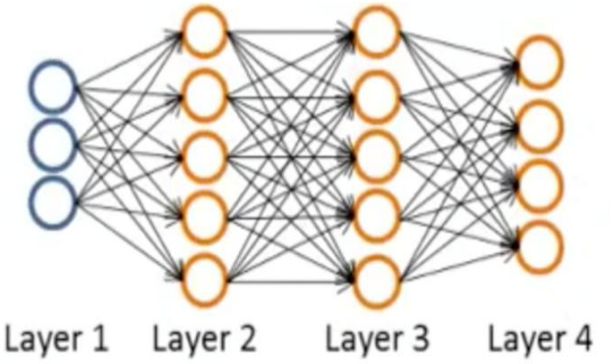


Fig 3. A neural network with two hidden layers

The above figure shows the basic representation of a neural network. As mentioned earlier, a neural net has one input layer, one or more hidden layers and the final output layer. Lets dive into its working. As seen earlier with the basic regression model, parameters *theta* is used to add to the given input features, similarly, here the parameters are called weights and are added to each and every node starting from the input node. The idea behind this is to play with the input data and see if the final hypothesis or prediction is same as the given or required output. At first iteration though, no algorithm is going to predict the correct output out of the box. This is the correct idea behind neural networks and deep learning. The model with the parameters *theta* is run against the real output data and is run continuously until the best fit *theta* values are found by the algorithm. Finally, the predicted *weights or theta* values are deployed to the model to predict the correct output afterwards. Now we will see it working step by step with this following example:

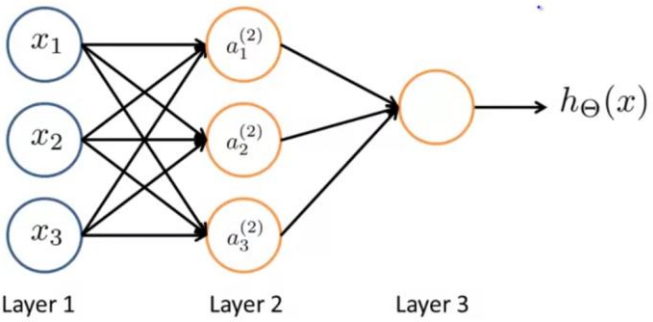


Fig 4. An example Neural Network

Here, *Layer 1* has three input nodes. These are the features of the data set. *Layer 2* has three nodes. These will be derived from the previous x values as well as the parameters *theta* associated with the first layer. Each layer will have a different set of *theta* which will be different size matrices. Finally these below operations will be performed giving us the final which is the output layer's output. At each level, with the exception of the input or layer 1, the weights or so called parameters *thetas* are added to the inputs of each node in a layer. This can be done iteratively using loops but, with vectorization, is so less computationally expensive. So, vectorization is always preferred. A slight intuition of that can be taken from the below equations where the *a*'s are the activation function of the nodes after getting multiplied by the previous layer's parameters. The a_i denotes the i^{th} node in the layer j denoted by $a^{(j)}$. Its shown below:

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

The hypothesis $h_\Theta(x)$ can be given by the formula:

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

This itself is not the end of this process, the $h_\Theta(x)$ found here is only the prediction and not the desired output. And the fact that the real output deviates a lot from this answer. So, another algorithm called *backpropagation* is used to correctly find the optimized values of the *theta* parameters as well as the total cost. Infact the total cost derivative is used for all operations. We will not go deeper into this concept as python already has enough advanced algorithms to do this for us. All we need to do is initialize random theta values as well as give the data to the model.

D. Open AI Gym, Python and TensorFlow.

These are the main components that we are going to use for the implementation of this program.

Python is chosen as it provides a lot of inbuilt libraries as well as third party libraries and they have simple to deploy but advanced and effective algorithms.
Open AI Gym is the primary library which is used to host the cartpole environment on our system. Gym is a toolkit which provides an environment for reinforcement learning models. It has a lot of inbuilt environments like cartpole which casn be used a a benchmarking tool to see if the model we provide is efficent enough to complete its objective.

So, here in the cartpole environment, Gym has predifined actions and observation records which will help us train our model without thinking much about its physics. As mentioned earlier, we are going to use neural networks to predict the correct moves and train our observation data to create an efficient model. But doing so will require a lot of compute power and time. Concretly, we are going to use Google's TensorFlow to compute on the parameters and weights of the nodes and find us an appropriate soluton. This again is computationally expensive. Owing to that, Nvidia's CUDA cores from an Nvidia GPU is used to power the computations. This will take the management load over the cpu and route it to the GPU. To give a basic intution of what TensorFlow is, Tensors are relative to arrays and operations on Tensors is called TensorFlow.

III. BUILDING THE MODEL

A. Setting up Python and required libraries

Python 3.7.3 is preferred for this along with Nvidia CUDA version 10.0. Installation requires an Nvidia GPU with a decent compute performance. After the installation of python and Nvidia drivers along with CUDA drivers, installation of *numpy*, *gym*, *tflearn*, *tensorflow~gpu* is required. TensorFlow can be also run on a CPU but a GPU is always preferred. When all is set, we can begin the coding part.

To install Python, head over to python's 3.7.3 webpage. Scroll down and download from the required versions based on the operating system, macOS/Windows and system bit configuration x32/x64. Make sure to check the add path option during installation. Well, you have just installed python.

Next to install CUDA development kits and make it work with TensorFlow, head over to Nvidia's site and download first, the graphics driver that supports CUDA. If you use a GeForce GT card, you can better use the CPU version of TensorFlow as these cards don't harness much compute power and this project's not that taxing on the CPU as well. Any GTX or RTX card/Quadros will work just like a charm. Make sure to have the correct version of CUDA Development kit installed with the correct version of Python. Well, in this case with python 3.7.3, CUDA version 10.1 should be installed. Note that with this version, numpy and othe modules might through depricated warnings, in ome cases you nedd to rool back to older versions of numpy other than that, ignore all and proceed.

B. Procedure

Our main goal of this simple model will be to atleast score a minimum of 190 points on an average with a run of 100 times. However, a score above 150 is acceptable too. First we import all the libraries into our python script as follows:

```
import gym
import random
import numpy
import tflearn
from tflearn.layers.core import input_data,
dropout, fully_connected
from tflearn.layers.estimator import
regression
from statistics import median, mean
from collections import Counter
```

Then, we create the main cartpole environment, reset it and run it just to have a clear view of what's going on. Then, some random actions and store the observations with a good average score. This will be out main training data. As we iterate further, more observations will be recorded and the average score and median will be found from the sets. I will not provide the entire code here, but using TensorFlow, this surely runs providing an effective score.

After the training data is obtained, the neural network is created using functions in python. The input layer inputs the features of the dataset including all the rows to be forward and backpropagated. There are *5* hidden layers with *128, 256, 512, 256, 128* nodes respectively. Activation function used for the hidden layers is *'relu'* and that for the final layer is *'softmax'*. Finally, the *'Adam'* optimizer is used in the regression model to find the best fit. This model is run until the loses value is decreased. Further, on deployment, the output looks something like this:

Fig 5. Final Results.

With an average score of 193, it's a pretty decent model, we save this model as *model.save('193.model')*. Further optimizing this model and training over saved models, better results are found which peaked about at 400 points.

IV. RESULTS AND DISCUSSION

The results can be summarized as follows:
1) The entire process of understanding the problem domain was done and various methods to solve the problem was studied thoroughly.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

2) The best method was found to train the model using random moves and record the observations at first. The actions that yielded the highest average scores was later used to train the neural network.
3) A score of 193 was achieved, which was quite a decent score for such a simple piece of code having only train and test data based off random initial moves.
4) At one point, a score of 400 was found, which was saved for later training. So, all high scores were dealt the same way.
5) The entire model would have been more effective if random initial lower scores were discarded and better scoring models would have been picked for training.
6) Finally, the paper had enough content to provide a basic intuition of machine learning models and techniques to ignite a tiny spark in any beginner's heart to pursue such interesting ML models later on.

V. CONCLUSION

Machine Leaarning is a required skill nowadays. AI with deep learning algorithms have revolutionized today's technology. This paper was to give a basic intution of some machine learning techniques as well as to solve a basic cartpole problem using python, neural networks and tensorflow. It can thus be concluded that the procedure of solving the problem completed its objective to score above 190 points on an average. Along with Machine Learning basics such as regression, linear and logistic, sigmoid function basics and hypothesis equations for neural networks and cost functions were given in this paper and hopefully was able to provide a basic idea how things work in Machine Learning. Also, infromation how to install and setup basic environment and libraries along with compatible versions was provided here. So, tensorflow along with python can be easily used in co-relation to create simple problem solving models like the one above.

REFERENCES

1. Savinay Nagendra, Nikhil Podila, Rashmi Ugarakhod and Koshy George PES Center for Int.Sys.Dept.of EEE, PESIT 2017 *Comparison of Reinforcement Learning Algorithms applied to the cartpole problem* 17414524 IEEE International.
2. KIM Zi Won *Comparison between different reinforcement learning algorithms on open AI Gym environment (Cart-Pole v0)* 2017.
3. Ankit Choudhary *A Hands-On Introduction to Deep Q-Learning using Open AI Gym in Python* IIT Bombay EEE [Online]. Available: *https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/*
4. Andrew NG Machine Learning Course Coursera [Online course]. Available: *https://www.coursera.org/learn/machine-learning*
5. Sentdex machine learning tutorials [Online]. Available: *https://pythonprogramming.net/openai-cartpole-neural-network-example-machine-learning-tutorial/*

AUTHORS PROFILE

Nishad Sandilya is currently pursuing B.Tech in Computer Science and Engineering in SRM Institute of Science and Technology, Chennai, India. Will graduate in 2021. He is very fascinated about technology and frequently writes tech blogs in websites. A tech geek, who loves algorithms and dives deep into the concepts, trying to find efficient ways of solving problems. A machine learning beginner who learned from scratch now is gearing up for future projects and ventures.

P Vinoth Kumar obtained his Bachelor degree in Computer Science from Vinayaka Missions University, Salem in 2005-2009 and Master degree in the department of Computer Science and Engineering from Vinayak Missions University, Salem in the year 2011. Currently, he is a faculty in the Computer Science and Engineering. Department in SRM Institute of Science and Technology. His main research interest is on Networking and Networking Security.