Ira A. Fulton Schools of Engineering
Arizona State University

CSE 571: Artificial Intelligence

Neural Network for Collision Prediction

Purpose

The purpose of this assignment is to familiarize you with neural networks and their applications. More specifically, you will learn to collect training data for a robotics task and, in turn, design a neural network that can process this data. In the assignment your goal is to help a small robot navigate a simulated environment without any collisions. To accomplish this, you will need to train a neural network using backpropagation that predicts whether the robot is going to collide with any walls or objects in the next time step. All of the theories and best-practices introduced in the class are applicable here. Your task is to make sure the little fellow can safely explore its environment!

Objectives

Students will be able to:

- Collect and manage a dataset used to train and test a neural network.
- Define and use PyTorch DataLoaders to manage a PyTorch Datasets.
- Design your own neural network architecture in PyTorch.
- Evaluate and improve a neural network model and verify an application in simulation.

Technology Requirements

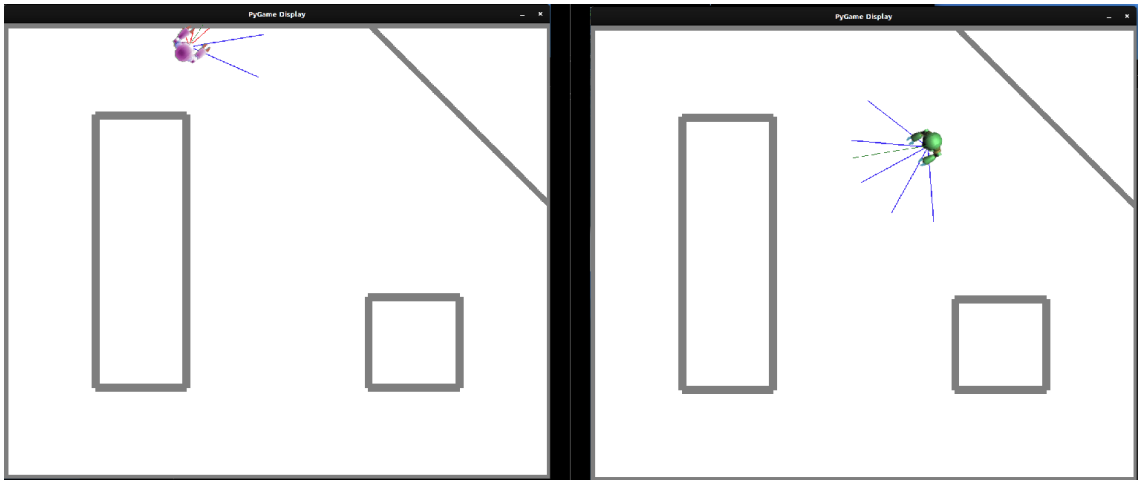
- System designed for use with Ubuntu 18.04
- Python and its related libraries. Using Anaconda is recommended.
- Python libraries: cython matplotlib sklearn scipy pymunk pygame pillow numpy noise torch

Project Description

Part 1

The first task is to collect data that can be used to train the model. Collect a single sample per action containing, in order, the 5 distance sensor readings, the action, and whether or not a collision occurred (0: no collision, 1: collision). This data should be saved as a .csv file with 7 untitled columns. For grading purposes, submit your 'submission.csv' containing 100 data samples. For training in the future parts, you will need to collect much more than this.

Files to edit:
collect_data.py



The robot should wander around with no regard for its environment or avoiding collisions. Also, see below a sample of what your submission.csv should look like.

	A	B	C	D	E	F	G
1	150	150	150	150	150	1	0
2	150	150	150	150	150	0	0
3	150	150	150	150	150	0	0
4	150	150	150	150	146.136932668436	-1	0
5	150	150	150	150	133.030654941255	2	0
6	150	150	150	150	150	0	0
7	150	150	150	150	150	1	0
8	150	150	150	150	150	5	0
9	150	150	150	150	150	5	0
10	150	150	150	150	150	2	0
11	150	150	150	150	150	0	0
12	150	150	150	150	150	1	0
13	150	150	150	150	150	5	0
14	150	150	150	150	150	3	0
15	150	150	150	150	150	1	0
16	150	150	150	150	150	1	0
17	150	150	150	95.125430301932	150	1	0
18	150	150	150	82.3197727321175	69.8521129671052	0	0
19	150	150	125.420080286128	62.671593936248	53.2714714260367	-1	1
20	150	150	150	150	150	-3	0
21	150	150	150	150	114.347309676051	-3	0
22	150	150	150	103.388726740617	66.9770547205216	0	0
23	150	150	150	150	64.3701480233596	-3	0
24	150	150	150	48.8728744798886	48.3207393134496	-1	1
25	150	150	150	150	150	0	0

Part 2

Now that you have collected your training data, you can package it into an iterable PyTorch DataLoader for ease of use. You may be required to prune your collected data to balance out their distribution. If your dataset is 99% 0s and 1% 1s, a model that outputs only 0 would achieve good loss, but it would not have learned anything useful. Make sure to create both a training and testing DataLoader. Use training_data.csv collected from the previous part. Make sure to use the PyTorch classes mentioned in the comments of Data_Loaders.py.

Files to edit:
Data_Loaders.py
saved/training_data.csv

Part 3

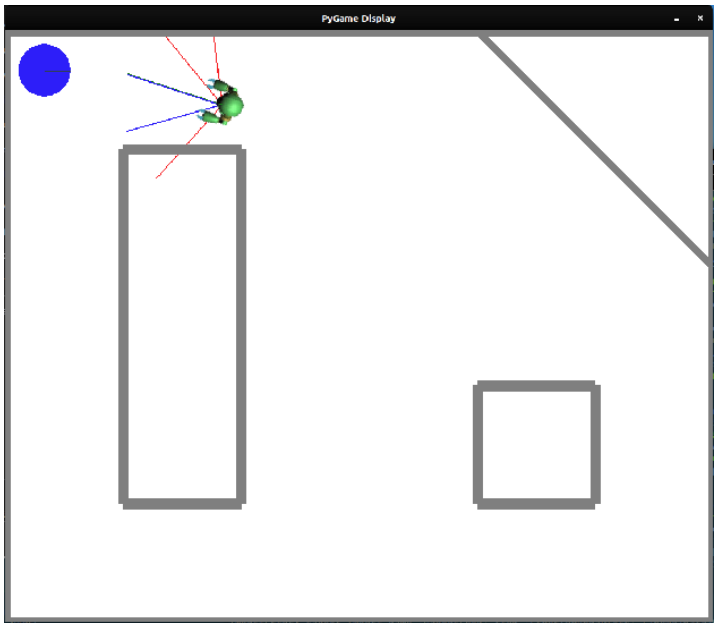
For Part 3, you will be designing your own custom neural network using PyTorch's torch.nn class. You will need to initialize a custom architecture, define a forward pass through the network, and build a method for evaluating the fit of a given model.

Files to edit:
Data_Loaders.py
Networks.py
saved/*

Part 4

In Part 4, you must train a model using your custom network architecture, which accurately predicts collisions given sensor and action information. Your grade will depend on the accuracy of the model. You may need to try many different strategies and architectures to achieve a well fit model. Keep track of your training and testing loss throughout the epochs, and generate a plot with these lines at the end. To see an application demo of the learning your robot has done, run goal_seeking.py, which will have the robot seek out goals while only taking possible actions it deems to be safe.

Files to edit:
Data_Loaders.py
Networks.py
train_model.py
saved/*



Submission Directions for Project Deliverables

Part 1

Files to submit:
submission.csv

Part 2

Files to submit:
Data_Loaders.py

Part 3

Files to submit:
submission.zip – containing:
> Data_Loaders.py
> Networks.py

Part 4

Files to submit:
submission.zip – containing:
> Networks.py
> saved/
>saved_model.pkl
>scaler.pkl

Evaluation

Part 1

Pass/Fail 1 point

Part 2

Pass/Fail 1 point

Part 3

Pass/Fail 1 point

Part 4

0-7 points. 3% of the max score will be deducted for every missed collision. 1% of the max score will be deducted for every false positive above 10.