Report on Machine Learning Lab, Ex 5

Mostafa Mohamed, Omar Kassem

February 17, 2017

1 Introduction

This is a report about the deep learning lab, exercise 5. The general task of this assignment was extending assignment 3 to support new variants of the problem. The task of assignment 3 was to use Tensorflow/Keras to implement a convolutional neural network to train it on solving a search problem and compare it to the A* search algorithm.

2 Search problem

The given problem was find a target position in a given maze. We were given a simulator that can visualize our algorithm or the given reference A* algorithm.

2.1 Variation

The variation of this problem that we attempt to solve in this assignment is solving the maze when the target position can change randomly across the map after each simulation.

3 Architecture

- Input: There was a given script that generates the train/validation data according to the A* algorithm. It starts from a variety of random positions and tries to search for the target. The A* reaches the target optimally.
- Network: We used Keras to implement our neural network. We implemented a network that consists of 2 fully-connected layers followed by 3 convolutional layers. We tried different activation functions on some layers (like tanh), mostly the relu showed the best results.

4 Approach

We had 3 different ideas to change in order to attempt improving the results; namely 'knowing the destination', 'getting landmarks' and 'mark districts in the map'. The results we previously had was an accuracy of 31% of solving given map given that the target is changing randomly.

Figure 1: Original maze

Figure 2: Simple colouring

4.1 Memorizing the target

The idea here was to basically make the agent have an idea how the target will be like while looking for it, the same idea is used in every day life, one needs to know how the destination is like before going in order to easily identify it. We implement this by storing an extra state (along with the history of memorized states) of the local view of the target, and maintaining that it will always be identified as the target view.

4.2 Coloring the walls

Originally the colors of the wall was only red, but in more realistic examples we can often identify some landmarks of different regions, like the general architecture of buildings and so on. Similarly here we start to color the walls depending on different regions. We had mainly 2 strategies for this coloring.

4.2.1 Simple coloring

This divides the map into 4 equal regions, and colors each one of them with a different color.

4.2.2 Gradient coloring

This colors the walls by a gradient color (with a small random variation) with a gradient through out the diagonal of the map.

Figure 3: Gradient colouring

4.2.3 Some problems we faced

- We changed the color of the target to avoid confusion with other colors.
- We tried to avoid strong bluish colors for walls to avoid confusion with the bot.
- The colors of the walls are varying between reddish and yellowish colors.

4.3 Gridding

The idea here was to divide the map into a set of grids and instead of training he changing target randomly across the map. We would make sure that each grid in the map is sampled equally. So that we don't end up knowing vacant areas more than tight areas. We sample inside each grid either a representative of that grid (the middle element) with probability 20% or randomly from the grid otherwise. The combination of both will ensure that we "know" enough about the grid (even if we didn't reach the target exactly, we can still get close), and the randomization part in order not to be fixed on a specific place (thus missing *on purpose* potential targets while testing).

4.3.1 Distance Accuracy

In order to test that we got "close" enough to the target we computed (in addition to the already implemented accuracy) the distance accuracy. It's computed by the formula $\frac{1}{1+\lfloor \frac{d}{g} \rfloor^2}$, where d is the distance between the bot and the target at the end of the game, and g is the side length of the grid that we are using.

5 Results

We tried different configurations for the running. The girdding factor g that we used is 3 (Normal) or 6 (Big).

5.1 Standard map

The map given in assignment 3, it's a 28×28 map, with a local view 5×5 We ran 8 experiments; for each we train 7 times and test 10 times for each of them,

Figure 4: Big Map

and these are the average results.

Target	Gridding	Color	Accuracy	Distance Accuracy
No	No	No	31%	-
Yes	No	No	70%	-
Yes	No	Simple	80%	-
Yes	No	Gradient	66%	-
Yes	Yes	No	70%	72%
Yes	Yes	Simple	79%	81%
Yes	Big	Simple	77%	82%
Yes	Big	Gradient	64%	71%

5.2 Big map

A new map of size 50×50 with a local view 9×9 We ran 4 experiments; for each we train 2 times and test 10 times for each of them, and these are the average results. The state size of this map was much bigger than the other maps which showed some restrictions while experimenting. Unfortunately we couldn't run more experiments due to lack of resources: time, quota for saving tables and RAM memory for loading large tables.

Target	Gridding	Color	Accuracy	Distance Accuracy
No	No	No	31%	-
Yes	No	Simple	57%	-
Yes	Big	Simple	62%	65%
Yes	Big	Gradient	49%	55%

5.3 Conclusion

We can conclude from the experiments that memorizing the target showed a big improvement, further, the coloring of the walls (especially simple coloring), and finally for the big maps the gridding.