Suggestion for Project 3, FYS-STK4155

Markus Pettersen

November 18, 2020

Introduction

In 2014 the Nobel prize in medicine was awarded to May-Britt & Edvard Moser alongside John O'Keefe, for their discovery of so-called grid cells, a special type of neuron that seems to be essential to how animals navigate. Grid cells get their name from the fact that when an animal navigates an environment, a single grid cell only reacts or fires at certain locations in the environment. Surprisingly, if we keep track of where such a cell fires over time, they trace out a stable hexagonal pattern in space. For a nice overview of how grid cells help us navigate and map space, see for example [1].

Perhaps even more interesting is the fact that the same structure was recently found to arise in recurrent neural networks (RNNs) trained to perform a navigation task [2]. The purpose of this project is to see if we can train an RNN to perform path integration[1], and to see whether grid cells occur in simple RNN models.

Task a): Creating a dataset

We want to emulate a real rat navigating an environment, and for simplicity, we assume that in the 2D case, the environment is a 1×1 square chamber, with no openings. In the 1D case, the chamber is a straight track of length 1.

The dataset should consist of random, smooth paths in this environment, and our simulated "rat" should change both direction and speed as it navigates. Initially, the input to the network should be the head direction and speed of our simulated rat or agent, and the targets or labels should be the agent's position in ((x, y)-coordinates). You can choose whether you want to try a 1D or 2D dataset. How do you handle boundary conditions?

Task b): Creating an RNN & preparing our dataset

First, create a simple/vanilla recurrent neural network using your favorite library (Tensorflow, Pytorch, your own?). Describe the RNN equation used in your model. Explain your choice of weight initializations.

Task c): Navigating

Train the RNN to path integrate/navigate. How should you initialize the state of the network? What kind of error do you achieve? Try out different recurrent activation functions and weight initializations to see if you can improve your results.

Task d): Looking for grid cells

Plot the hidden layer activities, i.e. the RNN states as a function of space. Can you find any repeating patterns in space? For the 2D case, the scipy function binned_statistics_2d could be useful.

[1] Predict its position using only current direction/heading and speed.

Task e): Looking closer

It might be that you did not find any grid cells in task d). To improve your results, you can try to follow the approach in [2] more closely: they add an additional densely connected layer after the RNN layer, with a linear activation. It is in the hidden state activations of this so-called readout layer that they find grid cells. Furthermore, [2] did not report finding grid cells before they added dropout to the readout layer. Inspired by this, you should try to implement some form of regularization to your network, for example weight or activity regularization. You should also consider adding dropout to the readout layer. Do your results improve?

In task f, you can choose to implement one of two options (or both!):

Task f): Going beyond Cartesian coordinates, option 1

In real rats, position is not encoded as a Cartesian coordinate, but rather in terms of the activity of so-called place cells, a special type of neuron which is only active in a single region of space. Unlike grid cells, however, place cells do not form patterns in space. Can you find a way of going from a Cartesian representation of space to a place cell representation? How well does your network navigate now? How can you decode the predicted position from the place cell activity?

Task f): Radial basis functions, option 2

We can also consider changing the inputs to our network, by replacing our speed and head angle inputs, to mimicking so-called speed and head direction *cells*. Add a so-called radial basis function input layer to your model, and compare its performance to the Cartesian case. Describe how you initialize the weights of the new input layer. Does this improve the error in the predicted position?

References

[1] *How Grid Cells Map Space*. en-US. URL: http://numenta.com/blog/2018/05/25/how-grid-cells-map-space/ (visited on 11/15/2020).

[2] Andrea Banino et al. "Vector-based navigation using grid-like representations in artificial agents". en. In: *Nature* 557.7705 (May 2018), pp. 429–433. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0102-6. URL: https://www.nature.com/articles/s41586-018-0102-6 (visited on 11/13/2020).