159.372 Intelligent Machines
Lab 6: Unsupervised Learning

Stephen Marsland

1 Introduction

The iris.py program in the Chapter 14 folder from my website demonstrates the use of both the k-means and Self-Organising Map (SOM) algorithms on the iris data that we saw when we looked at supervised neural networks. When you run the code, it shows you a plot of which network neurons were used to identify the different classes as red squares, and green and blue triangles. Any neurons that did not match any of the inputs are shown as black dots. The two plots are based on the training data and test data respectively. You should use this code (and dataset) as the basis for your answer to the first question below.

Question 1 Choosing an optimal network size for the SOM is mostly just a question of trying out different values. The question is what you should optimise the value on. You don't want any neurons that match inputs from two different classes, but you don't want too many neurons, either. Devise a scoring scheme that evaluates how well each map size does based on these two criteria, plus anything else that you think is important. Implement it in Python and use it to optimise the size of the network for the iris data.

Question 2 Suppose that a robot has 16 sonar sensors, which emit a pulse of sound (which you hear as a click) and wait for it to bounce off the nearest object and come back to the sensor. By timing how long that takes, the robot can estimate the distance to the nearest object.

It is these numbers that you can see in the files `sonar000.dat, sonar001.dat, ..., sonar021.dat`, which are in the file `sensors.zip` on Stream. The first number in each line of the file is just the index, which you should strip off before inputting the data to a SOM, but the other 16 numbers are the sensor readings taken by the robot as it travelled along some corridors. The robot took sensor readings continuously, and averaged the readings every 10 cm, storing that number. The aim of the averaging was to remove any errors in the readings.

The picture that the robot gets from its sensor readings is the distance to the nearest object in each direction, and this is the only information that the robot has about its environment. From that data it has to avoid walls, navigate, and try to recognise where it is. More modern robots have laser range-finders and cameras, but dealing with those data sources is more computationally expensive, so we will stick to sonar data here.

Based on this data:

- load some of the sonar data and have a look at it by hand
- decide on any pre-processing (normalisation) of the data and strip off the first entry of each row
- use k-means with different numbers of clusters to work out how many clusters there are (you'll have to invent an error criterion)
- there can be problems with local minima, so run the algorithm several times for each number of clusters
- use the SOM to decide how many different places there are in the sensor data. You'll have to choose a suitable size and amount of training for the SOM by experimentation. Then keep track of the winning nodes of the network.
- see how many of the data runs are from the same environment as each other. There are 4 different training environments in the 22 datasets. Train the network on 1, and run it on the others to see if the nodes that fire are the same ones. This will give you a hint as to whether or not they are different environments.

