CS3105 Artificial Intelligence

T5 Machine Learning

Practical 2

Aim

This practical aims to construct two neural systems, one capable of playing the 20 Questions game, and the other capable of programming itself. Both systems are to be constructed using Java and the Encog v3.3 Package (http://www.heatonresearch.com/encog).

Objectives

After doing the practical you should be able to

- Design, implement, and document a neural system for playing the 20 Questions game.
- Design, implement, and document a neural system capable of programming itself.
- Compare the similarities and differences of the 20Q system with standard neural pattern classification, and the self-programming system with conventional programming.

Part 1 20Q Specification

Overview

The 20 Questions game consists of a questioner asking up to 20 questions of an agent in order to make an informed guess as to what concept the agent is thinking of. A neural system is to be constructed to play the questioner. The system is to be based on a feedforward network using hyperplanar decision boundaries, sigmoidal activation, and non-stochastic output units.

Detail

A neural system is to be constructed to play the 20 Questions game. Construct the system using Java and the Encog v.3.3 package (http://www.heatonresearch.com/encog). This architecture is to have a fixed number of input units to cater for coded bit patterns representing the set of answers. A single layer of hidden units is to provide hyperplanes that partition the input space into various regions for concept identification. A fixed number of output units is to provide bit patterns for an index to particular locations in concept space. With sufficient answers, a correct concept should be isolated and output by the system.

Pre-Processing, Input, Output, and Post-Processing

Decide on a small but significant number of concepts you wish an initially fixed system to be able to handle. Also decide the questions to ask that will distinguish one concept from another according to their attributes. Each question is to have a yes/no answer. Construct enough input units to be able to represent the set of all possible questions.

Each Input-Output (I/O) pairing/pattern corresponds to a question set-answer pairing with a concept. Use and report a concept-pairings table that contains all the concepts and their question set-answer pairings. Construct enough output units to cater for the set of all possible concepts. Create and report a concept-output table whose entries are the encountered concepts and their associated neural output patterns.

Hidden layering

Construct a single layer of hidden units fixed in number and able to partition the input space into regions with a single binary class of output in each. Once the above initial fixed system works, modify your system to be extendable on the fly in interaction with humans. That is, add new concepts by having humans play 20Q with your system. In particular, after a sequence of questions and answers has occurred and the system has guessed, ask the human player for the concept.

To incorporate the new concepts into the neural network, add and train hidden units one at a time until separation of the I/O is complete. That is, while classification is incorrect, add one hidden unit at a time and retrain the whole network each time. Do this for at least 5 new concepts. By carrying out appropriate pre-processing and training, update your tables to store the information for each new concept given by the human player. Show select updating in your report.

How convinced are you that your system could scale up if the number of hidden units could be set to be an efficient number? Give your reasons. How fault tolerant is your system?

Choosing a question and when to guess
Choose questions that eliminates the most alternatives at each stage and have the system guess as soon as it is reasonably likely to guess correctly a familiar or close concept. You may have to use your tables to do this. Describe your mechanisms and explain why they are good ones.

Extensions
Evaluate how well your system keeps the number of hidden units under control. Improve your system to keep the number of hidden units under better control.

Question
What are the similarities and differences between the way your system generalises and standard neural generalisation?

Part 2 Self-Programming Specification

Overview
The self-programming involved consists of a system learning to be a finite state machine (FSM) for performing a given task. The system is to be based on a recurrent feedforward neural network of the Elman type, again using hyperplanar decision boundaries, sigmoidal activation, and non-stochastic output units. The given task is to be a virtual vending machine. The machine sells apples for 20p, bananas for 30p, and chocolates for 50p. The machine can accept only 10p, 20p, and 50p coins. Only one item is sold at a time. The machine rejects coins inserted once the balance in the machine is in excess of the most expensive item. Change is given by the machine.

Detail
The feedforward topology should consist of an input layer comprising a fixed number of units for representing the inputs to the vending machine, an output layer consisting of a fixed number of units for representing the vending machine output, and a single hidden layer with a variable number of units for representing the internal states of the vending machine.

You should start by attempting to have the neural network learn a state transition table with just 2 states before attempting larger tables, growing the hidden layer 1 unit at a time when necessary. Report on how many of the vending machine's state transitions your system is able to learn.

Extension
Use weight decay to see if you can reduce the number of significantly non-zero recurrent weights and hence the number of significantly different internal states.

Questions
How does the system's final internal state structure compare with a minimal manually created structure?
How fault tolerant is your system?
What further design features would be required for your self-programming system to
(a) learn rules such as addition using the carry rule
(b) replace conventional manual programming?

Report requirements

- For handing in, you should supply two programs, one for Part 1 20Q and one for Part 2 Self-Programming. Submit both source code and jar files for your programs together with sufficient I/O behaviour from them to demonstrate that they work correctly. Make sure you include a description of the net construction, the training, and any generalisation tests. Give select examples that show your 20Q system guessing well or strangely. Give select examples that show your self-programming system vending as required or strangely. Describe how fault tolerant your systems are. Make sure you also include answers to the questions.
- You need to state if your programs work fully or if there are non-working aspects. If the latter is the case, you should provide clear details.
- Documentation should as usual be such as enables straightforward understanding, compiling and running of the programs on a specified type of Junior Honours lab machine. A word limit of 4800 words is not to be exceeded. Your source code should be well structured and well commented. Be clear as to what is your own coding and what is others' code.

  Instructions for running must be for a jar file you have tested as running successfully on the specified lab machine. Name the jar file for Part 1 as "20QLearning.jar" and ensure it runs using "java –jar 20QLearning.jar". Name the jar file for Part 2 as "FSMLearning.jar" and ensure it runs using "java –jar FSMLearning.jar". Instructions for compiling and running must be confirmed through clear screen snapshots of them working. **If your instructions fail, your program will be treated as not compiling or running.**
- The start date for this practical is Friday week 7, i.e. 13th March. An electronic copy of your program and report should be placed in MMS by Thursday week 10, i.e. 16th April. The documents must be zipped into a single archive for MMS.

Check Table:
Use the accompanying check table to check you have addressed all the major aspects of the specification and then read through the specification again for finer details.

Feedback and Marks:
See the School Handbook for the section on feedback and marks.

Remember: Late work will be penalised by MMS.

MKW
13/3/15

Check Table:

Part 1 20Q		Done?
(i) Pre-Processing and Input	Clearly described and justified	
(ii) Hidden layering	Clearly described and justified for fixed system Clearly described and justified mechanism for adding units	
(iii) Output and Post-Processing	Clearly described and justified	
(iv) Training	Clearly described and justified	
(iii) Choosing a question	Clearly described and justified mechanism for effective choices	
(iv) When to guess	Clearly described and justified mechanism for effective guessing	
(vi) Answers to questions	Clearly described and justified	
(vii) I/O	Clear, easy to understand tables for question set-answer pairings and concept-output with main text explanation Selected testing and key examples of question and answer session	
(viii) Extension	The number of hidden units under evaluation and control	
(viii) Documentation for Part 1	Documentation is clear, appears comprehensive, and shows understanding and effort upon read-through	
Part 2 Self-Programming		
(i) Pre-Processing and Input	Clearly described and justified	
(ii) Hidden layering	Clearly described and justified including recurrent connections Clearly described and justified mechanism for adding units	
(iii) Output and Post-Processing	Clearly described and justified	
(iv) Training	Clearly described and justified	
(iii) Internal states	Clearly described and justified comparison with minimal manual state structure	
(vi) Answers to questions	Clearly described and justified	
(vii) I/O	Clear, easy to understand state transition diagrams and tables for neural and manual FSMs Selected testing and key examples of vending sessions	
(viii) Extension	Weight decay with the number of hidden units under evaluation and control	
(ix) Documentation for Part 2	Documentation is clear, appears comprehensive, and shows understanding and effort upon read-through	