Canterbury Christ Church University

Computing, Digital Forensics and Cyber Security

BSc Computer Science 2018-2019

Computational Algorithms

Assignment 2

Date: 10th December 2018

A.I. Algorithms

General guidelines for submission

- This is an **individual assignment** and must represent your own work.
- The required date of submission is **7th January 2019 14:00** via Blackboard as a zip folder called **MCOMD2CAL.**
- The report must adhere to the department guidelines.
- This assignment has been set by **Richard Henson**
- This assignment has been moderated by **Vijay Sahota**
- This assignment is worth 50% of the module.

General advice

- You are required to back up your work regularly onto your N: drive, and onto removable storage devices of your choice. Always check the date-stamp on your files before submission. You will need to submit the latest version of your software.
- You submission must be compatible with software versions we currently have on the University's network.

Tested Learning Outcomes

1. Demonstrate an understanding of computational complexity;
2. Demonstrate knowledge of existing algorithms;
3. Critically evaluate existing algorithms;
4. Implement and develop an effective algorithm to current issues in computing.

Introduction

The aim of this assignment is to explore algorithmic complexity in the context of Artificial Intelligence (A.I.). There are three tasks, each marked independently of the others. Task 1 is a research essay requiring a scientific understanding of a complex compiled algorithm. Tasks 2 and 3 are practical coding exercises. For the practical assessment, skeleton coding for two algorithms used within the A.I. are provided and students are required to perform a detailed analysis of each algorithm before enhancing and recoding each in order to satisfy the assignment specification. A moderate amount of independent research is necessary to understand each algorithm.

Task 1

Top-down processing typifies one approach to A.I. In order to understand this approach better, write a 1200 word technical account of an algorithm and evaluate its efficiency. Each student will be allocated a specific algorithm to research from the following list:

1. The Rete Pattern Matching Algorithm
2. The A* Algorithm
3. Genetic Algorithms
4. Best First Search
5. Depth First Search
6. Simulated Annealing
7. Levenshtein Distance Algorithm
8. WalkSAT algorithm
9. Minmax algorithm
10. Wangs algorithm
11. Manhattan Distance Algorithm

Ensure that you cover the implementation details of the algorithm and give examples of its use within the field of A.I. Your account should also include the operational details. Use annotated diagram and/or coded examples where appropriate. Use Harvard referencing to reduce the word count.

NB: Any diagrams or coded examples are not included in the word count.

Task 2

Appendix A illustrates a toy example of a very basic expert system written in JavaScript. The program selects the correct clothing based on weather conditions. The example has limitations, which needs to be identified and corrected. Re-implement the improved system using either C# or Python and include the following enhancements:

- A separate knowledge base (set of rules) encoded within the program;
- A simple parser or data structure to match rule patterns. These rule patterns are broken down into antecedent and consequent patterns. You may use a regular expression matcher to simplify the process or replace it with a suitable data structure and assume the rules have already been parsed;
- A query generator to accept user input where antecedent facts are unknown.

Finally, write a brief account of how Bayesian logic might could be introduced into a production rule system to more closely reflect the human decision making process.

Task 3

Bottom-up processing has emerged as a transformative technological with a sound mathematical basis. Artificial Neutral Networks (ANN) encapsulates this mode of thinking. In order to understand this approach better study the Python code presented in Appendix B.

Implement two alternative Activation functions to the one outline within the function `nonlin`. Assuming this activation function is a performance base line, compare the algorithm's efficiency with two alternatives sourced and re-written by you. Provide evidence of any issues that arise when executing the coded example and possible limitations when used with different ANN structures.

Deliverables

- Professionally written and compiled report for each task covering the key aspects of the design and implementation of the algorithms as well a written commentary on all research elements within the assignment brief.

- Documentation on any worked solutions or proofs outlined in the assignment brief were appropriate.

- Appendices containing references to any code sourced from elsewhere and used within the implementation and any high level utilities or libraries available in the selected programming langauge;

- Source and possible compiled files of the implementation in a suitable file structure for testing within the standard programming environment.

Appendix A

```
########################################
# Simple PR example written in JavaScript
########################################

<script>
  var numberOfFacts = 0;
  var FACTS = new Object();

  function addFACTS () {

      FACTS.sunIsShining = false;
      FACTS.weatherIsHot = false;
      FACTS.weatherIsCold = false;
      FACTS.wearSunGlasses = false;
      FACTS.summerTime = false;
      FACTS.winterTime = false;
      FACTS.darkClouds  = false;
      FACTS.itsRaining = false;
      FACTS.openUmbrella = false;
      FACTS.wearCoat = false;
 }

  function countFACTS () {
      count = 0;
      for (fact in FACTS) {
         if (FACTS[fact]) count++;
      }
      return count;
  }

  function printFACTS () {
      for (fact in FACTS) {
         document.write("<p>" + fact + " is " + FACTS[fact] + "<p>");
      }
  }

  function infer () {
    do {
       numberOfFacts = countFACTS();
       FACTS.winterTime ? FACTS.wearCoat = true : FACTS.wearCoat = false;
       FACTS.weatherIsCold ? FACTS.winterTime = true: FACTS.summerTime = true;
       FACTS.itsRaining ? FACTS.darkClouds = true : FACTS.sunIsShining = false;
       (FACTS.sunIsShining && FACTS.summerTime) ?  FACTS.wearSunGlasses = true : FACTS.wearSunGlasses = false;
       FACTS.itsRaining ? FACTS.openUmbrella = true : FACTS.openUmbrella = false;
       (FACTS.sunIsShining && FACTS.winterTime) ? FACTS.wearSunGlasses = false : FACTS.darkClouds = false ;
       FACTS.summerTime ? FACTS.weatherIsHot = true : FACTS.weatherIsHot = false;
    }
    while(numberOfFacts!=countFACTS())
 }

  addFACTS();
  FACTS.itsRaining = true;
  FACTS.weatherIsCold = true;
  infer();
  printFACTS();
</script>
```

Appendix B

```
########################################
# Standard ANN example written in PYTHON
########################################

import numpy as np
###########################
# ACTIVATION FUNCTIONS
###########################

# sigmoid function
def nonlin(x,deriv=False):
    if(deriv==True):
        return x*(1-x)
    return 1/(1+np.exp(-x))


###########################
# DEFINE ANN AND IO
###########################

# input dataset converges
X = np.array([[0,0,1],[1,1,1],[1,0,1],[0,1,1]])
# input dataset fails to coverge
#X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# output dataset
y = np.array([[0,1,1,0]]).T

###########################
# INTIALISE AND SEED ANN
###########################

np.random.seed(1)

# initialize weights randomly with mean 0
syn0 = 2*np.random.random((3,1)) - 1

############################
# SEEK CONVERGENCE (TRAIN)
############################

for iter in range(1000):
    # forward propagation
    l0 = X
    l1 = nonlin(np.dot(l0,syn0))

    # how much did we miss?
    l1_error = y - l1

    # multiply how much we missed by the
    # slope of the activation function at the values in l1
    l1_delta = l1_error * nonlin(l1,True)

    # update weights
    syn0 += np.dot(l0.T,l1_delta)

###########################
# REPORT RESULTS
###########################

print ("Output After Training:")
print (l1)
```

BSc Computer Science Year 2
2018-2019
Computational Algorithms

Assignment 2

Date Set: 10th December 2018

Course Tutor: Richard Henson

Feedback Sheet

Student's Name: **%**

	Max Mark	Awarded
Task 1:	**50**	
Knowledge and Understanding	20	
Analysis Discussion and Evaluation	20	
Expression Organisation and Presentation	10	
Task 2:	**25**	
Analysis and Translation of code	10	
Implementation Enhancements	10	
Supporting documentation	5	
Task 3:	**25**	
Python Implementation of Activation functions	10	
Analysis of Results	10	
Supporting documentation	5	
Total:	**100**	

Comments:

Marking scheme

Each practical Task 2 and 3 are graded out of a maximum of 25 marks and banded according to the following criteria:

Banding:	**Description:**
0-9 marks (fail)	Missing answer and/or code and very weak response to task. All written commentary lacks coherence and all theoretical elements are missing. Little or no referencing. Answers, if any, not relevant to task with inappropriate coding or irrelevant examples of coding as appropriate.
10-12 (below average pass)	Relevant response to task, but missing elements and only partially working code where required. Overall, there is a weak account of theoretical elements as required. Some evidence of referencing. Tendency to be very descriptive. Does not demonstrate understanding of implementation language or mastery of chosen re-implementation language or understanding of the modelling language.
13-14 (pass)	Reasonable response to task, but with some missing elements with some code weakness apparent. Reasonable theoretical account with references included. Mostly descriptive, but some analytical content included in responses. Surface level understanding of A.I. Some understanding of implementation language and mastery of chosen and/or modelling language.
15-17 (good pass)	Good response to task with all elements covered although may show some weaknesses in coding. Well referenced commentary with tendency to be more analytical rather than purely descriptive. Limited evidence of working beyond the curriculum, but with a good knowledge of A.I. Good understanding both implementation of re-implementation language. .
18+ (excellent pass)	Concise responses to task with efficient well-documented code where appropriate showing mastery of development environment. Well-argued discussions thoroughly referenced throughout. Good mathematical content where relevant. All discussions analytical in nature with concise descriptive accounts given where necessary. Clear evidence of research beyond the module curriculum and appropriately applied to the task. Complete understanding of task and its relevance to A.I. Shows a deep understanding of the subject matter.

NB: Double the banding marks for Task 1