|  |  |  |
| --- | --- | --- |
|  | **FACULTY of COMPUTING, ENGINEERING & SCIENCE** | Final mark awarded:\_\_\_\_\_ |

**Assessment Cover Sheet and Feedback Form 2016/17**

|  |  |  |  |
| --- | --- | --- | --- |
| Module Code:CS3S602 | Module Title: Modern Applications of AI | | Module Lecturer: C W Morris |
| Assessment Title and Tasks: Identification Systems | | | Assessment No. 2 of 2 |
| No. of pages submitted in total including this page:  Completed by student | | | Word Count of submission  (if applicable): Completed by student |
| Date Set: 10/02/17 | | Submission Date:24/03/17 | Return Date: Normally within 20 working days |

|  |  |
| --- | --- |
| ***Part A: Record of Submission (to be completed by Student)*** | |
| **Extenuating Circumstances**  If there are any exceptional circumstances that may have affected your ability to undertake or submit this assignment, make sure you contact the Advice Centre on your campus prior to your submission deadline. | |
| **Fit to sit policy**:  The University operates a fit to sit policy whereby you, in submitting or presenting yourself for an assessment, are declaring that you are fit to sit the assessment. You cannot subsequently claim that your performance in this assessment was affected by extenuating factors. | |
| **Plagiarism and Unfair Practice Declaration:**  By submitting this assessment, you declare that it is your own work and that the sources of information and material you have used (including the internet) have been fully identified and properly acknowledged as required[[1]](#footnote-1). Additionally, the work presented has not been submitted for any other assessment. You also understand that the Faculty reserves the right to investigate allegations of plagiarism or unfair practice which, if proven, could result in a fail in this assessment and may affect your progress. | |
| **Intellectual Property and Retention of Student Work:**  You understand that the University will retain a copy of any assessments submitted electronically for evidence and quality assurance purposes; requests for the removal of assessments will only be considered if the work contains information that is either politically and/or commercially sensitive (as determined by the University) and where requests are made by the relevant module leader or dissertation supervisor. | |
| **Details of Submission:**  Note that all work handed in after the submission date and within 5 working days will be capped at 40%[[2]](#footnote-2). No marks will be awarded if the assessment is submitted after the late submission date unless extenuating circumstances are applied for and accepted (Advice Centre to be consulted). | |
| You are required to acknowledge that you have read the above statements by writing your student number(s) in the box: | Student Number(s):  13017861 |

**IT IS YOUR RESPONSIBILITY TO KEEP RECORDS OF ALL WORK SUBMITTED**

|  |
| --- |
| **Part B: Marking and Assessment**  **(to be completed by Module Lecturer)** |
| This assignment will be marked out of 100%  This assignment contributes to 25% of the total module marks.  This assignment is bonded. |
| **Assessment Task:**   1. Design and code a simple nearest neighbour identification system. Use the iris data set (data in learning schedule on Blackboard) to test and demonstrate your program. Report on:  * your development and your findings * how your results relate to theory/expectation * the process and solution compared with a neural network solution  1. This task requires you to use all the experience that you have gained to analyse a real data set from scratch. You should note that you may need to reformat and/or pre-process the data set. You may also run into problems with limitations of the NN package: this is typical of real world applications and you should find ways to work around these problems.   The data set provided is from a medical diagnosis system with data from 4 conditions.   * This is data from 4 conditions and there are 200 examples from each in the file. * Analyse these data sets using a BP neural network and optimise the networks where appropriate. This will require experimentation with the architecture and training parameters of the neural network and the amount of data to use for training. * You will need to design all experiments carefully and justify all runs. Report on your experiments - note that you must summarise the results and present them in a readable format. Do not submit pages of raw output. Critically review the process and the outcomes of your experiments both from a purely neural network perspective and a practical application/problem perspective.  1. Write a short critical review (approximately 3 pages) of the use of neural networks as a problem solving paradigm in modern computing, highlighting the main features of their typical application areas.   **Submission Instructions**  The report should be a single document, word processed in Word format (this **excludes** Open Office and PDF), and it should present your findings clearly and concisely. Take care that code is neatly laid out in the report – correctly formatted and with all code in Courier New font.  You must submit **only** an electronic version of the coursework (via Blackboard) which may be submitted to a plagiarism detection system.  All sources of information **must** be fully referenced and full URLs of all Internet sources must be given. Use Harvard referencing style. |
| **Learning Outcomes to be assessed** (as specified in the validated module descriptor <http://icis.southwales.ac.uk>):  Demonstrate knowledge, comprehension and discernment in the effective application of AI paradigms to common problems. |
| **Grading Criteria:**   |  |  | | --- | --- | | **Assessment Criteria** | | | **Performance Level** | **Criteria** | | Fail  (< 40%) | Unacceptable documentation has been handed in. Poor presentation of material. KNN program has bugs and does not function as required. Practical use of neural networks not demonstrated. Critical review short or content shallow. Dated material used for the review. | | PASS  (40% - 59%) | A basic level of understanding of the application of neural networks is demonstrated. Some experimentation is evident but little evidence of a logical and systematic approach to experimentation and reporting of results. KNN program working in a very basic form. Minimal documentation of code presented. Critical review shallow and mainly repeats lecture material or uses material that is now outdated. | | MERIT  (60% - 69%) | A good report showing evidence of a good understanding of the application of neural networks. A good approach to experimental design and a clear set of results presented and discussed in a logical manner. KNN program works efficiently and is suitably documented to allow it to be easily understood. Critical review shows evidence of some up to date reading and some good understanding of the field. | | DISTINCTION  (70% +) | A well-presented report that shows an excellent understanding of the application of neural networks. A clear experimentation methodology is presented with a solid discussion of presented results. High standard of presentation of all findings and good insights into applications of neural networks are evident. The KNN program is well designed and efficiently implemented. Good documentation is provided to support the code. The critical review covers a range of up to date and relevant material and shows insight into the applications of neural networks. | |

|  |  |  |
| --- | --- | --- |
| **Indicative Marking Scheme** | **Marks Available** | **Marks Awarded** |
| **k- nearest neighbour program** | 30 |  |
| **Basic MLP training and testing** | 50 |  |
| **NN critical review** | 20 |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
| **Part B: Marking and Assessment**  **(to be completed by Module Lecturer)** |
| **Assessment Task:** |
| **Learning Outcomes to be assessed** (as specified in the validated module descriptor <https://icis.southwales.ac.uk/> ): |
| **Grading Criteria:** |

|  |  |  |
| --- | --- | --- |
| **Feedback/feed-forward** (linked to assessment criteria):   * Areas where you have done well: * Feedback from this assessment to help you to improve future assessments: * Other comments | | |
| **Mark:** | **Marker’s Signature:** | **Date:** |
| **Work on this module has been marked, double marked/moderated in**  **line with USW procedures.** | | |
| *Provisional mark only: subject to change and/or confirmation by the Assessment Board* | | |

|  |  |
| --- | --- |
| **Part C: Reflections on Assessment**  **(to be completed by student – optional)** | |
| **Use of previous feedback:**  In this assessment, I have taken/took note of the following points in feedback on previous work: | |
| **Please indicate which of the following you feel/felt applies/applied to your submitted work**   * A reasonable attempt. I could have developed some of the   sections further.   * A good attempt, displaying my understanding and learning, with   analysis in some parts.   * A very good attempt. The work demonstrates my clear   understanding of the learning supported by relevant literature and scholarly work with good analysis and evaluation.   * An excellent attempt, with clear application of literature and   scholarly work, demonstrating significant analysis and evaluation. | |
| **What I found most difficult about this assessment:** |  |
| **The areas where I would value/would have valued feedback:** |  |

AI Assignment 2

Part 1:- Nearest Neighbour

Design of code:

The nearest neighbour code is supposed to be designed to read in the files from the training and test data into two arrays then using that data look at the values of each line of data and find the values each is closest to in the data and decide which group is the nearest one it should be assigned to.

Pseudo Code:

for (load training array x data){

for (load training array inputs and outputs(7)){

place data into training data file

}

}

for (load test array x){

for (load test array inputs and outputs(7)){

place data into test data file

}

}

for (load X test data){

for (load x training data){

set sum to 0

for (run for the amount of inputs){

find the power found between the data in the training and the test array and add it and store it into sum

}

set the distance array to the square root of sum

}

}

set minimum value to 9999.0

for (load x distance array){

if the minimum is greater than the value stored in the distance array{

set the minimum to the distance array value

store the index value

}

}

for (load outputs information{

if the distance array index value and the output both equal 1{

output the category that the test point belongs to

}

}

Actual Code:

#include <iostream>

#include <fstream>

#include <cmath>

using namespace std;

// part 1

int main(){

fstream infile("trainingdata.txt", ios::in);

fstream infile2("testdata.txt", ios::in);

float trainingArray[50][7];

float testArray[25][7];

float distanceArray[50];

int index;

for (int i = 0; i < 50; i++){

for (int j = 0; j < 7; j++){

infile >> trainingArray[i][j];

}

}

for (int i = 0; i < 25; i++){

for (int j = 0; j < 7; j++){

infile2 >> testArray[i][j];

}

}

for (int i = 0; i < 25; i ++){

for (int j = 0; j < 50; j++)

{

float sum = 0.0;

for (int k = 0; k < 4; k++){

sum += pow(trainingArray[j][k]-testArray[i][k], 2);

}

distanceArray[j] = sqrt (sum);

}

float min = 9999.0;

for (int p = 0; p < 50; p++){

if (min > distanceArray[p]){

min = distanceArray[p];

index = p;

}

}

for (int m = 4; m < 7; m++){

if (trainingArray[index][m] == 1){

cout << "test point " << i + 1 << " is catagory " << m-3 << endl;

}

}

}

// 24 out of 25 correct

return 0;

}

Development and findings:

The tests for this set of data were done by checking how the neural network guesses the test data given after using the training data with the test data using 25 out of the maximum 75 and the training data using the remaining 50. The training data was done using 5 hidden nodes and 2000 cycles.

4 in – 5 hidden – 3 out (2000 cycles)

Input patterns included

Teaching output included

#1.1

0.224 0.624 0.067 0.043

1 0 0

0.96567 0.06905 0

#2.1

0.749 0.502 0.627 0.541

0 1 0

0.02951 0.9802 0.01034

#3.1

0.557 0.541 0.847 1

0 0 1

0.00378 0.00022 0.99991

#4.1

0.11 0.502 0.051 0.043

1 0 0

0.96298 0.07391 0

#5.1

0.722 0.459 0.663 0.584

0 1 0

0.0203 0.84945 0.11449

#6.1

0.776 0.416 0.831 0.831

0 0 1

0.00491 0.00054 0.99972

#7.1

0.196 0.667 0.067 0.043

1 0 0

0.96676 0.06466 0

#8.1

0.612 0.333 0.612 0.584

0 1 0

0.02469 0.96398 0.02614

#9.1

0.612 0.416 0.812 0.875

0 0 1

0.0045 0.0004 0.99981

#10.1

0.055 0.584 0.067 0.082

1 0 0

0.96233 0.07067 0

#11.1

0.557 0.541 0.627 0.624

0 1 0

0.01779 0.6295 0.29771

#12.1

0.165 0.208 0.592 0.667

0 0 1

0.01355 0.34594 0.69884

#13.1

0.027 0.376 0.067 0.043

1 0 0

0.95628 0.08967 0

#14.1

0.639 0.376 0.612 0.498

0 1 0

0.03389 0.99361 0.00337

#15.1

0.667 0.208 0.812 0.71

0 0 1

0.00637 0.00294 0.99841

#16.1

0.306 0.71 0.086 0.043

1 0 0

0.96688 0.0669 0

#17.1

0.196 0 0.424 0.376

0 1 0

0.06192 0.99952 0.00014

#18.1

0.612 0.502 0.694 0.792

0 0 1

0.00713 0.00346 0.9976

#19.1

0.137 0.416 0.067 0

1 0 0

0.96121 0.0808 0

#20.1

0.471 0.082 0.51 0.376

0 1 0

0.05487 0.99949 0.0002

#21.1

0.694 0.416 0.761 0.831

0 0 1

0.0056 0.00099 0.99944

#22.1

0.416 0.831 0.035 0.043

1 0 0

0.97155 0.05906 0

#23.1

0.361 0.376 0.439 0.498

0 1 0

0.06159 0.99854 0.00029

#24.1

0.416 0.333 0.694 0.957

0 0 1

0.00472 0.00054 0.99975

#25.1

0.306 0.792 0.051 0.125

1 0 0

0.9671 0.06629 0

The findings received in the predictions records for this set of nodes is that the neural network managed to successfully get all 25 out of 25 predictions correctly.

Nearest Neighbour output:



The nearest neighbours output managed to get 24 out of 25 of the points correctly.

Relation to expectations:

The expectations of the outputs was that the neural network would get more answers right than the nearest neighbour as the neural network would train through the data and learn the logic behind it. The nearest neighbour program just runs through the data once making its best guess by assigning the point to the nearest group of points around it which is not always necessarily the best option. The expectations turned out to be right as the neural network managed to successfully group all of the correct points into the correct category.

Process and solution compared with a neural network solution:

The process of the Nearest neighbour solution is to look at each data point separately and assign it to the group the point is nearest to. Whereas the neural network solution requires it to use similar data it has run through similar to find the best fitting category for that point of data. The comparison to draw here is that each part is working off of different amounts of data to come to a conclusion. This means that while the nearest neighbour algorithm is using the proximity of the data to come to its conclusion this is not always the most sensible of choices for the algorithm to make as groups of data can be very close together but are still separate. The neural network solution however has learned via its run through of similar data whether the data should be added to one group or the other.

Part 2: - Analysis of real Data

For the analysis of each set of the outputs instead of going through all 200 outputs and since outputs are grouped together in each section, for each I have decided to look at and display the first 10 sets of data from each output section. The data used has 800 total split into 600 that is used for the training data and 200 used to test the trained data against. The data uses 6 inputs and 4 outputs with tests of various different amounts of hidden nodes using 5 originally, then 10 and finally 2 tested with 2000 cycles each. I have also decided to run the first set of data with the test data running at 5000 cycles.

Data outputs:

6 in – 5 hidden – 4 out (2000 cycles)

#1.1

0.21155 0.11733 0.18388 0.03568 0.08028 0.24632

0 0 0 1

0.00007 0.95877 0.00581 0.00406

#2.1

0.21933 0.09674 0.17725 0.03842 0.04643 0.20675

0 0 0 1

0.0001 0.98272 0.00032 0.1522

#3.1

0.2422 0.12968 0.23786 0.10635 0.11641 0.25318

0 0 0 1

0.00023 0.68111 0.00043 0.08096

#4.1

0.22139 0.1251 0.21384 0.04002 0.08279 0.23945

0 0 0 1

0.00004 0.92668 0.00029 0.19141

#5.1

0.23054 0.12739 0.22825 0.04254 0.08485 0.24151

0 0 0 1

0.00003 0.87712 0.00013 0.40932

#6.1

0.24334 0.11046 0.21796 0.04368 0.08691 0.23945

0 0 0 1

0.00004 0.89976 0.00015 0.3483

#7.1

0.24586 0.10864 0.20515 0.05946 0.07936 0.22459

0 0 0 1

0.00009 0.95569 0.00022 0.20157

#8.1

0.21453 0.1283 0.2111 0.04803 0.06884 0.23603

0 0 0 1

0.00005 0.95913 0.0005 0.10029

#9.1

0.21979 0.10086 0.21064 0.03545 0.00869 0.23374

0 0 0 1

0.00006 0.98581 0.00036 0.133

#10.1

0.25752 0.1251 0.23008 0.05329 0.10017 0.25272

0 0 0 1

0.00004 0.88997 0.00025 0.17935

#51.1

0.38186 0.19682 0.16421 0.11517 0.13615 0.32271

0 0 1 0

0.00297 0.35503 0.99989 0

#52.1

0.41977 0.2226 0.23373 0.16295 0.18747 0.35961

0 0 1 0

0.00221 0.09999 0.99972 0

#53.1

0.32953 0.17761 0.16522 0.09192 0.10708 0.29869

0 0 1 0

0.00153 0.57485 0.99927 0

#54.1

0.28807 0.14171 0.11492 0.0722 0.07902 0.23727

0 0 1 0

0.00429 0.78469 0.98967 0

#55.1

0.41244 0.20541 0.18519 0.14424 0.19328 0.32119

0 0 1 0

0.00343 0.24339 0.99977 0

#56.1

0.36846 0.20592 0.17179 0.10708 0.11694 0.29818

0 0 1 0

0.00157 0.59719 0.99955 0

#57.1

0.33256 0.17508 0.15688 0.09166 0.12023 0.29364

0 0 1 0

0.00169 0.56172 0.99933 0

#58.1

0.23095 0.10734 0.08433 0.04743 0.06007 0.27999

0 0 1 0

0.01084 0.31595 0.99984 0

#59.1

0.26735 0.14222 0.13691 0.0631 0.0631 0.2901

0 0 1 0

0.00224 0.57652 0.99938 0

#60.1

0.23979 0.12958 0.11593 0.05703 0.08509 0.2719

0 0 1 0

0.00286 0.54589 0.99907 0

#101.1

0.24409 0.13716 0.19505 0.05728 0.09242 0.22235

0 1 0 0

0.00006 0.98408 0.00097 0.03419

#102.1

0.18873 0.07549 0.13716 0.06209 0.06942 0.21603

0 1 0 0

0.00397 0.7889 0.01317 0.00075

#103.1

0.28807 0.16042 0.22867 0.08686 0.11669 0.24131

0 1 0 0

0.00006 0.97519 0.001 0.02444

#104.1

0.20921 0.10177 0.16598 0.0449 0.0454 0.22741

0 1 0 0

0.00021 0.98149 0.00669 0.00261

#105.1

0.35784 0.1766 0.16295 0.1177 0.17053 0.18797

0 1 0 0

0.0006 0.98911 0.00732 0.00087

#106.1

0.27013 0.16649 0.23347 0.07018 0.07018 0.24485

0 1 0 0

0.00004 0.98998 0.0013 0.0193

#107.1

0.16775 0.07549 0.14904 0.0497 0.04945 0.21805

0 1 0 0

0.00116 0.86093 0.00377 0.00658

#108.1

0.19176 0.08155 0.13641 0.0401 0.05501 0.20718

0 1 0 0

0.00057 0.95882 0.00463 0.00426

#109.1

0.25774 0.13716 0.21224 0.06942 0.09545 0.23651

0 1 0 0

0.00007 0.9735 0.00093 0.03211

#110.1

0.21047 0.11189 0.16851 0.03833 0.03959 0.21906

0 1 0 0

0.00011 0.99105 0.00325 0.00729

#151.1

0.19176 0.07346 0.14601 0.14475 0.05223 0.2044

1 0 0 0

0.99657 0.00529 0.00847 0.00007

#152.1

0.21704 0.08762 0.12073 0.13363 0.08029 0.17255

1 0 0 0

0.98606 0.02113 0.00263 0.00034

#153.1

0.21856 0.10986 0.14854 0.168 0.1038 0.17584

1 0 0 0

0.99568 0.00731 0.00099 0.00095

#154.1

0.29414 0.13514 0.08054 0.07902 0.12629 0.15587

1 0 0 0

0.00962 0.93004 0.07421 0.00002

#155.1

0.20921 0.09217 0.12301 0.14652 0.08888 0.18115

1 0 0 0

0.99604 0.00643 0.0111 0.00004

#156.1

0.20769 0.09571 0.13312 0.13767 0.09444 0.18266

1 0 0 0

0.96533 0.02414 0.00373 0.00033

#157.1

0.19176 0.09419 0.13615 0.15334 0.06664 0.17432

1 0 0 0

0.99769 0.0054 0.00144 0.00057

#158.1

0.27013 0.10531 0.15713 0.16775 0.10001 0.19834

1 0 0 0

0.99173 0.02063 0.00689 0.00006

#159.1

0.27897 0.11441 0.15739 0.17458 0.12149 0.19909

1 0 0 0

0.991 0.01766 0.01188 0.00003

#160.1

0.20921 0.1038 0.15764 0.15562 0.11087 0.19884

1 0 0 0

0.96686 0.01383 0.00378 0.00036

6 in – 10 hidden – 4 out (2000 cycles)

#1.1

0 0 0 1

0.00009 0.95109 0.02073 0.00216

#2.1

0 0 0 1

0.00131 0.96967 0.00065 0.22312

#3.1

0 0 0 1

0.00186 0.71769 0.00077 0.16446

#4.1

0 0 0 1

0.00003 0.90596 0.00185 0.16784

#5.1

0 0 0 1

0.00002 0.86761 0.00095 0.43601

#6.1

0 0 0 1

0.00005 0.8753 0.00074 0.44165

#7.1

0 0 0 1

0.00063 0.93435 0.00044 0.36866

#8.1

0 0 0 1

0.00014 0.94986 0.00184 0.09702

#9.1

0 0 0 1

0.00077 0.9846 0.00084 0.16755

#10.1

0 0 0 1

0.00003 0.88172 0.00139 0.20363

#51.1

0 0 1 0

0.0049 0.34619 0.9999 0

#52.1

0 0 1 0

0.0069 0.0871 0.99949 0

#53.1

0 0 1 0

0.0023 0.82501 0.9983 0

#54.1

0 0 1 0

0.00934 0.93509 0.98637 0

#55.1

0 0 1 0

0.00622 0.16862 0.99949 0

#56.1

0 0 1 0

0.0013 0.8356 0.99905 0

#57.1

0 0 1 0

0.00231 0.79421 0.99837 0

#58.1

0 0 1 0

0.03131 0.66384 0.99993 0

#59.1

0 0 1 0

0.00511 0.88926 0.99913 0

#60.1

0 0 1 0

0.00675 0.8562 0.99826 0

#101.1

0 1 0 0

0.00022 0.97281 0.0024 0.02949

#102.1

0 1 0 0

0.05153 0.75796 0.01184 0.00103

#103.1

0 1 0 0

0.00017 0.9715 0.00202 0.02351

#104.1

0 1 0 0

0.00261 0.97646 0.01139 0.00166

#105.1

0 1 0 0

0.00079 0.99004 0.01097 0.00013

#106.1

0 1 0 0

0.00015 0.99052 0.00342 0.01016

#107.1

0 1 0 0

0.0172 0.85355 0.00477 0.00904

#108.1

0 1 0 0

0.0075 0.94228 0.00768 0.00354

#109.1

0 1 0 0

0.00029 0.96259 0.00196 0.03594

#110.1

0 1 0 0

0.0012 0.98767 0.00796 0.0034

#151.1

1 0 0 0

0.99176 0.01584 0.01228 0.0001

#152.1

1 0 0 0

0.96478 0.04698 0.00849 0.00025

#153.1

1 0 0 0

0.98476 0.0184 0.00374 0.00075

#154.1

1 0 0 0

0.00846 0.97172 0.17947 0

#155.1

1 0 0 0

0.98503 0.01832 0.02318 0.00004

#156.1

1 0 0 0

0.94978 0.05132 0.00766 0.00033

#157.1

1 0 0 0

0.99071 0.01616 0.00468 0.0006

#158.1

1 0 0 0

0.9773 0.03509 0.0207 0.00002

#159.1

1 0 0 0

0.9718 0.03154 0.03224 0.00001

#160.1

1 0 0 0

0.96104 0.03017 0.00613 0.00041

6 in – 2 hidden – 4 out (2000 cycles)

#1.1

0 0 0 1

0 0.99227 0.00006 0.00649

#2.1

0 0 0 1

0.00008 0.91137 0 0.21095

#3.1

0 0 0 1

0.02165 0.10169 0 0.5845

#4.1

0 0 0 1

0.00003 0.95694 0 0.18127

#5.1

0 0 0 1

0.00008 0.9104 0 0.37577

#6.1

0 0 0 1

0.00012 0.87936 0 0.37891

#7.1

0 0 0 1

0.00037 0.75068 0 0.41143

#8.1

0 0 0 1

0.00004 0.94571 0 0.1536

#9.1

0 0 0 1

0.00005 0.93676 0 0.17359

#10.1

0 0 0 1

0.00005 0.94089 0 0.18784

#51.1

0 0 1 0

0.06253 0.05062 0.99959 0

#52.1

0 0 1 0

0.06567 0.04861 0.99955 0

#53.1

0 0 1 0

0.01488 0.14803 0.99786 0

#54.1

0 0 1 0

0.00273 0.40317 0.98345 0

#55.1

0 0 1 0

0.05539 0.05575 0.99948 0

#56.1

0 0 1 0

0.02194 0.11265 0.99871 0

#57.1

0 0 1 0

0.01519 0.14596 0.9979 0

#58.1

0 0 1 0

0.04593 0.0646 0.99934 0

#59.1

0 0 1 0

0.01492 0.14771 0.99772 0

#60.1

0 0 1 0

0.00893 0.20748 0.99524 0

#101.1

0 1 0 0

0.00001 0.98336 0.00001 0.03893

#102.1

0 1 0 0

0.00589 0.25173 0.00015 0.00405

#103.1

0 1 0 0

0.00002 0.9748 0.00001 0.03166

#104.1

0 1 0 0

0.00002 0.96793 0.0001 0.0044

#105.1

0 1 0 0

0.00001 0.98975 0.00154 0.00021

#106.1

0 1 0 0

0 0.9953 0.00003 0.01131

#107.1

0 1 0 0

0.00259 0.39111 0.00001 0.04546

#108.1

0 1 0 0

0.0001 0.90068 0.00005 0.00992

#109.1

0 1 0 0

0.00003 0.95853 0.00001 0.05427

#110.1

0 1 0 0

0.00001 0.98979 0.00005 0.0077

#151.1

1 0 0 0

0.99951 0.00001 0.0066 0.00018

#152.1

1 0 0 0

0.9773 0.00027 0.00172 0.00059

#153.1

1 0 0 0

0.99724 0.00005 0.00055 0.00231

#154.1

1 0 0 0

0.00002 0.97375 0.05398 0.00001

#155.1

1 0 0 0

0.99731 0.00005 0.00957 0.00011

#156.1

1 0 0 0

0.98158 0.00023 0.00084 0.00127

#157.1

1 0 0 0

0.9981 0.00004 0.0011 0.00113

#158.1

1 0 0 0

0.99753 0.00005 0.014 0.00007

#159.1

1 0 0 0

0.99755 0.00005 0.02186 0.00005

#160.1

1 0 0 0

0.99614 0.00006 0.0006 0.00203

6 in – 5 hidden – 4 out (5000 cycles)

#1.1

0 0 0 1

0 0.99524 0.00073 0.00047

#2.1

0 0 0 1

0 0.94396 0.00014 0.16218

#3.1

0 0 0 1

0 0.93255 0.00003 0.03357

#4.1

0 0 0 1

0 0.8946 0.00005 0.22165

#5.1

0 0 0 1

0 0.60942 0.00002 0.7434

#6.1

0 0 0 1

0 0.6856 0.00002 0.69536

#7.1

0 0 0 1

0 0.87475 0.00004 0.30238

#8.1

0 0 0 1

0 0.96117 0.00009 0.04665

#9.1

0 0 0 1

0 0.95063 0.00008 0.13019

#10.1

0 0 0 1

0 0.88237 0.00003 0.27457

#51.1

0 0 1 0

0.00402 0.00112 0.99965 0

#52.1

0 0 1 0

0.00435 0.00345 0.99797 0

#53.1

0 0 1 0

0.00136 0.01201 0.99739 0

#54.1

0 0 1 0

0.00064 0.16568 0.98211 0

#55.1

0 0 1 0

0.00278 0.00404 0.99822 0

#56.1

0 0 1 0

0.00135 0.00391 0.99921 0

#57.1

0 0 1 0

0.00128 0.01296 0.99728 0

#58.1

0 0 1 0

0.00744 0.00429 0.99969 0

#59.1

0 0 1 0

0.0019 0.01533 0.99812 0

#60.1

0 0 1 0

0.00166 0.04265 0.99537 0

#101.1

0 1 0 0

0 0.99042 0.00021 0.00662

#102.1

0 1 0 0

0.00097 0.9699 0.00136 0.00037

#103.1

0 1 0 0

0 0.99312 0.00013 0.00354

#104.1

0 1 0 0

0.00001 0.99624 0.00133 0.00023

#105.1

0 1 0 0

0.00009 0.99529 0.00231 0.00005

#106.1

0 1 0 0

0 0.99486 0.00031 0.00108

#107.1

0 1 0 0

0.00011 0.97337 0.00072 0.0016

#108.1

0 1 0 0

0.00004 0.99077 0.00171 0.001

#109.1

0 1 0 0

0 0.98924 0.00013 0.00743

#110.1

0 1 0 0

0 0.99658 0.00138 0.00051

#151.1

1 0 0 0

0.99897 0.01701 0.00141 0.00045

#152.1

1 0 0 0

0.99173 0.02962 0.00275 0.0048

#153.1

1 0 0 0

0.99765 0.01475 0.0011 0.00595

#154.1

1 0 0 0

0.00147 0.87623 0.1965 0

#155.1

1 0 0 0

0.99822 0.01582 0.00324 0.00062

#156.1

1 0 0 0

0.98309 0.04754 0.00178 0.00268

#157.1

1 0 0 0

0.99912 0.00988 0.00171 0.00235

#158.1

1 0 0 0

0.98958 0.05588 0.0015 0.00133

#159.1

1 0 0 0

0.986 0.06708 0.00164 0.00082

#160.1

1 0 0 0

0.98343 0.05454 0.00081 0.00174

Review of outputs:

6-5-4:

The test data doesn’t train very well in this version as the neural network has trouble when guessing the correct output at the beginning of the test data thinking that all values in column four are others. However later on in the other parts it does start to guess the correct data with quite a high certainty.

6-10-4:

The test data in this version still doesn’t train very well in the beginning of this version however the neural network does lean a little more toward the correct answer than in previous versions but still gets it completely wrong. The rest of the data in doesn’t change its guess very much.

6-2-4:

While the data at the end of these tests does not change very much everything in the first second and third column appears to be successfully guessed correctly. It is still the beginning of the test data that the neural network has a problem with and this arrangement of nodes managed to confuse the network even more in most places and makes guesses it is sure is right in places that it definitely isn’t. However it does succeed to get one or two in this category correctly this time.

6-5-4 (5000 cycles)

Increasing the amount of cycles it goes through the data does seem to have the desired effect of increasing the amount of right guesses on the fourth column of data. However in these changes the network still isn’t completely certain that this data is correct and is usually guessing between two with the right answer only narrowly being right.

In conclusion this data isn’t very trainable in a neural network it should learn the data in the fourth column as easily as the rest but for some reason it just gets lost. The amount of wrong guesses can be minimised but not by enough to make this data able to be used effectively.

Part 3:- A Review on Neural Networks

Neural networks excel at recognising the patterns in data not just characters it does this by identifying differences within different types of data and then showing the probability using neural networks as a problem solving paradigm can be used to solve a lot of real world problems. The problems in the real world that can be solved with neural networks can be both complex and simple problems such as using a neural network to create a self-driving car as posted on the NVidia website (2015). This problem would require being solved by using a deep learning approach which would include solving problems like identifying how to make a machine drive whilst obeying the rules of the road such as how to identify different types of vehicles identifying stop signals, where to turn and when to pull out etc. Once the problem of a driveable car via a machine is solved by a neural network other neural networks could be created in order to analyse the data between cars driven by people and cars driven by machine. These neural networks could then analyse and predict how safe each of these can be and the resulting data can be compared to check the safety of both.

Another application of using a neural network in order to solve a problem is using one to try to recognise handwritten digits as explained by Nielsen (2017) that doing so in a human brain requires the use of multiple visual cortices in order to recognise the shapes of these numbers. In order for a neural network to successfully identify someone’s handwriting it would take the neural network learning the pattern through using an image recognition pattern on a large number of images on handwritten digits. The code that can be used to achieve this can be written in just 74 lines and can have a success rate of up to 96 percent. This sort of neural network is used by types of companies that include banks when processing cheques and the post office when trying to recognise written addresses.

The problem with using neural networks in general not just to solve problems is that the data selected for them must be selected carefully in order to avoid any faults in its logic. The further problem also occurs that should the neural network develop a fault in the first place it likely won’t be detected until the whole thing blows up with an error.

Some typical application areas of neural networks according to Prince, N. N. (2011) include in businesses for allocating resources and scheduling. Neural networks can also be used for marketing analysis purposes. In order to do this the neural network could be used when creating a model of the statistics obtained so that management can make decisions of these models when it comes to their customers. An example of an area that this would be useful is to assist in the marketing of airline seat allocations. The financial market would also be a good place that the neural networks could be used in order to feed in the information of customers to predict the sales of items in order to set prices higher or lower which is known as forecasting. Other areas a neural network could be useful in predictions would be in the medical area with an online physician application. This application could be used to store a large amount of medical files from various patients and then learn these patients’ specific data when it comes to their symptoms, diagnosis and the treatment for the case. This data can then be applied to new customers who enter the details of their symptoms to which the network would output its best thought on their diagnosis and treatment. In conclusion using a neural network to solve problems is widely used in multiple areas of businesses and beyond in order to sort their data. It is very useful when checking through huge amounts of data so that problems can be solved with minimal errors.

References:

Nielsen, M. (2017). *Neural Networks and Deep Learning*. [online] Neuralnetworksanddeeplearning.com. Available at: http://neuralnetworksanddeeplearning.com/chap1.html [Accessed 6 Mar. 2017].

Prince, N. N. (2017). [online] Available at: http://www.ipcsit.com/vol20/36-ICAIT2011-A4078.pdf [Accessed 20 Mar. 2017].

Shapiro, D. (2017). *How Deep Learning Will Accelerate Self-Driving Cars | NVIDIA Blog*. [online] The Official NVIDIA Blog. Available at: https://blogs.nvidia.com/blog/2015/02/24/deep-learning-drive/ [Accessed 5 Mar. 2017].

1. University Academic Misconduct Regulations [↑](#footnote-ref-1)
2. Information on exclusions to this rule is available from the Advice Centre at each Campus [↑](#footnote-ref-2)