Aston University
BIRMINGHAM UK

College of Engineering & Physical Sciences
Assignment Brief

CS4720 Computational Intelligence	Experimental Study
Harry Goldingay h.j.goldingay1@aston.ac.uk	https://wass.aston.ac.uk/pages/viewcalendar.page.php?cal_id=469

Assignment Brief/ Coursework Content:

This experimental study is based on the content delivered in weeks 1-7 in the module.

The assignment is designed to assess the learning outcomes:

1. Implement and evaluate the performance of different computational intelligence techniques and the impact of different problem representations on the performance of those techniques.
2. Choose and reason about appropriate computational intelligence techniques for a given problem, and explain the consequences of decisions. Where appropriate, use this understanding to design novel variants of computational intelligence techniques for solving the problem.

Context:

Note: This section gives some background information about the problem you will be working on and describes the problem itself. The aim is to motivate your work as well as to give you some ideas about where you might look for information about similar applications in the academic literature. However, you do not need to understand this context in depth to produce a good solution to the coursework – it would be possible to simply treat this as a black box optimization problem based on the information in the Problem Details section of the specification and still obtain a high mark. In particular, you do not need to understand neural networks (how they are structured and/or trained in traditional machine learning) unless this is important to your chosen source of novelty.

In the first half of this module, we have been studying a variety of Computational Intelligence algorithms for solving optimization problems. When discussing these algorithms in class, we have made reference to how they can be used to solve business problems directly (e.g., finding a good route connecting all destinations in a routing problem, finding a good design for an antenna array). However, they can also be combined with other computational methods to solve problems which, at first glance, might not have obvious relationships with optimization.

One common example of this is in machine learning. When trying to explain a pattern in a dataset using machine learning we pick a representation which we think might be able to explain the pattern. A representation will typically specify the general type of pattern we are looking for (for example, a straight line in a 2D graph). We then fit the model to the data that we have (if our model is a straight line, then this means choosing a specific line: for example, the line of best fit through the datapoints) by setting the model parameters (for a straight line, the gradient and intercept). We want our fitted model to explain the data “well” – where “well” means the best performance we can find in terms of some

performance measure (e.g. low mean squared error, high coefficient of determination). To summarise, given some (parameterised) machine learning representation we want to find a set of parameters which results in the best (optimal!) model according to some quality or loss function. Hopefully it is clear to you that this is an optimisation problem and, hence, why optimisation is a key component in machine learning.

In this problem, you will be optimising the weights and biases (a set of numerical parameters) of an Artificial Neural Network (ANN) with the aim of predicting the price of a car based on some information about that car. The problem is based on a dataset from Kaggle, but the data has been pre-processed to make it suitable for this task and so will look significantly different from the online dataset. In particular:

- it now contains 21 features (compared to the original 26).
- many of the features (including the price) have been transformed to make them suitable for consumption by an ANN so you should not expect the data to be easily interpretable.

The evaluation code given on Blackboard is based on an ANN with a hidden layer containing 2 nodes. Therefore, by default, you will be optimising 44 weights (21x2 connecting the inputs and the hidden layer and 2x1 connecting the hidden layer to the output) and 3 biases (2 for the hidden layer and 1 for the output). As stated in the note at the start of this section, you do not need to understand these details (which are taken care of in the evaluation code provided) and can just treat the problem as one of optimising an array of 47 double valued parameters.

The performance of the ANN will be measured in terms of Mean Squared Error (MSE) – the average squared difference between the predictions of the ANN given the features and the true values of the target variable. Again, calculating the MSE is taken care of in the code provided.

Problem Details:

In this assignment, you will architect **TWO novel Computational Intelligence solutions** aiming to find the weights and biases of an ANN which minimise MSE for predicting car prices.

A candidate solution will consist of the design of an ANN, along with its weights and biases. If using the evaluation code provided, which assumes a fixed structure for the ANN (described above), this means that a candidate solution will be an array of 47 double valued parameters, with the first 44 being the weights of the network and the last 3 being the biases.

The input data for the problem is contained in three CSV files: `train.csv`, `validation.csv` and `test.csv`. Each of these CSV files has 22 columns. Each row of a CSV file represents a car, with the first 21 columns being its features and the 22nd being a transformed version of its price ($column\ 22 = \ln(1 + price)$). You do not need to work with these files directly – they will be dealt with by the provided evaluation function (described below).

To evaluate the mean squared error of a candidate solution, you will need to call the `carpriceprediction.make_evaluator` function of the code provided. This function takes a string argument to select which of the three CSV files should be use. It will accept the arguments `'train'`,`' validation'` or `'test'`. It will return a function which in turn takes your weights and biases - an array of 47 real numbers as described above – as input. The function will return the MSE for those weights and biases on your selected dataset.

Assignment Details

You are expected to design and implement two different computational intelligence solutions to the problem and to compare these two solutions to identify the 'winning' solution. Your solutions can be based on a standard computational intelligence algorithm **but should have some 'novel' original element that you propose and create**.

The steps for creating the 'winning' solution, as discussed and practiced in the module, are:

- For each of the two approaches:
    - Create an appropriate representation for candidate solutions.
    - Apply the evaluation function.
    - Design and implement the operators.
    - Design and implement a run of your algorithm.
    - Perform preliminary experiments with various parameter settings, operators to improve the quality of your results.
    - Investigate, evaluate, and report on the ability of the approach to solve the problem.
- Design the experiments that will allow you to compare the two solutions.
- Compare the two approaches and propose, with justification, the winning solution.

You may base your novel algorithms on any of the methods discussed in the lectures (i.e. any particle swarm optimisation, ant colony optimisation, artificial immune system, evolutionary algorithm, genetic programming).

Note that at the time of release we have not covered genetic programming yet.

A Note on Performance:

Because of the problem of overfitting in machine learning, the best MSE obtained on the dataset used to optimise the parameters is not a good measure of performance. Essentially, we don't just want an ANN which is good at remembering the prices of cars it has already seen during optimization. Instead, we want one which is good at predicting the prices of cars it has never seen before. This is why you have been provided with three datasets, which you should use as follows:

- When optimising, you should use the `evaluate` method on the `train.csv` dataset.
- However, if you want to compare the performance of two algorithms you should take the parameters they find after optimisation on `train.csv` and evaluate them using unseen data (either `validation.csv` or `test.csv` – see the next bullet points for which).
- You can use `validation.csv` as many times as you wish. It would be typical to use it to select the most promising candidate algorithm from a set. For example, if you had designed two novel operators, you might use performance on the validation set to decide which one was most promising.
- You should use `test.csv` exactly once, at the end of your study. It should be used to compare the performance of your novel method and the baseline.

A Note on Novelty:

To repeat what has been said above: "*In this assignment, you will architect **TWO novel Computational Intelligence solutions…***". For the purposes of this coursework, a novel algorithm is one which differs in some non-trivial way from the standard algorithms introduced in the course. The novel aspect of your algorithm could be designed yourself, or could be something that you have taken and adapted from a literature source. Some examples of novel changes to you could make to an algorithm would be:

- a new solution representation,
- a new mutation or crossover operator for an EA,
- a new velocity update mechanism for PSO,

and other similar changes to those or other CI algorithms. You might also want to include more than one novel aspect in your solution.

In terms of scope, a reasonably well-justified modification to a single operator would be of "sufficient quality and complexity" (as per the 50-59 mark band, below). Note that the mark scheme looks at both quality and complexity. A more complex change to a standard CI technique would likely be eligible for more marks than a simple one; however, a really well-designed (and justified) small change would likely get you more marks than a more complex but less well-designed change. Of course if your original aspect is both high-quality and (appropriately) complex then that would likely be even better!

The evaluation code provided assumes a fixed structure for the ANN (a single hidden layer with two nodes). In this setting, the problem is to find a set of weights to maximise performance. However, another way to approach the problem would be to optimise both the network structure (number of hidden layers, number of nodes in each layer) alongside the weights. This would be a challenging problem and would require you to do independent research into neural networks and to rewrite the evaluation code. However, such a solution would be acceptable and a good attempt would likely show significant novel thinking.

There are many other ways of achieving novelty in the assignment and the above is not recommended unless you want a serious challenge. You may also want to draw on literature sources for inspiration. A couple examples of relevant papers to get you started are:

- Carvalho, M. and Ludermir, T.B., 2007, September. Particle swarm optimization of neural network architectures andweights. In *7th International Conference on Hybrid Intelligent Systems (HIS 2007)* (pp. 336-339). IEEE.
- Jin, Y., Okabe, T. and Sendhoff, B., 2004, June. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)* (Vol. 1, pp. 1-8). IEEE.

Common Pitfalls:

- Make sure that your chosen algorithms are both computational intelligence algorithms. Don't, for example, choose random search, local search or backpropagation as a candidate.
- Because of the word limit in the report, you need to be concise. Do not use your word limit explaining standard aspects of your chosen algorithms (e.g. the principles of evolutionary algorithms, the details of the standard PSO velocity function).
- Do, however, explain your own design choices in sufficient detail for your assessor to understand exactly what you have done – not just your general idea. Focus your explanation on any novel aspects you have introduced.
- Good quality analysis of results goes beyond simply stating what you have observed (e.g. algorithm A does outperform algorithm B). You should make an attempt to explain your algorithms' behaviour, drawing on evidence to support your explanations where possible.

Descriptive details of Assignment:

The assignment consists of three parts: (1) a report on your solution to the problem that are assessed, (2) a demonstration of your winning solution and (3) the source code which is not directly assessed:

Report

- **Report** title: *Firstname_Lastname_CS4720_CI_study*, with preferred submission format as a single doc, rtf, docx or pdf file

- The report is limited to at most 1000 words and can include additional tables or graphs presenting results, up to 6 pages in total. The report should be typeset and presented in A4 format, using any easy to read font (for example Arial), size 12.

- The report should be structured in the following sections:

    - **The proposed Computational Intelligence solutions**: Justify and explain your two CI solutions, including any illustrations that you consider appropriate.
    - **Experimental methodology**: Describe the design of your experiments, including any parameters and show your experimental results, including tables and/or graphs.
    - **Analysis of experimental results**: Interpret the results of the experiments, compare your two solutions and justify your choice of winner.
    - **References**

- Preferred reference style: Harvard referencing

Demonstration of winning solution

You are asked to demonstrate the working of your winning solution in a one-to-one meeting with one of the module tutors. You are expected to give a **5 minute demonstration** of your solution and then to answer questions.

Source code

Additionally, you are asked to **submit your source code**, as a single doc, rtf or pdf file with title: *Firstname_Lastname_CS4720_CI_code***,** for the sole purpose of originality checking.

Note that the source code itself will not be assessed. This module is not concerned with teaching programming. Programming is necessary to enable the experimental study in this assignment. As the assignment is individual, it is essential that the underlying code is original and your own work.

Recommended reading/ online sources:

- For the baseline methods, as well as experimental design and ideas for original elements:
  Agoston E Eiben, James E Smith (2015). Introduction to evolutionary computing. Springer.
- References to academic articles suitable as a starting point are given above under "A Note on Novelty".
- On Blackboard, you will find all relevant lecture material in the **Lecture material** folder within **Learning Resources** and all relevant lab material in the **Lab material** folder within **Learning Resources**.

Key Dates:

21/02/2025	Coursework set
24/04/2025 – 30/04/2025	Demo sessions, to be booked by each student as a single individual WASS appointment of 15 minutes
23/04/2025	Submission date for report and source code
21/05/2025	Expected feedback return date. Individual Feedback will be provided on Blackboard.

Submission Details:

- Demo: book session as described above (please monitor Blackboard announcements and emails about this)
- Report: Online on Blackboard
- Source code: Online on Blackboard

Marking Rubric:

1. **Explanation, quality and justification of solutions – weight 60%**

Note: marks for this aspect are informed by both your demonstration and report.

0-39 Brief, irrelevant, confused, incomplete. Does not come close to meeting the required learning outcomes.

40-49 Evidence that some learning outcomes have been achieved or most learning outcomes achieved partially. Although work may include brief signs of comprehension, it contains basic misunderstandings or misinterpretations, demonstrates limited ability to meet the requirements of the assessment.

50-59 The novel solution methods are comprehensibly described, although possibly with some omissions. The explanation and justification are of satisfactory depth and detail. The solutions are of satisfactory quality and complexity. Appropriate references are used.

60-69 Substantial and well-presented explanation and justification of solutions, founded on suitable literature. The original solutions indicate thinking beyond book-standard solutions.

70-79 Thorough and in-depth explanation and justification of solutions. Very good original solution demonstrating some novelty or insight, based on recent literature.

80-100 Excellent explanation and justification. Claims are fully supported with well-chosen references. Excellent original solution, demonstrating high levels of novelty and insight, stemming from, but beyond published literature.

2. **Experimental methodology – weight 20%**

0-39 Brief, irrelevant, confused, incomplete. Does not come close to meeting the required learning outcomes.

40-49 Experiments have been designed with the clear intention to compare the solutions, but are flawed such that conclusions drawn from them would be unreliable.

50-59 There is evidence of a suitable design of experiments and comparison of the two solutions.

60-69 The design of experiments follows the methods taught in the MSc programme and is performed competently.

70-79 Very good design of experiments indicating insight. The experiments are carefully executed, characterised by attention to detail.

80-100 Excellent design of experiments indicating in-depth understanding. Very proficient execution of experiments.

3. Analysis of experimental results – weight 20%

0-39 Brief, irrelevant, confused, incomplete. Does not come close to meeting the required learning outcomes.

40-49 Some analysis has been attempted. There is a clear attempt to draw conclusions about comparative quality but this is compromised by either inconsistency with the evidence presented or a lack of depth.

50-59 The results are correctly interpreted and conclusions drawn about the comparative quality of the solutions are of appropriate depth and are consistent with the evidence presented.

60-69 Well-justified conclusions about the relative quality of the two solutions are drawn. The analysis attempts to explain the differences in performance of the solutions and is supported by evidence from the experiments.

70-79 The analysis draws on the experimental data and thorough understanding of the solutions to convincingly explain observed performance differences.

80-100 In addition to the above, the insights gained from the analysis are substantial.