Biologically-Inspired Computation (F20BC/F21BC) Coursework

Due 3:30pm local time, Friday 21st November 2025

This is a pair-based assessment and is worth 40% of your overall course mark

Marks and individual feedback will be provided after the exam diet

Overview

The aim of this assessment is to increase your understanding of artificial neural networks (ANNs) and particle swarm optimisation (PSO), two biologically-inspired techniques which are taught in the course. It involves implementing both ANN and PSO, experimentally investigating how PSO can be used to optimise an ANN to carry out a specified task, and doing some wider reading.

Please read this document in full. It's quite long, but it includes a lot of important details and pointers that may have a significant impact on your mark.

To encourage discussion, the assessment involves working in pairs, i.e. 2 people. You need to choose a partner from the same level (i.e. an F20BC student cannot be paired with an F21BC student). You should report your team choice on Canvas by **Friday 26th September**:

- For Edinburgh students, see Edinburgh Information → Edinburgh Team Formation
- For Dubai students, see Dubai Information → Dubai Team Formation

Each member of a pair should contribute equally. We will be asking you to summarise your contributions when you submit your work, and we will rebalance the marks within a pair in cases where one member contributes substantially more than the other. We also reserve the right to hold individual vivas when it is unclear to us whether a student has met the learning outcomes.

You are allowed to use generative AI for the implementation part of the assessment – see the next page for more details on this. Generative AI should not be used for writing your report, or for carrying out experimental work.

This is assessed coursework. You are allowed to discuss this assignment with students outside of your pair, but you should not copy their work, and you should not share your own work with other students. We will be carrying out automated plagiarism checks on both code and text submissions.

Special note for using existing code or generative AI. If you are using code that you have not yourself written, then this must clearly be indicated, making clear which parts were not written by you and clearly stating where it came from. If your code is found elsewhere by the person marking your work (including in the output of large language models) and you have not mentioned this, you may find yourself having to go before a disciplinary committee.

Late submission and extensions. Late submissions will be marked according to the university's late submissions policy, i.e. a 30% deduction if submitted within 5 working days of the deadline, and a mark of 0% after that. The deadline for this work is not negotiable. If you are unable to complete the assignment by the deadline due to circumstances beyond your control (e.g. illness or family bereavement), you should complete and submit a mitigating circumstances application. You can find further information at this link.

What you are asked to do:

1. Implement a multi-layer ANN architecture from scratch
2. Implement PSO from scratch
3. Couple together your ANN and PSO implementations
4. Train the ANN to solve a specified problem
5. Experimentally investigate the effect of ANN and PSO hyperparameters
6. Write a short report and submit both the report and your code to Canvas
7. Indicate your individual contributions on the group signing sheet

The tasks are described in more detail below. You can start work on Task 1 of the assessment once ANNs have been covered in lectures. At that point, you can also write the evaluation code for Task 4, and can test this code on ANNs with random weights. PSO will not be covered in lectures until week 7, though if you are eager to get started on this part, the introductory lecture on PSO will be made available before then under Week 7 on Canvas.

Implementation should be done using a language of your choice (e.g. Python, Java, Matlab, C, C++). The aim of this assessment is for you to understand how biologically-inspired approaches are implemented at the code level, so you are strongly encouraged not to use existing ANN or PSO libraries/APIs, since doing so will significantly reduce your marks. However, you can use libraries for handling data, numeric calculations, input/output, logging etc. without penalty.

<u>Important – use of generative AI</u>: Whilst you are encouraged to implement all the code yourself, we accept that generative AI systems (such as Copilot, which is part of the university's Microsoft Office subscription) are now widely used within software development. You are permitted to use them, without penalty, within this coursework, but according to the following strict caveats:

- You must describe how generative AI was used, including any prompts you used and the name and version of the generative AI system(s).
- You should make it clear which code in your submission was written by you, and which was produced by the generative AI system(s).
- All code should be commented, showing how lines, or related sections, of code (generated by AI or written by yourself) correspond to elements of the specification.

<u>Note that a failure to do these things will results in large penalties being applied.</u>

1. Implement a multi-layer ANN architecture from scratch

You should implement a simple feedforward multilayer architecture. The number of nodes in each layer, the number of layers, and the activation function used in each layer should be readily configurable.

Note that your ANN will be applied to a regression task in task 4, and that the number of nodes in the input and output layers will be determined by this.

You do not need to implement any classical training algorithms, e.g. backpropagation, since you will be using PSO to train the parameters (i.e. weights and biases) of the model.

Here is a list of activation functions that should be implemented, though you may also investigate other suitable functions:

- Logistic function: $\frac{1}{1+e^{-x}}$
- ReLU (rectified linear unit): $\max(0, x)$

- Hyperbolic tangent: $\tanh x$

2. Implement PSO from scratch

You should implement Particle Swarm Optimisation (PSO), specifically, the form of PSO (labelled Algorithm 39) described in Section 3.5 of the book "Essentials of Metaheuristics", which is also described in the Week 7 PSO lecture: https://cs.gmu.edu/~sean/book/metaheuristics/

Unlike the original version of PSO (which you may still find described on websites or generated by some large language models), this form of PSO uses informants. That is, each particle should be influenced by a subset of the other particles in the swarm, rather than just the swarm best. See the lecture for more information on this.

A specific requirement for this task is that the comments in the code indicate the corresponding line numbers in the pseudocode, so that it is clear how your code relates to the specification.

You are also strongly encouraged to extend these minimum requirements in some way, and this will contribute towards the marks you receive under the "Going further" section of the marking rubric. For example, you might implement different boundary handling techniques (which will be discussed in lectures) or consider other approaches you come across by reading the PSO literature – there are some pointers to this literature on Canvas.

3. Couple together your ANN and PSO implementations

PSO is an optimisation algorithm that optimises a vector of values. In this case, this vector of values should represent the set of parameters (i.e. weights and biases) of an ANN, and the goal is to find values for each of these parameters so that the ANN correctly solves a specific problem (see Task 4 below for details about this). So, you need to use the PSO code you wrote in task 2 to optimise the parameters of the ANN code you wrote in Task 1, i.e. you'll need to figure out how to couple together your PSO and ANN code. There will be some hints on how to do this in the in-person lecture in Week 7.

Here are some points to bear in mind:

- Each particle within the PSO swarm represents an entire ANN (not just a single neuron) as a fixed-length vector of floating-point values, each of which encodes the value of a particular parameter of the ANN.
- Each time a particle is evaluated in PSO, the values in its vector should be used to set the parameters of the ANN, and the ANN should then be evaluated in order to measure its accuracy on the given problem. This accuracy value then becomes the particle's fitness.
- Although there are versions of PSO that can handle variable-length vectors, you are not expected to know about these. Consequently, the architecture of the ANN (i.e. the number of layers and neurons) should be specified at the beginning of a PSO run and remain fixed.
- One advantage of using PSO, rather than backpropagation, is that you can optimise other aspects of the ANN in addition to the weights and biases. For instance, you are encouraged to also try encoding the activation functions used by the ANN within the PSO solution vector – there will be extra marks available under the "Going further" section if you do this.

4. Train the ANN to solve a specified problem

The problem domain for this work is regression, i.e. training an ANN to predict the correct numeric output for a given set of input values. You will be using this dataset:
https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength

Basically, the aim is to predict the compressive strength of a sample of concrete given 8 numeric features that describe its characteristics. That is, your ANN will take 8 numeric values as inputs, and its output will be its prediction of the compressive strength.

You can download a CSV (comma-separated values) version of the data set from the following link; each row represents one sample, with the first 8 numbers being the input features and the last number being the output value that you want your ANN to predict; there's also a header row:
https://www.kaggle.com/datasets/elikplim/concrete-compressive-strength-data-set

The dataset comprises 1030 instances/samples (i.e. rows). You should split this into a training set and a test set, using the former to train your neural network, and the latter to measure how well it generalises. A typical split is 70% of the data used for training, and 30% used for testing. You're welcome to use more advanced forms of evaluation if you like, such as cross-validation, but are not expected to do so. If you've not done any data science before, there are plenty of tutorials online that describe the basics. For example, you might find these links useful:
https://developers.google.com/machine-learning/crash-course/overfitting/dividing-datasets
https://developers.google.com/machine-learning/crash-course/overfitting/overfitting

To evaluate how well your ANN works, you should use a regression metric. The simplest is the mean absolute error (MAE), which is basically the average error between the expected outputs (i.e. the last column of the CSV file) and the outputs your trained ANN actually predicts on the test set. Again, there is plenty of information online, including:
https://towardsdatascience.com/what-are-rmse-and-mae-e405ce230383

5. Experimentally investigate the effect of ANN and PSO hyperparameters

As discussed in the lectures, both ANNs and PSO have numerous hyperparameters. For ANNs, this includes the number of layers, the number of neurons in a layer, and the activation function used in each layer. For PSO, it includes the values of the acceleration coefficients, the number of informants, the swarm size and the number of iterations.

The aim of this experimental investigation is to get some insight into how these values effect the ability of PSO to optimise an ANN than correctly solves the specified problem.

Specifically, you are asked to try and answer at least the following questions:

- What effect does the ANN architecture have on its ability to solve the problem? Architecture refers to aspects such as the number of layers, the number of neurons in a layer, and the activation function used in each layer.
- What is the best way of allocating solution evaluations? That is, the number of solution evaluations in a run of PSO is the product of the swarm size and the number of iterations. For a fixed budget of evaluations, say 500, would it be better to have a swarm of size 10 with 50 iterations, a swarm of size 50 with 10 iterations, or somewhere in between?
- What is the effect of varying the acceleration coefficients in the PSO velocity update equation? That is, look at how different balances between the coefficients affect PSO's ability to solve this particular problem. Of particular interest is the balance between the social and cognitive components, but you might also look at the other coefficients if you have time. Looking at the PSO literature might help you to find sensible values for these.

You are also encouraged to investigate the influence of other hyperparameters and design decisions, including any you introduced by extending the core PSO specification – for example, choosing between different boundary handling techniques. Again, there will be extra marks available under the "Going further" section of the marking rubric if you do this.

Here are some points to bear in mind when you do your experimental investigation:

- Pick a sensible range of values for each hyperparameter that you investigate. This could be guided by values you find in lectures, books, published papers etc. You can also use a more systematic approach, such as random search or grid search, though this is not required.
- PSO is a stochastic algorithm. That is, each time you run it, the particles start in different positions and how they move has a random element. This means that, for the same set of hyperparameter values, you will likely get different results each time you run it. To address this, it is strongly recommended that you carry out at least 10 runs for each set of hyperparameter values, and present the average and standard deviation across these 10 runs, rather than the results of individual runs.

6. Write a short report and submit both the report and your code to Canvas

Your report should:

- Be no more than **<u>6 pages</u>** in length (max of 3000 words), excluding references and appendices. You should take this into account when planning your experiments. If you have more results than you have space for, then pick the results that you think are most insightful and briefly mention which other experiments you carried out.
- Be written in Arial, Calibri, or a similar font, with a minimum **font size of 12**.

It should contain the following sections:

- **Implementation**:
  - Briefly describe what you implemented and how your code is structured; for instance, what does each function do?
  - Explain exactly how you coupled PSO to the ANN.
  - Remember to include appropriate comments in your code too (there's a separate submission link for code) – see guidance in each section above.
  - Describe how you developed your code; did you write it all yourself, or did you use generative AI? If the latter, explain exactly how and where you used it. Include details of the generative AI system and prompts used, either here or in an appendix.
  - You can also use this section to note any interesting aspects of your implementation, including any extra functionality you implemented beyond the minimum specification.
  - You can assume the reader already knows the basics of ANNs and PSO, so there's no need to introduce these in your report.

- **Results and Discussion**:
  - State which questions you attempted to answer, which hyperparameters you investigated, and how you implemented your experimental study.
  - For each hyperparameter, indicate which values you looked at and why you chose these. You are encouraged to use references to motivate these choices.
  - Show your results using tables and plots. When carrying out multiple runs to compensate for stochasticity (see above), tables should show averages (and ideally standard deviations) rather than results of individual runs. Plots are useful for illustrating trends that can be hard to spot in tables alone.
  - What did you discover? How well did your ANNs solve the problem, and how did the hyperparameter settings affect this?

  - Don’t just describe what you saw; also try to explain why this is the case, e.g. why do you think a particular hyperparameter had a particular effect?
  - You might also look at previous published results for this problem and compare against these.
  - You are encouraged to use references to the literature to support your arguments.

- **Conclusions**:
  - Provide a brief summary of what you did and what you discovered.
  - Say what else would you look at if you had more time. You can use this part to show your wider awareness of PSO and ANNs.

- **References**:
  - These should be in a standard format, e.g. Harvard style, and should be cited in the earlier sections.
  - Make sure the references exist, and don’t include references that you don’t use.

- **Appendices (optional)**:
  - You can include additional results here, if you want to. However, your marker won’t look at this section in any detail, so anything important and key to your discussion should appear in the earlier sections.
  - If you used generative AI, you can also use this additional space to include details of prompts etc. if there’s not enough space in the implementation section.

You should submit both your report (as a **pdf** file) and your code (as a **zip** file) to Canvas using the links provided. Make sure it’s clear how to run your code.

7. Indicate your individual contributions on the group signing sheet

The Coursework Group Signing Sheet is available on Canvas. Both members of your team should outline their contribution to the coursework, indicating which parts of the code, investigation and report they contributed to, and sign the sheet. It should then either be embedded in your report, or included in the code submission zip file. **No marks will be issued until this has been submitted.**

Marking and Feedback

See the table on the next page for the marking scheme. Note that the weightings are slightly different for F20BC and F21BC, and reflect a higher expectation in terms of investigation, analysis and use of the literature for the latter. The grade requirements outlined in the table are indicative only, and we’ll take into account your achievements across the assessment when deciding a mark.

Once your work is marked, we will provide you with marks and feedback. To reduce stress during this period, the school’s policy not permit us to release marks during the exam diet, so individual marks and feedback will be released just after the diet ends. However, any general feedback that is pertinent to the exam will be circulated beforehand.

You can also get informal feedback before you submit your work. In Edinburgh, lab helpers will be available during the Friday lab session for this purpose. They will also be able to help you if you get stuck, so please make use of this resource.

[marking scheme on next page…]

Criteria	Weight	A (70-100%)	B (60-69%)	C (50-59%)	D (40-49%)	E/F (<40%)
Implementation ***(i.e. code, comments, development report, description and motivation in the report)***	40% for F20BC 35% for F21BC	All code is working, meets the required specification, and is suitably commented. Development process is documented. Shows good understanding of how the code works.	As for A grade, but there are minor issues with respect to correctly implementing the requirements, documenting development, or showing understanding of how the code works.	As for B grade, but there are more significant issues.	There are major issues with implementation, documentation or showing understanding, including the omission of a development report or no attempt to explain how the code works.	Critical issues: for example, incorrect algorithms implemented, or clear lack of understanding of PSO and ANNs.
Experiments, results and discussion ***(i.e. choice and validity of experiments performed, presentation of results, understanding demonstrated in discussion and analysis)***	30%	Core questions have been answered well. Suitable hyperparameters investigated and values chosen well. Suitable results collected and presented informatively. Clear, insightful discussion that shows good understanding and includes suitable references.	Some minor issues in terms of hyperparameters investigated and their values, the experiments performed, or the presentation and discussion of results.	Significant issues in terms of hyperparameters investigated and their values, the experiments performed, or the presentation and discussion of results.	Some major issues: experiments do not make sense, have invalid results, the study is not adequately described, or the discussion is uninformative.	Some critical issues: experimental study is nonsensical or missing, inappropriate experiments, or there is no discussion.
Going further ***(i.e. extending the core specification, extra experiments, showing wider understanding, use of the literature)***	20% for F20BC 25% for F21BC	Goes well beyond the minimum requirements, and shows broad understanding of ANNs and PSO.	Goes significantly beyond the minimum requirements, or shows broad understanding of ANNs and PSO.	Some attempt to go beyond the minimum requirements and to demonstrate wider understanding.	No attempt to go beyond the minimum requirements, very little wider understanding demonstrated.	No attempt to go beyond the minimum requirements, no wider understanding demonstrated.
Report ***(i.e. structure, language, referencing etc.)***	10%	Report is well structured and correctly divided into sections; readable, with good use of language; respects page limit and formatting guidelines	Report is suitably structured and divided into sections; mostly readable with good use of language; respects page limit and formatting guidelines	Report is structured but not divided into sections; language issues that affect readability; notable formatting issues	Report is poorly structured; poor readability; significant formatting issues	Report has a nonsensical structure; very hard to read; problematic formatting