Capstone Project FAQs

Q: Where are the initial data (.npy) files?

A: The initial data points are provided in **.npy** file format. You can directly download these files either here or through the link provided in Mini-lesson 12.8.

To access the data:

1. Extract the zip file containing the **.npy** files.
2. Use `np.load()` in Python to load and work with these files.

The **.npy** files contain the starting data points for each function, which are essential for the optimisation process.

Q: What do I submit? What goes into the portal?

A: You only submit your input for each function (the numeric input string) in the following format: `x1-x2-x3-...-xn`. You do not upload the **.npy** files, code or plots to the portal. For more information on the requirements for the input format, refer to Mini-lesson 12.7.

Q: How many data points should I use? Should I use all the provided data points?

A: Yes, you should use all the data points provided in the initial **.npy** files for each function. If you see more than ten data points for any function, this is expected, and you should use all of them.

Here's a breakdown of the initial data points for each function:

- Function 1: 10 data points, 2D ((10, 2))
- Function 2: 10 data points, 2D ((10, 2))
- Function 3: 15 data points, 3D ((15, 3))
- Function 4: 30 data points, 4D ((30, 4))
- Function 5: 20 data points, 4D ((20, 4))
- Function 6: 20 data points, 5D ((20, 5))
- Function 7: 30 data points, 6D ((30, 6))
- Function 8: 40 data points, 8D ((40, 8))

These data points are important because they define the initial state of the model and provide the necessary information for the Bayesian optimisation process. In the first submission, these data points should be used as provided. In future submissions, new data points (feedback from your optimisation) should be appended to these initial data sets.

Q: Should inputs be random or based on the initial data?

A: Inputs should be based on the initial data provided. While you may use methods such as random search or grid search, it's important to begin with the initial data points (from the **.npy** files) as your foundation. From there, you can apply techniques such as Bayesian optimisation, which builds on these starting points. Your approach should use the provided data as the starting point for further exploration and optimisation, and you should be able to explain your strategy.

Q: Does my work need a positive result?

A: No. During the optimisation process, the outputs of the objective functions may be **negative by design** (for example, when penalties are applied). This is expected, and intermediate output values may be negative while you're exploring the input space.

Input values must always lie in the range **0.000000 to 0.999999** (i.e. greater than or equal to 0 and strictly less than 1). Based on these valid inputs, the objective functions may return either positive or negative outputs.

Each task should be treated as a maximisation problem. Your optimisation strategy should therefore aim to maximise the output value of each objective function.

Q. What is the primary goal of this project?

A: This capstone project puts you in the role of an ML practitioner solving real-world optimisation challenges. You'll develop practical skills in Bayesian optimisation – a technique used across industries when testing is expensive, time-consuming or resource-intensive, such as testing drug formulations with limited clinical trials, tuning manufacturing parameters with costly prototypes or optimising hyperparameters when model training takes hours. You'll learn to make informed decisions with limited data and build a portfolio piece that demonstrates advanced optimisation techniques.

You need to optimise eight synthetic black-box functions. Each function:

- Has a different input dimensionality (2D to 8D)
- Returns a single output value
- Is framed as a maximisation problem (even if the real-world analogy is minimisation, as it's transformed so that higher is better)

Your task is to find the input combination that maximises the output, using limited queries and the initial data provided in **.npy** files.

Q: What does 'black-box function' mean in this capstone project? Why use a 'black-box function'?

A: A black-box function mirrors real-world scenarios in which you can't access the internal workings of a system, such as proprietary algorithms, complex simulations or biological processes. This teaches you to optimise systems when you can only observe results, not mechanisms.

In this project, it means:

- You cannot see the internal formula or logic of the function.
- You can only query it occasionally (limited evaluations).
- You must rely on observed inputs and outputs to guide your optimisation strategy.

Q: What is provided to me?

A: You will be provided with the following items:

- Initial data in **.npy** files:
    - `initial_inputs.npy`, which contains the input combinations tried so far
    - `initial_outputs.npy`, which contains the corresponding outputs
- A description of each function and its real-world analogy

Q. How do I read the initial data files?

A: To load the data for Function 1, for example, use the code below:

```
import numpy as np

# Load initial data

X = np.load(r"C:\initial_data\function_1\initial_inputs.npy")

Y = np.load(r"C:\initial_data\function_1\initial_outputs.npy")
```

Q: What optimisation method should I use?

A: The recommended approach is Bayesian optimisation because:

- Evaluations are expensive (limited queries).
- Functions may exhibit non-linearity, noise and multiple local maxima.
- Bayesian optimisation balances exploration (trying new areas) and exploitation (refining near promising points).

Q: What are the steps to complete the project?

A: You should aim to complete the following steps.

- **For the first submission:**

- Understand the problem and function dimensionality:
  - Begin by fully understanding the problem statement and the input–output relationships of each function.
  - Check the dimensionality of each function to understand the number of variables involved and their constraints.
- Load the initial data:
  - At the **start of the project**, load the initial input and output data sets provided for each function using `np.load()`. These initial data sets serve as the starting point for Bayesian optimisation.
- Define the search space:
  - Define the search space for each function by specifying the range of values that the inputs can take.
  - For example, you might define ranges such as [0, 1] for each input variable or specify more complex boundaries depending on the problem.
- Implement Bayesian optimisation:
  - Fit a surrogate model: choose a surrogate model to approximate the objective function. The most commonly used model is a Gaussian process (GP), which provides predictions along with uncertainty estimates.
  - Define an acquisition function: select an acquisition function to guide the search for new points. Common choices include expected improvement (EI) and upper confidence bound (UCB). The acquisition function helps decide where to query next, based on the surrogate model's predictions and uncertainty.
  - Iteratively select new points: use the acquisition function to suggest new points at which to query the function. This is where the optimisation loop occurs. After each query, update the surrogate model with the new data point (the input–output pair).
- Run the optimisation for a fixed number of iterations, for example, 20 queries:
  - Set a fixed budget for the number of function evaluations, such as 20 queries.
  - Repeat the process of querying, updating the surrogate model and suggesting new points until the budget is exhausted.
- Record the best input and output found:
  - After completing the queries, record the best-performing input–output pair (the one that produced the highest output).
  - This data will be useful for benchmarking and future analysis.
- Visualise progress and summarise results:

    - Plot the progress of the optimisation process, such as how the best output improves over time.
    - Create visualisations to show how the acquisition function and the surrogate model evolve with each iteration.
    - Summarise the results in a report or graph to communicate the effectiveness of the optimisation process.
- **For the following submission:**
  - Append data from the first submission:
    - After completing the first submission, append the new feedback data (input–output pairs) to the existing data set for each function.
    - Continue accumulating data as you progress through the submissions.
  - Repeat the same steps for the subsequent submission:
    - For each following submission, follow the same steps as in the first submission:
      - ➢ Load the existing data (which now includes data from the previous submissions).
      - ➢ Update the search space (if needed) based on the newly accumulated data.
      - ➢ Implement Bayesian optimisation, using the updated data set.
    - Run the optimisation.
    - Record new best inputs/outputs, and visualise the progress.

The goal for each submission is to refine the model by adding more data and further optimising the inputs to achieve better output.

- **Example code for appending new data points:**

```
import numpy as np


# Load existing data

X = np.load(r"C:\initial_data\function_1\initial_inputs.npy")

Y = np.load(r"C:\initial_data\function_1\initial_outputs.npy")


# New data from the first submission (example inputs and
outputs)

X_w1_new_point = np.array([1.00000e-06, 9.99999e-01],
dtype=np.float64) # from input.txt
```

```
Y_w1_new_point = np.array([0.0], dtype=np.float64) # from
output.txt


# Append the new data points

X_updated = np.vstack((X, X_w1_new_point))

Y_updated = np.append(Y, Y_w1_new_point)


# Save the updated arrays

np.save(r"C:\initial_data\function_1\initial_inputs.npy",
X_updated)

np.save(r"C:\initial_data\function_1\initial_outputs.npy",
Y_updated)
```

- **Notes for the following submissions:**
  - Data storage: keep track of all previously collected data points to build a larger data set as you progress through each submission.
  - Optimisation continuity: the optimisation process should continue using all available data, ensuring that the model is refined over time.
  - After completing the capstone project every submission, please follow the submission guidelines provided at the link below: https://classroom.emeritus.org/courses/12242/files/4851390?wrap=1

Q: How should I handle minimisation problems?

A: Some real-world analogies involve minimisation (or side effects). These are transformed into maximisation by negating the output so that higher values are better. Your optimisation algorithm will always aim to maximise the transformed score.

Q: What is the primary deliverable for this activity?

A: The primary deliverable is a public GitHub repository that contains all materials related to the Bayesian black-box optimisation (BBO) capstone project. The repository should clearly present the project from start to finish and be suitable as a portfolio artefact.

As you work through the query submission process and guided reflection activities, maintain a clear and organised record of your work in a GitHub repository. You will refine and submit this repository as the final deliverable in Module 25.

Q: What code should be included in the repository?

A: The repository should include **well-documented Python code**, preferably presented in **Jupyter Notebooks**, that implements Bayesian optimisation for all eight functions. The code should be clear, reproducible and easy to follow.

Q: Are results and visualisations required?

A: Yes. The deliverables should include **plots that show optimisation progress**, such as convergence or performance over iterations, to clearly demonstrate how the optimisation performs for each function.

Q: What documentation is required?

A: The repository must include the following:

- A **README** file with an approximately **100-word non-technical summary** explaining the purpose, process and results of the project
- A completed **datasheet** describing the data used, including limitations and context
- A completed **model card** outlining model behaviour, assumptions, limitations and interpretability considerations

Q: Is a written summary of findings required?

A: Yes. The deliverables should summarise:

- The **best input and output values** found for each function
- **Why Bayesian optimisation was chosen** for this problem
- **Challenges encountered and key insights** gained during the project

This summary can be included within the notebook or the README file.

Q: How should data be handled in the repository?

A: Any **large data sets should not be stored directly on GitHub**. Instead, they should be described in the README file, which should include a link to the external data source.

Q: How should the repository be submitted?

A: The repository must be set to **public**, and the **GitHub link should be submitted** via the discussion board as the final submission.

Q: What are some tips for success?

A: Consider the following:

- Start with **Function 1** (2D) to understand the workflow.
- Gradually move to higher dimensions.
- Visualise results to check whether optimisation is improving.
- Document your reasoning and choices clearly.

Q: What type of processed data points will I receive after every submission?

A: After submitting your work to the capstone portal, you will receive the input values you submitted for each function, along with the corresponding output values, in the format shown below. Additionally, you'll be able to download files containing cumulative NumPy array data points from the first submission through the current submission.

- **This submission's input values:**

Function 1:	[0.472352, 1.055087]
Function 2:	[0.977791, 0.261960]
Function 3:	[0.116130, 0.051462, 0.314168]
Function 4:	[0.445393, 0.174795, 0.339088, 0.116347]
Function 5:	[0.797599, 0.775484, 1.047075, 1.122867]
Function 6:	[0.697910, 0.990583, 1.035622, 0.974136, 0.655285]
Function 7:	[0.090408, 0.599655, 0.299291, 0.287006, 0.599215, 0.826037]
Function 8:	[0.010000, 0.661416, 0.010000, 0.212650, 0.889104, 0.671374, 0.010000, 0.450308]

- **This submission's output values:**

Function 1:	-1.8022788966239472e-144
Function 2:	0.021320835230587712
Function 3:	-0.1482910205289665
Function 4:	-7.6005772787938355
Function 5:	6619.923795891087
Function 6:	-2.0390133791009304
Function 7:	0.33553729563585927
Function 8:	9.5673490736736

Q: How long does it take to receive the processed data point with every submission?

A: The system will process each submission at the end of each module.

Q: What if I make a late submission?

A: Late submissions will be processed within 24–48 hours. If your submission or late submission is not processed within 48 hours, kindly raise a support ticket, and our support team will assist you.

Q: What should I do if I receive the error 'Function 1 input does not match the required pattern' when submitting my queries?

A: This error occurs because the submission format does not fully meet the required specifications. Each query must follow this format: $\mathrm{x1} - \mathrm{x2} - \mathrm{x3} - \ldots - \mathrm{xn}$, where:

- Each value **must start with 0**, such as 0.123456.
- Each value must have **exactly six decimal places**.
- Values must be separated by **hyphens, with no spaces**.

For example:

- **Incorrect:** 0.498317 - 0.625531
- **Correct**: 0.498317-0.625531

Please review all your function inputs, and remove any spaces around hyphens. Ensure every number has six decimal places and starts with 0.

Once corrected, you should be able to upload successfully. If the issue persists after fixing the format, kindly raise a support ticket.