CM50268 Coursework Two

Part 2C

Bayesian Machine Learning Mini-Project — March/April 2025

CM50268 Part 2C: "Consultancy" Project Overview

Logistics

This is "Part 2C", the third (and final) part of "Coursework Two".

This portion of the coursework is worth **30 marks**.

The submission deadline for all three parts is **8pm, Tuesday April 29, 2025**.

Along with your submissions for Parts 2A and 2B, for this Part 2C you should submit a single **PDF project report** and a single **CSV data file**, both of which will be assessed (details are given below). In addition, **you should submit a Jupyter notebook** (`.ipynb` file) which contains all the code that was used to generate the results for this part of the coursework.

Overall Objectives

As part of a "consultancy" project, you've been tasked with undertaking an investigation of the reliability of a particular mechanical component on behalf of an engineering company ("the client").

The client builds and operates machinery incorporating a particular mechanical component (denoted "K") which is critical to the overall function of its hardware. This specific component is subject to significant wear and tear while operating, and the degradation for any given component over time is tracked by an "integrity" measurement, on a scale of 0 to 100 (essentially, a percentage).

For your investigation, the client is supplying a dataset comprising time-series observations from 75 "K" components, with integrity measured irregularly (and noisily) over a period of up to 45 days. The first 50 components in the dataset are mainly well populated with measurements, but the remaining 25 cases were acquired later and as such have fewer associated data points (see the below figure for examples). These latter 25 cases will be the subject of a "blind test" (of prediction).

An important client objective is to predict the likely integrity of a component at a future time, and this task will inevitably be more difficult for those components which have only a few days' worth of data. After dabbling unsuccessfully with other machine learning methods in the past, the client now believes that Bayesian modelling may be the answer to their problems in this low-data regime. In particular, their understanding is that a combination of *hierarchical* Bayesian methods with the latest stochastic approximation techniques should offer the best results. To this end, they have engaged you to investigate …

The Supplied Data

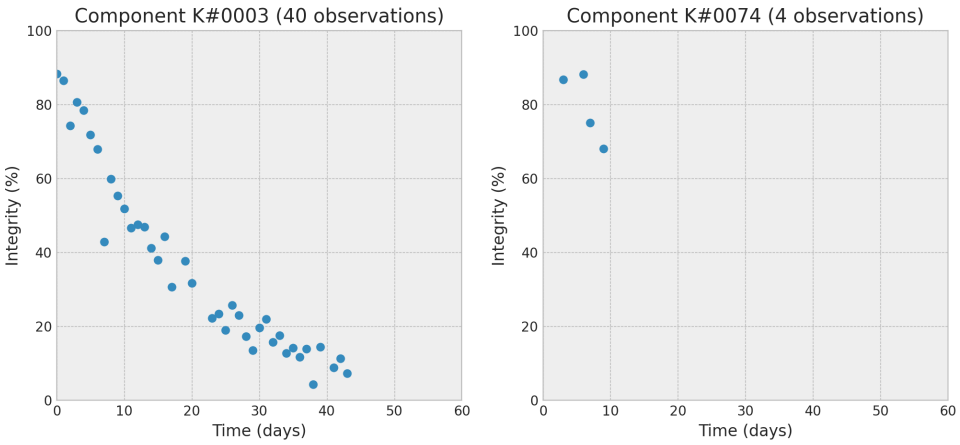
For the purposes of your investigation, the client is supplying a dataset in CSV (comma-separated value) format, in the text file "`component-data-2c.csv`".

That file comprises a header line, with a further 879 lines of data, one line per measurement, spread across the 75 components (which are grouped consecutively in the file).

Each row of data comprises:

- `ID`: a textual identifier for the component (purely for convenience of reference in graphs *etc*),
- `index`: a unique index for each component (to utilise in code), in $[0, 74]$,
- `X1` to `X5`: five real-valued characteristics of each individual component (constant over time for each component),
- `day`: the time point at which a measurement was taken, in $[0, 45]$,
- `integrity`: the measurement itself, in $[0, 100]$.

Data for two of the components is illustrated below. That on the left has a generous number of observations. That on the right is from the "test" examples, for which the data is sparse.



Client Brief

The client has three primary objectives:

1. to accurately model and characterise degradation rate (decrease in integrity) based on historical data,
2. to be able to make the best future "prediction" of component integrity, especially in the case where a given component has little associated data,
3. to better understand potential causes of degradation, based on the additional component characteristics (`X1` to `X5`).

Baseline Model

Based on experience and prior analysis, the client suggests that the integrity measurements y_i (on a scale 0 to 100) for component i (`index` 0–74) measured over time t (in days) may be initially modelled as a noisy realisation of an exponentially decaying function f_i:

$$\begin{aligned} y_i(t) &= f_i(t) + \epsilon, \text{with,} \\ f_i(t) &= u_i \exp\{-v_i t/100\}, \end{aligned} \tag{1}$$

where u_i (an "intercept" term, the value of f_i when $t = 0$) and v_i (a "decay rate" term) are specific to component i. The additive factor ϵ may be considered a measurement noise or error term, and is generally accepted to be approximately Gaussian with some unknown standard deviation σ_y. The fixed time divisor of 100 is simply a scaling convenience (it makes v_i more interpretable), and should be retained in any analysis.

In terms of likely parameter values, the client advises that there will be some variation of u_i and v_i across components, with u_i likely to be in the range 80 to 100, and v_i perhaps between 1 and 10.

Enhanced Model

Based on engineering considerations, the client hypothesises that degradation may (or may not) be influenced by certain physical component characteristics, detailed measurements of which are included in the dataset, as fields `X1` to `X5`. These diagnostics inform an enhanced functional model:

$$f_i(t) = u_i \exp\left\{-\left(v_i + \sum_{j=1}^{5} w_j x_{ji}\right) t/100\right\}, \tag{2}$$

where x_{1i} is the measurement of `X1` for component i as tabulated in the dataset, and similarly for `X2` to `X5`.

The "weightings" w_j associated with the diagnostics are common across all components, although the client cannot advise as to the likely scales of those weights. It is of particular interest to the client to know which of the diagnostics is most relevant, in terms of most accurately predicting degradation, as this may inform an "intervention". The client notes that as a result of incorporating the w_j parameters in the model, the expected magnitude of v_i will be reduced, compared to (1).

Blind Test

At the point the data was captured, the last 25 components in the dataset (`index` values 50–74) only had a few measurements, all taken before $t = 10$. For those 25 components, as a "blind test", the client would like you to submit a file containing your predictions of the probability of the integrity of that component being below 30% at time $t = 30$. (The 30% integrity level is that at which the component would normally be refurbished or replaced.)

The client will continue to gather data around those components, and will be able to subsequently assess your predictions.

Additional Queries from the Client Team

As part of your investigation, and in your explanatory report, the client has asked that you address two further questions from their engineering team regarding the choice of Bayesian and/or other modelling methods.

First, one member of the client team is a little sceptical about your proposed approach. He says:

> "The Bayesian hierarchical approach sounds quite complex. Can you not simply build a black-box model that maps from some of our usable data to directly output a prediction? *e.g.* just train a neural network?"

Second, one member of the client's engineering team is not familiar with Bayesian modelling, but has had some previous exposure to machine learning in other contexts. She asks:

> "I've seen some work in the past that made use of a popular model called the *Support Vector Machine*. It was said to be very accurate, and if I remember correctly, was deployed in many applications at the time. Can you explain why you do not use this model in your analysis, and instead use Bayesian techniques?"

Tasks, Outputs & Mark Allocation

Deliverables

Details concerning content are given below, but in summary you should submit:

- a short project report (as a guideline: no more than 2,000 words), featuring supporting graphs and figures (which do not count against the word limit), in PDF format,
- a CSV file ("`predictions.csv`") containing your probability predictions for the "blind test" (see Page 7 for more detailed instructions regarding this file),
- a Jupyter notebook file that contains all the code developed to produce the above two deliverables.

Project Report

Predictive Modelling (10 marks)

Client objectives 1 and 2.

For both the baseline (1) and enhanced (2) degradation equations, briefly explain how you incorporated those equations within a Bayesian model, and illustrate how the model performs.

For the benefit of the client, include the following graphs *in both cases*:

- the output of the `arviz` diagnostic function `plot_trace` for the posterior samples of the main variables in your model(s):
  - u_i and v_i (use `compact=True`)
  - any higher-level variables in the hierarchy that u_i and v_i are conditioned on
  - the observation noise standard deviation σ_y
  - in the case of equation (2), *exclude* the enhanced model weights w_j and associated hyperparameter(s) for now (these come later: see below)
- illustrations of the model fit for four individual components:
  - one component with plentiful data (from the first 50),
  - three components with limited data (chosen from the latter 25)
- when illustrating the model fits above, plot the following:
  - the data as shown in the earlier figure (using the same axis scales)
  - the posterior predictive mean function, up to $t = 60$
  - error-bars capturing uncertainty of the underlying function estimate, which in this case should be 1.96 times the posterior predictive standard deviation *of the function* (*i.e.* to include 95% of the distribution)
  - wider error-bars capturing overall uncertainty, of width 1.96 times the posterior predictive standard deviation *of the function with data noise*
- take care to render the two different error-bars so they can be easily viewed (perhaps make use of the `alpha` transparency parameter)

Potential Causes of Degradation (5 marks)

Client objective 3.

For the model incorporating the enhanced degradation equation, show the `plot_trace` output for the weight parameters w_j and any associated hyper-parameter(s). (These plots were specifically omitted above.)

In your report, evaluate the significance of your results and explain (as if to the client) the implications in terms of what component characteristics are relevant for the prediction of degradation.

Overall Report Quality (4 marks)

Up to four marks will be awarded for overall quality of the report, taking into account structure of material, clarity of text, formatting, and utility of graphs and illustrations.

Address Engineering Team Queries

A Black-box Model (4 Marks)

For illustration, train a "black-box-style" function model that maps some input features from $t \in [0, 10]$ to output a prediction of the integrity value at $t = 30$. This can be anything you like (from linear regression to deep neural network), but should obviously be non-Bayesian.

You can devise your own training set with input features here, but a satisfactory approach would be to extract a subset of components that have populated data up to $t = 30$, then use some derived statistics of the available data up to $t = 10$ to predict the value at (or near) $t = 30$. Those statistics might include the average value of integrity for $t \in [0, 10]$, the average gradient in that range, the average log value *etc*. To these statistics, you will likely want to append the component characteristics `X1` to `X5`.

In your report, briefly describe your modelling approach and include some specimen prediction graphs (perhaps contrasting with the equivalent Bayesian methods).

Don't spend too much time on this! The results are purely illustrative and need not be assessed for accuracy. Just build a sensible model and show that it works, at least in a basic practical sense, and offer a minimally viable comparison with your Bayesian approach.

The Support Vector Machine (2 Marks)

Respond to the team member's question by outlining your reasons for using a hierarchical Bayesian modelling approach in this particular application, and explain why a support vector machine would be inappropriate in general for many real-world tasks.

Blind Test Predictions (5 marks)

For your preferred model, which should be the one based on Equation (2), compute the probability that integrity is 30% or under at $t = 30$ for the last 25 components in the data set (`index` 50–74).

Your submission should include a *text* file "`predictions.csv`" containing your 25 predictions. Use the below comma-separated value format, and take care to use the correct column labels.

```
ID,index,probability
K#0050,50,0.007294
K#0051,51,0.104290
K#0052,52,0.082604
... (etc)
K#0074,74,0.421453
```

Hints & Suggestions

Getting Started

You should apply Bayesian hierarchical models to address the client brief (of course, you should always give the client what they ask for). Implement your models using the `numpyro` library, specifically the Hamiltonian Monte Carlo (HMC) sampler with the "No U-Turn" (NUTS) kernel. To get started, you can re-purpose some of the code from the Coursework Part 2B as a framework.

In addition, you may find the example that featured in Lecture 10 — modelling *pulmonary fibrosis* — to be helpful. The lecture notebook is available on Moodle and is based on an original tutorial you can find here: https://num.pyro.ai/en/stable/tutorials/bayesian_hierarchical_linear_regression.html)

The Lecture 10 code features a purely linear model, which may be worth implementing as a starting point, even though the underlying function model is not the correct one. Then when the linear model is running, it may be modified and upgraded as appropriate.

Hints

Running HMC Effectively

- For final results to go in the report, run your `MCMC` sampler for at least 2,000 warm-up plus 2,000 samples (this may take up to 30 seconds). You can of course cut this down while experimenting and developing your model.
- Your results will only be reliable if your sampler has run correctly, so use `print_summary` to check that:
  - `Number of divergences` is zero
  - `r_hat` is always 1.00
  - `n_eff` is not alarmingly low (less than one-tenth of the `num_samples`)
- If the sampler is proving problematic, you may have mis-specified your model or chosen very poor priors. However, see also the "Advanced Suggestion" below.
- It is a good idea to run several parallel MCMC chains. *e.g.* pass `num_chains=4` to the `MCMC` constructor. To make this work efficiently (on a modern multiple-core CPU), you also need to add `numpyro.set_host_device_count(4)` immediately after your `import numpyro` statement. For some reason, running parallel multiple chains can actually run faster than a single chain.

Miscellaneous

- **Important:** remember that the posterior mean function (as used for prediction) is the average of the functions evaluated separately for all samples of u_i and v_i *etc.* It is *not* the function evaluated once for the average of u_i and v_i.
- Similarly for calculating the posterior standard deviation of the function.
- To get the wider error-bars, you need to add the posterior function *variance* to the noise *variance*, then take the square root of that sum.
- For the enhanced model of Equation (2), existing Bayesian linear regression principles can be applied to incorporate the diagnostic weightings w_j.
- To limit the graphs produced by `plot_trace` from `arviz`, you can pass the `var_names` argument to specify the variables that you want.
- For the blind test, to predict the probability that integrity is 30% or below given your model, there are two ways you might do this:
  - use your existing computation of posterior mean and standard deviation evaluated at $t = 30$, then appropriately apply the `normal.cdf` function to work out the amount of probability mass that falls on or below the 30% threshold
  - count up the number of sampled functions f that predict $y \leq 30$ at $t = 30$ and divide by `num_samples` — simple, yet correct!
- In either case, your estimate of the noise variance is not needed for the blind test.

Advanced Suggestion (That You May Need)

If you have problems running your HMC sampler (*e.g.* you have divergences) and you are confident that your model is structurally correct and your priors are sensible, then the only cure may be *re-parameterisation*. This is a relatively complex subject which we can't realistically cover in the lectures. For some informative discussion on the topic, see:
https://mc-stan.org/docs/2_19/stan-users-guide/reparameterization-section.html

For the purposes of this project, the following simple modification may help. Consider you are sampling a variable `z` which is then utilised further down the hierarchy, and `z` is sampled from a Gaussian (`Normal`) with adjustable location and scale:

```
z = numpyro.sample("z", dist.Normal(mu, sigma))
```

This can be re-parameterised in terms of a standard `Normal`, without changing the model, as:

```
a = numpyro.sample("a", dist.Normal())
z = mu + a*sigma
```

or as:

```
a = numpyro.sample("a", dist.Normal())
z = numpyro.deterministic("z", mu + a*sigma)
```

In practice, the latter is likely to be the preferred option, as declaring `z` to be `numpyro.deterministic` means that it will be returned by `get_samples()` (with key `z`). Otherwise, you would have to calculate `z` separately for predictive purposes.

If you are making use of `numpyro.render_model` to visualise your model, then you will notice that `deterministic` sites are rendered as dashed nodes.