The Australian National University
School of Computing
Conveners: Thang Bui & Jing Jiang

Semester 1, 2025
Assignment 1
Weight: 20%

COMP4670/8600: Statistical Machine Learning

Release Date. 28th February 2024.
Due Date. 28th March 2024 at 23:59 AEDT.
Maximum credit. 100 Marks.

Gaussian Linear Dynamical Systems, Bayesian Bandits, and Gaussian Processes

Many different types of machine learning models fall under the label of "Bayesian models" or probabilistic models. You might have noticed that in the first half of the course, we introduce a non-Bayesian treatment of a machine learning problem which is then followed up by its Bayesian counterpart, *e.g.*, from logistic regression to Bayesian logistic regression; or kernel regression to Gaussian process regression. Bayesian modelling and inference are often justified because of their ability to incorporate "prior beliefs" into models, to update the beliefs based on observed data, and to quantify uncertainties over unknowns (e.g., you have a probability over outputs). So far in the course, we have looked at probabilistic models for regression and classification. In this assignment, we'll be exploring how to use the same machinery for linear Gaussian dynamical systems and multi-armed bandits. We will also look at a Gaussian process model with a non-Gaussian likelihood.

For submission Name your answer file as `uIDNUMBER.pdf`, add the file together with *two* python files `glds.py` and `bandits.py` into a zip file, name it as `uIDNUMBER.zip`, and then submit the zip file on Wattle. Ensure that the coding solutions are in the designated files (as specified). Do not include additional files, such as your locally installed packages or additional figures.

Coding The code has been tested with standard packages such as `numpy` and `matplotlib`. You do not need additional packages for this assignment.

Collaboration You are free to discuss the material in the assignment with your classmates. However, every answer and code submitted must be written *by yourself without assistance*. You should be able to explain your answers if requested.

Late submission policy We allow a 5-minute grace period after midnight. Assignment submissions that are late from 5 minutes to 24 hours attract a 5% penalty (of possible marks available). Submissions late by more than 24 hours without an approved extension will get zero. Please submit early to avoid potential connection issues.

Extension requests will be processed via the Extension online form available on the CSS website.

Regrade requests will be processed via a Microsoft Form. This will be released in due course.

Other notes:

- When writing proofs, use the equation numbers when referring the equations in the assignment, *i.e.*, "Through Equation (3.1) we can show ...".
- You are not required to complete the assignment linearly, you may want to solve whichever questions you can regardless of order of appearance.
- Unless stated otherwise, code is only graded on the correctness of functions implemented, not on performance or code quality. Our main requirement on performance/code quality is that we can run your code in a reasonable time when marking it.
- You do not need to attempt the optional questions.

Section 0: Bayes' rule warm-up (0 Points)

In machine learning, we often consider Bayes' rule in the context of a training dataset $\mathcal{D} = \{\mathbf{X}, \boldsymbol{y}\}$ and model parameters $\theta \in \Theta$. In particular, we use it to compute the *inverse* or posterior probability distribution (or density) $p(\theta|\mathcal{D})$ based on a likelihood $p(\mathcal{D}|\theta)$, and a prior distribution $\pi(\theta)$,

$$\underbrace{p(\theta|\boldsymbol{y}, \mathbf{X})}_{\text{“posterior”}} = \frac{\overbrace{p(\boldsymbol{y}|\theta, \mathbf{X})}^{\text{“likelihood”}} \cdot \overbrace{p(\theta)}^{\text{“prior”}}}{\underbrace{p(\boldsymbol{y}|\mathbf{X})}_{\text{“marginal likelihood”}}}. \tag{0.1}$$

For example, in Bayesian linear regression, we use a Gaussian likelihood and a Gaussian prior as follows,

$$p(\theta) = \mathcal{N}(\theta; \mu_0, \Sigma_0) \tag{0.2}$$

$$p(\boldsymbol{y}|\theta, \mathbf{X}) = \mathcal{N}(\boldsymbol{y}; \mathbf{X}\theta, \sigma_y^2 \mathrm{I}), \tag{0.3}$$

where $\mathbf{X} \in \mathbb{R}^{N \times D}$, $\boldsymbol{y} \in \mathbb{R}^N$, $\theta \in \mathbb{R}^D$, σ_y^2 is the observation noise, and μ_0 and Σ_0 are the prior mean and covariance, respectively. *Derive the posterior* $p(\theta|\boldsymbol{y}, \mathbf{X})$ *and the marginal likelihood* $p(\boldsymbol{y}|\mathbf{X})$. Since this is a warm-up exercise and the answers are already discussed in the lectures and readily available in standard textbooks, you do *not* need to include your answer in the submission.

Section 1: Gaussian linear dynamical systems (55 Points)

In many real-world engineering settings, the states of physical objects over time such as location or speed, often governed by complex and unknown dynamics, are only observed through noisy sensors. It is thus important to *denoise* and *estimate* the true states. One way to do this is to write down a model of the true state, how it changes over time, and how it is linked to the observed sensor measurements, and subsequently perform "inference" to estimate the state given the observations. We will study a model class with linear dynamics and Gaussian noise as follows,

$$\boldsymbol{s}_t = \mathbf{A}\boldsymbol{s}_{t-1} + \boldsymbol{w}_t, \tag{1.1}$$

$$\boldsymbol{y}_t = \mathbf{C}\boldsymbol{s}_t + \boldsymbol{v}_t, \tag{1.2}$$

where $s_t \in \mathbb{R}^{D_s}$ is the state at time t, $\boldsymbol{w}_t \in \mathbb{R}^{D_s}$ is the noise in the dynamics, $y_t \in \mathbb{R}^{D_y}$ is the observation at time t, $\boldsymbol{v}_t$ is the observation noise. We assume the noises are Gaussian-distributed, $\boldsymbol{w}_t \sim \mathcal{N}(0; \mathbf{Q})$ and $\boldsymbol{v}_t \sim \mathcal{N}(0; \mathbf{R})$, and $\mathbf{A} \in \mathbb{R}^{D_s \times D_s}, \mathbf{C} \in \mathbb{R}^{D_y \times D_s}, \mathbf{Q} \in \mathbb{R}^{D_s \times D_s}, \mathbf{R} \in \mathbb{R}^{D_y \times D_y}$ are fixed over time. The initial state $\boldsymbol{s}_0$ is assumed to be drawn from $\mathcal{N}(\mu_0, \Sigma_0)$.

It is perhaps more pedagogical to see how to sequentially generate data from such a model using the following steps/densities,

$$\begin{aligned}
p(\boldsymbol{s}_0) &= \mathcal{N}(\boldsymbol{s}_0; \mu_0, \Sigma_0) \\
p(\boldsymbol{s}_1|\boldsymbol{s}_0) &= \mathcal{N}(\boldsymbol{s}_1; \mathbf{A}\boldsymbol{s}_0, \mathbf{Q}) \\
p(\boldsymbol{y}_1|\boldsymbol{s}_1) &= \mathcal{N}(\boldsymbol{y}_1; \mathbf{C}\boldsymbol{s}_1, \mathbf{R}) \\
&\vdots \\
p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}) &= \mathcal{N}(\boldsymbol{s}_t; \mathbf{A}\boldsymbol{s}_{t-1}, \mathbf{Q}) \\
p(\boldsymbol{y}_t|\boldsymbol{s}_t) &= \mathcal{N}(\boldsymbol{y}_t; \mathbf{C}\boldsymbol{s}_t, \mathbf{R})
\end{aligned}$$

We illustrate two trajectories from such a system below.

Assume that we know the dynamics and noise configurations, $\mathbf{A}, \mathbf{C}, \mathbf{Q}, \mathbf{R}, \mu_0, \Sigma_0$ and have observed data up to T, $\boldsymbol{y}_{1:T}$, the typical task is to estimate the hidden state $\boldsymbol{s}$ as data are collected, e.g., estimate the location and speed of the robot in the example above. We will attempt to do this step by step.

Question 1.1: Find $p(\boldsymbol{s}_1)$ and $p(\boldsymbol{s}_1|\boldsymbol{y}_1)$ (5 Points)

Question 1.2: Find $p(\boldsymbol{s}_t|\boldsymbol{y}_{1:t-1})$ and $p(\boldsymbol{s}_t|\boldsymbol{y}_{1:t})$ (15 Points)

Given $p(\boldsymbol{s}_{t-1}|\boldsymbol{y}_{1:t-1}) = \mathcal{N}(\boldsymbol{s}_{t-1}; \mu_{t-1}, \Sigma_{t-1})$, find $p(\boldsymbol{s}_t|\boldsymbol{y}_{1:t-1})$ and $p(\boldsymbol{s}_t|\boldsymbol{y}_{1:t})$ in terms of $\mu_{t-1}, \Sigma_{t-1}, \mathbf{A}, \mathbf{C}, \mathbf{Q}$, and $\mathbf{R}$. Show the steps, not just the final result.

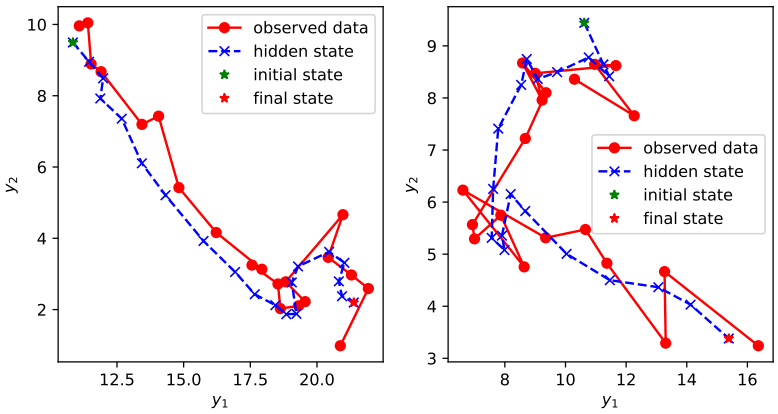

Figure 1: Here we have a robot moving in a two-dimensional environment and we get to observe only the noisy version of the robot's locations. This moving scenario can be modelled by a simple Gaussian linear dynamical system with $\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$, $\mathbf{Q} = 0.1 * \text{np.eye}(4)$, $\mathbf{R} = 0.5 * \text{np.eye}(2)$, $\mu_0 = \begin{bmatrix} 10 & 10 & 1 & -1 \end{bmatrix}^\intercal$, and $\Sigma_0 = 0.5 * \text{np.eye}(4)$. We have two example trajectories from the robot. In each plot, the observed data $\boldsymbol{y}_{1:20}$, and the first two dimensions of the true hidden states $\boldsymbol{s}_{0:20}$ are shown. The task is to estimate the blue paths given the red paths.

Question 1.3: Implement the results in Question 1.2 (15 Points)

The steps to obtain the above conditional distributions are often called the *predict* step and *update* step in Kalman filtering, respectively. This is because the former is the predictive distribution over $\boldsymbol{s}_t$ given the observations up to $t-1$, and the latter is the updated belief about $\boldsymbol{s}_t$ once an extra observation $\boldsymbol{y}_t$ is added to our data. This is also an instance of the *forward* message passing step in a tree or chain-structured graphical models that you will learn in the later part of this course. Now fill in the code in `./code/glds.py` using the results in Question 1.2. You can run `./code/glds_example.py` to visualise the results on the example in Figure 1 and double-check your implementation.

Question 1.4: Find $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})$ (10 Points)

Write the results in terms of $\mu_{t-1}, \Sigma_{t-1}, \mathbf{A}, \mathbf{C}, \mathbf{Q}$, and $\mathbf{R}$. Now fill in the code in `./code/glds.py`.

Question 1.5: Find the log marginal likelihood $p(\boldsymbol{y}_{1:T})$ (10 Points)

Given the result in the last question, discuss how to efficiently compute the log marginal likelihood $\log p(\boldsymbol{y}_{1:t})$ and provide an implementation in `./code/glds.py`. In practice, this quantity can be used for model selection and hyperparameter optimisation.

Question 1.6: [Optional] How do we deal with a non-linear dynamics between $\boldsymbol{s}_{t-1}$ and $\boldsymbol{s}_t$ and/or a non-linear mapping between $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$? (0 Points)

Section 2: Bayesian bandits (25 Points)

A central question in our daily lives is how we make decisions *sequentially* in the face of *uncertainty*. For example, a doctor needs to decide what medicine to prescribe to a patient, or a social network might want to decide on a version of its website to show to a user to increase interaction, or a movie streaming service needs to decide what movie to suggest to a user. To be concrete, consider a simplified setting with two potential actions and binary outcomes. Imagine a medical authority is trialling two drugs, A and B, and would like to understand their efficacy in treating a disease while aiming to treat as many patients as possible. When a new patient commits to the trial, a drug is chosen to give to the patient. The outcome of the treatment (treated or not treated) is recorded. The testing is then repeated for the next new patient. Now let's pause here and ask: why not randomly assign a drug to a patient, measure the successful treatment rate at the end of the trial and conclude? For example, we allocate 2000 patients at complete random to A and B or roughly 1000 each. Assume the unknown drug efficacies for A and B are 0.5 and 0.6, respectively, then at the end of the trial, roughly 50% of patients in pool A or 500 patients will be treated, and roughly 60% of pool B or 600 patients will be treated. We can then eyeball the success rates and conclude that B is better. However, this process results in 900 unsuccessful treatments. Can we leverage the data collected *during* the trial to reduce this number and adaptively adjust the patient-drug allocation? That's where bandit algorithms come in.

This class of problems is often called *multi-armed bandits* since it resembles the setting where a person gambles at a row of slot machines or pokies with potentially different probabilities of winning. The gambler's task is to decide which machine to play next based on the previous plays, to efficiently identify the machine with the highest reward *or* maximise average long-term rewards. There are three key elements in these problems: a belief about the efficacy of each arm or action (e.g., drug A/B is xx/yy% effective), a chosen action (e.g., drug A is prescribed instead of B), and the outcome given the chosen action (e.g., the patient is treated or not). We assume that the efficacy of each drug is the same for different patients with the same disease and that the arms are independent. In this exercise, we will study *how to update the belief given the chosen actions and outcomes*. We will provide code implementing three strategies to *choose* an action based on the current belief, random, ϵ-greedy and Thompson sampling, but you do not need to know their details to complete this exercise.

First, let's revisit the drug trial example and let θ_a and θ_b, both $\in [0, 1]$, be the unknown efficacy of the drugs, respectively. The outcome for each patient allocated to each drug pool can be modelled as a Bernoulli trial as follows,

$$p(y_a|\theta_a) = \theta_a^{y_a}(1-\theta_a)^{1-y_a} \tag{2.1}$$

$$p(y_b|\theta_b) = \theta_b^{y_b}(1-\theta_b)^{1-y_b}, \tag{2.2}$$

where $y_a = 1$ means treated, $y_a = 0$ means not treated, and similarly for y_b. Assume that we have allocated $\mathbf{Y}_a$ and $\mathbf{Y}_b$ patients to Groups A and B, respectively, and of which, the treatment for S_a and S_b patients were successful. We place a Beta prior over θ_a and θ_b, $p(\theta_a) = \text{Beta}(\theta_a; 1, 1)$ and $p(\theta_b) = \text{Beta}(\theta_b; 1, 1)$. Note the pdf of $\text{Beta}(\theta; \alpha, \beta)$ is $p(\theta) = \text{B}(\alpha, \beta)^{-1}\theta^{\alpha-1}(1-\theta)^{\beta-1}$, where B is the beta function.

Question 2.1: Find $p(\theta_a|\mathbf{Y}_a)$ and $p(\theta_b|\mathbf{Y}_b)$ (5 Points)

Question 2.2: Implement the results above and discuss the simulation results. (10 Points)
Fill in the code template in `./code/bandits.py` and run the simulation in `./code/bandits_example.py`. Comment on the simulation outputs.

We will now consider a two-arm setting with noisy positive outcomes (e.g., basket size in an online order). The outcome for each arm can be modelled using a log-Normal distribution with an unknown positive mean θ,

$$p(r_a|\theta_a) = \text{logNormal}(r_a; \log\theta_a - \sigma_y^2/2, \sigma_y^2) = \frac{1}{r_a\sigma_y\sqrt{2\pi}}\exp\left(-\frac{(\log r_a - \log\theta_a + \sigma_y^2/2)^2}{2\sigma_y^2}\right) \tag{2.3}$$

$$p(r_b|\theta_b) = \text{logNormal}(r_b; \log\theta_b - \sigma_y^2/2, \sigma_y^2) = \frac{1}{r_b\sigma_y\sqrt{2\pi}}\exp\left(-\frac{(\log r_b - \log\theta_b + \sigma_y^2/2)^2}{2\sigma_y^2}\right) \tag{2.4}$$

We place a log-Normal prior over θ,

$$p(\theta_a) = \text{logNormal}(\theta_a; \mu_o, \sigma_o^2) = \frac{1}{\theta_a\sigma_o\sqrt{2\pi}}\exp\left(-\frac{(\log\theta_a - \mu_o)^2}{2\sigma_o^2}\right) \tag{2.5}$$

$$p(\theta_b) = \text{logNormal}(\theta_b; \mu_o, \sigma_o^2) = \frac{1}{\theta_b\sigma_o\sqrt{2\pi}}\exp\left(-\frac{(\log\theta_b - \mu_o)^2}{2\sigma_o^2}\right) \tag{2.6}$$

Assume we have collected $\mathcal{D}_a = \{r_{a,i}\}_{i=1}^{N_a}$ and $\mathcal{D}_b = \{r_{b,i}\}_{i=1}^{N_b}$ for the two arms.

Question 2.3: Find $p(\theta_a|\mathcal{D}_a)$ and $p(\theta_b|\mathcal{D}_b)$ (10 Points)

Question 2.4: [Optional] Implementation (0 Points)
Using the scaffold code that we provided for Beta-Bernoulli bandits, implement the log-Normal–log-Normal case above.

Question 2.5: [Optional] Best-arm probability (0 Points)

Compute the probability of each arm being the best arm in the examples above. Can that be done analytically? If the goal is to identify the best arm quickly, how should we choose an arm to play next?

Section 3: Gaussian process with negative binomial likelihood (20 Points)

Assume that we have observed data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, where $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{0, 1, 2, \ldots\}$. We can model such data using a Gaussian process (GP) model with a negative binomial likelihood. That is, we have a GP prior over the underlying function, $f \sim \mathcal{GP}(0, k(\cdot, \cdot))$. Denoting $\mathbf{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)]$. By the definition of GPs, we have $\mathbf{f} \sim \mathcal{N}(0, \mathbf{K})$ where $\mathbf{K}$ is the covariance evaluated at the training points. We assume $\mathbf{K}$ to be invertible. We consider the following negative binomial likelihood:

$$p(y_n | f(\boldsymbol{x}_n)) = \binom{y_n + r - 1}{y_n} \pi_n^{y_n} (1 - \pi_n)^r, \quad i = 1, \ldots, n \tag{3.1}$$

where $\pi_n = (1 + \exp(-f(\boldsymbol{x}_n)))^{-1}$ and $r > 0$ is a fixed dispersion parameter.

Question 3.1: Find the log-posterior and gradient (10 Points)

Compute $\log p(\mathbf{f}|\mathcal{D})$ up to an additive constant and derive its gradient with respect to $\mathbf{f}$.

Question 3.2: Find the Hessian (10 Points)

Compute the Hessian matrix of the log-posterior and prove that it is negative definite.

Question 3.3: [Optional] Laplace approximation (0 Points)

Implement a procedure to obtain the Laplace approximation based on the above results.

Question 3.4: [Optional] Gaussian variational approximation (0 Points)

We now wish to use Gaussian variational inference instead of Laplace. Write down the variational objective. Can we obtain this and its gradients analytically? What additional approximations do we need to make? And finally, discuss the computational cost required for both Laplace and variational approximations.