Lab_3

February 16, 2025

1 Lab 3: Kalman Filtering and Online Bayesian Inference

1.0.1 Lab Date: Wednesday, February 12

1.0.2 Part I Due: Wednesday, February 19

1.1 Instructions

Work with your lab group to complete the following notebook. Your work on Part I will be reviewed by your peers in lab next week (Wednesday, February 19th). This is a multi-part lab. If you finish lab 3, you may go to Lab 4 to complete parts II and III. Lab 4 (Parts II and III) will be due in two weeks (Wednesday, February 26).

In the next two labs, you will:

1. Practice with the normal-normal model, [Part I]
2. *See how easily (some) conjugate models extend to thorny multi-dimensional problems, and, accordingly, why they are so popular as building blocks for large problems, [Part I and II]*
3. *See how the self-consistency of Bayesian inference allows for straightforward on-line inference (inference as the data is collected). [Parts II and III]*

Together, these demonstrate two of the common selling points for Bayesian methods: (1) they extend easily and consistently to multiple dimensions without requiring the user to make arbitrary choices to reduce dimension (e.g. picking a test statistic), (2) they are very natural to work with when tracking sequential data. It also demonstrates a subtler computational message. The larger our problems, the more useful analytic results are. As you read ahead, try to imagine what the procedure would look like if we replaced the simple algebra that follows from conjugacy with sampling, or, god forbid, numerical integration.

These learning goals will be presented to you through the Kalman filter, a fundamental tool for data assimilation in dynamical systems. The Kalman filter is an iterated application of a Gaussian likelihood, a Gaussian prior on the mean, and a linear update model to represent a dynamical system with a measurement protocol. Until this year, this approach was state-of-the-art for very large online prediction problems such as weather forecasting (see GraphCast, 2023 and GenCast, Nature, 2025).

The Kalman filter involves a lot of algebra. We've scaffolded that algebra, and the associated inferential code, for you. Focus on the statistical story represented in the algebra.

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials. You may use the same procedure you used last week to work with that folder.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

```
[1]: # load whatever packages you prefer here. We've added a reference list here
     import numpy as np
     import matplotlib.pyplot as plt
```

1.2 [PART I] Conjugate Gaussian Models

So far we've looked at one family of conjugate models, the (Beta, Binomial). In this lab we'll be working with the next natural conjugate family, the (Normal, Normal) family. For reference, see BDA Chapter 2.5, 3.2, and 3.3.

1.2.1 The Model:

Consider a joint model of the form: - **Prior:** $M \sim \mathcal{N}(\hat{m}, C_m)$ where $\hat{m} = \mathbb{E}[M]$ and $C_m = \text{Cov}[M]$ - **Likelihood:** $Y|M = m \sim \mathcal{N}(m, C_y)$ where C_y is the covariance in X given m.

This is a conditionally Gaussian model for Y, conditioned on an unknown mean, M. In this context, C_y is often interpreted as the covariance of some noise. Combining a Gaussian prior with a conditionally Gaussian likelihood produces an elegant conjugate family. This family is very convenient for large problems, since it allows analytic posterior inference in arbitrary dimension. The inference proceeds via a sequence of linear algebraic updates, so is easy to apply as a step in larger inferential problems.

1.2.2 Solving for the Posterior (in 1 Dimension)

Suppose that $M \in \mathbb{R}^1$ and $X \in \mathbb{R}^1$: - **Prior:** $M \sim \mathcal{N}(\hat{m}, \sigma_m^2)$ - **Likelihood:** $Y|M = m \sim \mathcal{N}(m, \sigma_y^2)$

Q1.1: In the space below, write out the explicit form for the joint distribution at (m, y) by evaluating the product of the prior and likelihood. As usual, drop any normalizing constants (the evidence is implied). Based on your answer, what class of distributions must the posterior be a member of?

Fill in this cell with your answer.

Q1.2: Complete the square inside the exponential to solve for the mean and variance in $M|Y = y$. Simplify your answers in terms of the signal-to-noise ratio (SNR), σ_m^2/σ_y^2, and the precisions, σ_m^{-2}, σ_y^{-2} (inverse variances). As usual, it helps to remember that you can ignore any constant multiplicative factors. These appear as constant terms inside the exponential. Since we are looking for a distribution over M, think carefully about which terms are constant after observing $Y = y$. See equation (2.10) in BDA.

Note: it helps to start by assuming $\hat{m} = 0$, then noticing that, introducing $\hat{m} \neq 0$ is equivalent to translating the origin of your space to be located at $\hat{m}$.

Fill in this cell with your answer

Q1.3: Express the posterior mean, $\hat{m}_{|y}$ as a convex combination of the form $(1 - s(\text{SNR}))\hat{m} + s(\text{SNR})y$ where $s(\cdot)$ is some monotonic, saturating function of the SNR. Is your answer sensible?

Fill in this cell with your answer

Q1.4: Express the posterior precision, $\sigma_{m|y}^{-2}$, in terms of the precisions σ_m^{-2} and σ_y^{-2}. Is your answer sensible?

Fill in this cell with your answer.

Q1.5: Write out, a simple rule that explains how to update the prior (produce the posterior from the prior) after an observation.

Fill in this cell with your answer

1.2.3 Solving for the Posterior (in d Dimensions)

Save deriving the update in multiple dimensions for the end of your lab work. It is worth doing once to justify the heavier formulas that come later. All the main ideas were captured in the 1D example. We will do parts of this in class on Thursday.

Q1.6: In the cell below, repeat the analysis you performed in the 1D case for the generic case. That is, for M and Y in $\mathbb{R}^d$. It helps to be strategic here. Follow the same sequence of steps you performed in the 1D case, dropping constants everywhere you can. If you need help completing the square in multiple dimensions, come talk to us in office hours or raise a hand.

Fill in this cell with your answer

1.2.4 Inspecting the Update Rule

In the previous cell you should have derived the following update:

$$M|Y = y \sim \mathcal{N}(\hat{m}_{|y}, C_{m|y}) \text{ where: } \begin{cases} \hat{m}_{|y} = \hat{m} + (I + (C_m C_y^{-1})^{-1})^{-1}(y - \hat{m}) \\ C_{m|y} = (C_m^{-1} + C_y^{-1})^{-1} \end{cases}$$

There are a couple things to note here. 1. The update is entirely expressed by updating the parameters (here, the mean and the covariance). Here we see the power of conjugacy on full display. Moreover, if we observe multiple samples observations sequentially, then we can just update the mean and covariance recursively. 1. The mean update is sensible. As before, it updates the mean by shifting it towards the observation. The product $C_m C_y^{-1}$ is playing the role of the SNR, and the prefactor is the matrix equivalent to $s(\text{SNR}) = [1+\text{SNR}^{-1}]^{-1}$. If the covariance in the measurement is much larger than the covariance in the noise, then this term is small, so $(I + C_y C_m^{-1})^{-1} \approx I$, so $\hat{m}_{|y} \approx \hat{m}$. 1. The covariance is update is sensible. As before, the precisions add after the observation.

These sorts of updates can be easily extended to similar scenarios. For example, since all linear transformations of normal distributions are normal, we can derive a similar rule for the conditional distribution of some unknown mean M when the observable Y is expressed as a linear transformation of M, plus Gaussian noise. That is $y = Hm + \epsilon$ where $\epsilon \sim \mathcal{N}(0, C_y)$ represents the noise, H represents the mapping responsible for converting state to measurement, and where we have used lower case letters for y and m to distinguish vectors from matrices. This is a standard convention at this stage. In general, we will continue to use hats for fixed quantities (expected values), and capital letters for matrices.

Introducing a linear mapping between the unknown m and the observation y is useful since, in most high-dimensional settings, we have (far) fewer measurements than unknowns. We will assume that H is $n \times d$ where, for, most practical purposes, $n \ll d$.

Introducing the linear mapping only decorates the original expression. Read through the expression carefully to track the change in dimensions. Each time we switch from n to d we are moving from the measurement domain to the state space. Each time we map from d to n we are moving from the state space to the measurement space.

$$m|y \sim \mathcal{N}(\hat{m}_{|y}, C_{m|y}) \text{ where: } \begin{cases} \hat{m}_{|y} = \hat{m} + (C_m^{-1} + H^\top C_y^{-1} H)^{-1} H^\top C_y^{-1}(y - H\hat{m}) \\ C_{m|y} = (C_m^{-1} + H^\top C_y^{-1} H)^{-1} \end{cases} \tag{1}$$

For the next part, it will also be convenient to understand how the posterior distribution updates if we replace m with $Am + \zeta$ where $\zeta \sim \mathcal{N}(0, C_\zeta)$, and $A \in \mathbb{R}^{d \times d}$ is invertible. This represents a noisy linear transformation of m.

Q 1.7: In the space below, find the distribution for $Am + \zeta$. Remember, any linear transformation of a Gaussian vector produces a Gaussian vector, and any sum of Gaussian vectors is a Gaussian vector. Also remember that expectation is linear, so commutes with linear transformations, and the covariance of a sum of independent vectors is the sum of their covariances.

Fill in this cell with your answer.

1.3 Visualization

Q 1.8: In the cell below, write a code that can visualize any 2-D Gaussian density function as a surface. Use color to represent the height of the surface (*and, optionally, add contour lines*). Note, your function should work for singular Gaussian densities. Since we don't care about normalization, your function does not need to plot a properly normalized Gaussian density.

```
[ ]: # function to display 2D Gaussian
```

Q 1.9: Then create a visual showing: 1. A standard normal prior density with $\hat{m} = 0$ and $C_m = I$. 1. The likelihood function when $H = [1, -2]$ and $y = 0.5$. You may assume the noise variance, $C_y = \sigma_y^2 = 0.25$. 1. The resulting posterior distribution.

```
[ ]: # visualizate prior, likelihood, posterior
```

Q 1.10: Set the posterior produced from (Q 1.9), and use it as your prior. Make plots showing: 1. The likelihood function when $H = [1, -1.6]$ and $y = 0.4$. You may assume the noise variance, $C_y = \sigma_y^2 = 0.25$. 1. The resulting posterior.

```
[ ]: # visualize likelihood and posterior
```

Q 1.11: In the space below, comment on how the posterior updated in each step. Are the resulting updates sensible?

Fill this cell with your answer

Q 1.12: Did the second measurement effectively reduce the uncertainty in the posterior? Why or why not? What do you notice about the direction in the state space where uncertainty is constrained when using a one dimensional measurement (*how does it relate to the rows of H?*). Propose a new 1×2 measurement matrix H that you think will lead to the greatest reduction in posterior uncertainty. Explain how you selected your measurement matrix H and in what sense you expect it to maximally reduce uncertainty.

Fill this cell with your answer.

Q 1.13: Use the posterior from (Q 1.12) as your prior and make plots showing: 1. The likelihood function using your new H and a hypothesize measurement that is consistent with past observations (*you may calculate the prior mean $\hat{m}$, and use $H\hat{m}$ so that our posterior mean stays still.*). Keep the noise variance, $C_y = \sigma_y^2 = 0.25$. 1. The new posterior.

Comment on whether your choice of measurement effectively reduced uncertainty.

```
[ ]: # visualize likelihood and posterior
```

Congratulations! You've reached the end of part I. You should have finished up to here by Wednesday, February

1.4 [PART II - Preview] Kalman Filters

Consider the following problem:

You are a meteorologist insterested in predicting the weather. You have access to a large, varied, set of real-time data sources. These provide a constant stream of measurements. Even so, the real weather system is unimaginably high-dimensional. So, even with input from satellites, and weather stations, and weather bouys, and balloon launches, and everything else NOA attempts, omniscience is still far off. Given your data, the true state of the weather at any time is unknown.

Worse still, the weather is constantly changing.

These two ingredients pose a data assimilation problem. In data assimilation, we make measurements in real time of an unknown, *potentially changing*, quantity, usually called the state vector. Unlike the examples we've considered before, asymptotic consistency guarantees don't apply since the unknown is constantly moving. Our aim is to keep "assimilating" new data into a model for the motion of the unknown in order to keep a good guess at the range of possible states given the observed sequence of measurements. In other words, we want to sequentially update a posterior after each measurement.

Data assimilation has two key components: 1. A dynamical systems model that specifies how we would expect the state vector to change over time if we knew it's exact initial conditions at some point in time. This system may be deterministic, or stochastic. In the weather analogy, this is a massive PDE simulator, built on a combination of fundamental physics (i.e. Navier-Stokes) and geochemistry (water cycles, etc.). In all that follows the true, but unknown, state of the system will be represented as $x(t)$ (if in continuous time), or x_t in discrete time. We will treat this as random quantity, but use lower case letters to maintain a clear distinction between matrices and vectors. 1. A statistical model that specifies a prior distribution over the unknowns before any measurement, and an error model relating the true state of the system to the observed measurements. In all that follows we will let y_t represent the measurements made at time t. Typically, y_t is a low dimensional object relative to x_t. Hence, we are in the standard "small sample" statistics regime of Bayesian statistics where we aim to repeatedly solve an ill-posed problem.

Then, we update a posterior distribution after each new measurement, using the posterior after the previous measurement as our prior, and using the dynamical systems model and measurement model to specify the likelihood.

A Kalman filter linearizes the dynamical system and measurement model (assumes small time steps and concentrated posteriors), and uses a (Normal, Normal) conjugate model. In this setting, the posterior over the unknown state remains Gaussian at all times. Since a Gaussian is fully specified by its mean and covariance, applying data assimilation reduces to running an algebraic recursion driven by a constant stream of input measurements.

1.4.1 An overview.

The idea in a Kalman filter is to continuously update a Gaussian approximation to the posterior distribution of the unknown state given a series of sequential measurements. In what follows, let x_t denote the (unknown) state at time t, and y_t denote the corresponding observation. Between each observations, x_t is updated by a noisy dynamical system. As long as the time step is small, the mapping imposed by the system is approximately linear. Similarly, as long as we can collect enough measurements to keep the posterior concentrated, then the action of any, potentially nonlinear, set of measurement functions, are approximately linear. Then, every update is either, a linear transformation plus Gaussian noise, or a posterior inference step based on a linear measurement. Each of these steps preserve the family of Gaussian distributions, so the entire pipeline can be represented by deriving a recursion on the mean and covariance of the state vector $\{\hat{x}_t, P_t\}_{t=1}^{\cdots}$. Since we will condition on all past observations to constrain our inference, we will write, $\hat{x}_{t|t-1}, P_{t|t-1}$ to denote the posterior mean and covariance over the state vector after measurement y_t.

Then, after initializing with a Gaussian prior mean $\hat{x}_0$ and covariance P_0, we will run a recursive process of the form:

1. Push the posterior distribution left after the observation at time t forward to a prior for time $t+1$ by running the dynamics (apply the linear update equation you derived in Q 1.7).
2. Make an observation, y_t
3. Condition on the observation, using the new prior, to derive a new posterior via equation (1).
4. Iterate.

1.4.2 Link to Lab 4:

To go to lab four click this link: Lab 4

Note: Lab 4 is a work in progress. Most of the exposition and code is available for you. Activity prompts will post by next Monday.