CS 287, Fall 2019 Homework #4 Multivariate Gaussians, Kalman Filtering, Maximum Likelihood, EM

Deliverable: Report submitted through Gradescope by Monday Nov 4th, 23:59.

For theory part, please make sure you have detailed steps. Final grades are given based on steps not the final answer. Various starter files are provided in the ipython-notebook. Your report should be a PDF file containing:

Page 1: Problem 1

Page 2: Problem 2

Page 3: Problem 3

Pages 4 & 5: Problem 4 (c)

Page 6 and onwards: PDF rendering of completed ipython notebook, which will contain your answers to Problems 4(a), 4(b), 5, 6.

---

1. Maximum Likelihood [5 pts]

The Poisson distribution is a discrete probability distribution that expresses the probability of a number of events occurring in a fixed period of time if these events occur with a known average rate and independently of the time since the last event. Assume we obtain m i.i.d. samples $x^{(1)}, \ldots, x^{(m)}$ distributed according to the Poisson distribution $P(x = k) = \frac{\lambda^k e^{-\lambda}}{k!}$ for $k = 0, 1, 2, 3, \ldots$. What is the maximum likelihood estimate of λ as a function of $x^{(1)}, \ldots, x^{(m)}$?

2. Linearity of Expectation, Positive Semi-definiteness [5 pts]

A matrix $A \in \mathbb{R}^{nxn}$ is positive semi-definite (often denoted by $A \succeq 0$) if and only if:

$$A_{ij} = A_{ji}$$
$$\forall z \in \mathbb{R}^n : z^\top A z \geq 0$$

If A is also symmetric, prove that covariance matrices, i.e., matrices of the form $\Sigma = \mathrm{E}[(X - EX)(X - EX)^\top]$ are guaranteed to be positive semi-definite.

3. The gaussian integral and completion of squares trick [10 pts]

Let $A \in \mathbb{R}^{n \times n}$ be a positive definite matrix, $b \in \mathbb{R}^n$, and $c \in \mathbb{R}$. Prove that,

$$\oint_{x \in \mathbb{R}^n} \exp(-\frac{1}{2} x^T A x - x^T b - c)\, dx = \frac{(2\pi)^{\frac{n}{2}}}{|A|^{\frac{1}{2}} \exp(c - \frac{1}{2} b^T A^{-1} b)}$$

Hint: You can directly use the fact that the integral of a multivariate Gaussian equals 1.

4. Kalman Filtering, Smoothing, EM

(a) **Implementation of Kalman Filtering** [10 pts]

In this question you will implement a Kalman Filter. Look at Homework4_Q.ipynb for more detailed instructions. **Results should be included in the completed ipython notebook.**

(b) **Implementation of Smoothing** [10 pts]

In this question you will implement a Kalman Smoother. Look at Homework4_Q.ipynb for more detailed instructions. **Results should be included in the completed ipython notebook.**

(c) **Implementation EM.** [10 pts]

In this question you will implement the EM algorithm to estimate the covariance matrices. Look at Homework4_Q.ipynb for more detailed instructions. **Results should be included in the completed ipython notebook.**

(d) **Application to Species Population Size Estimation from Observations of Total Population Size.**[10 pts]

Consider three species U, V, W that grow independently of each other, exponentially with growth rates: U grows 2% per hour, V grows 6% per hour, and C grows 11% per hour. The goal is to estimate the initial size of each population based on the measurements of total population.

Let $x_U(t)$ denote the population size of species U after t hours, for $t = 0, 1, \ldots$, and similarly for $x_V(t)$ and $x_W(t)$, so that

$$x_U(t+1) = 1.02x_U(t), \quad x_V(t+1) = 1.06x_V(t), \quad x_W(t+1) = 1.11x_W(t).$$

The total population measurements are $y(t) = x_U(t) + x_V(t) + x_W(t) + v(t)$, where $v(t)$ are IID, $\mathcal{N}(0, 0.36)$. (Thus the total population is measured with a standard deviation of 0.6).

The prior information is that $x_U(0), x_V(0), x_W(0)$ (which are what we want to estimate) are IID $\mathcal{N}(6, 2)$. (Obviously the Gaussian model is not completely accurate since it allows the initial populations to be negative with some small probability, but we'll ignore that.)

How do you formulate this problem as a Kalman filtering problem by providing your A, B, C, d, Q, R. How long will it be (in hours) before we can estimate $x_U(0)$ with a variance less than 0.01? How long for $x_V(0)$? How long for $x_W(0)$? Starter code can be found in `Homework4_Q.ipynb`. **Results should be included in the completed ipython notebook.**

(e) **Correlated Noise.** [10 pts]

In many practical situations the noise is not independent. Consider the following stochastic system, for which the noise is not independent:

$$
\begin{aligned}
&x_0 \sim N(\mu_0, \Sigma_0) \\
&x_{t+1} = Ax_t + w_t \\
&w_t = 0.3w_{t-1} + 0.2w_{t-2} + p_{t-1} \\
&p_t \sim \mathcal{N}(0, \Sigma_{pp}) \\
&y_t = Cx_t + v_t \\
&v_t = 0.8v_{t-1} + q_{t-1} \\
&q_t \sim \mathcal{N}(0, \Sigma_{qq}) \\
&p_{-1} = q_{-1} = v_{-1} = w_{-1} = w_{-2} = 0
\end{aligned}
$$

Describe how, by choosing the appropriate state representation, the above setup can be molded into a standard Kalman filtering setup. In particular, describe the state, the dynamics model, and the measurement model such that the problem is transformed into the standard Kalman filtering setup with uncorrelated noise.

5. Sensor Selection [10 pts]

We consider the following linear system:

$$
\begin{aligned}
x_{t+1} &= Ax_t + w_t \\
z_t &= C_t x_t + v_t
\end{aligned}
$$

where $A \in \mathbb{R}^{n \times n}$ is constant, but C_t can vary with time. The noise contributions are independent, and

$$x_0 \sim \mathcal{N}(0, \Sigma_0), \quad w_t \sim \mathcal{N}(0, \Sigma_w) \quad v_t \sim \mathcal{N}(0, \Sigma_v).$$

Here is the twist: the measurement matrix C_t at each time comes from the set $\mathcal{S} = \{S_1, \ldots, S_K\}$. In other words, at each time t, we have $C_t = S_{i_t}$. The sequence $i_0, i_1, i_2, \ldots$ specifies which of the K possible measurements is taken at time t. For example the sequence $2, 2, 2, \ldots$ means that $C_t = S_2$ for all t. The sequence $1, 2, \ldots, K, 1, 2, \ldots, K, \ldots$ is called round-robin: we cycle through the possible measurements, in order, over and over again.

Here is the interesting part: *you* get to choose the measurement sequence $i_0, i_1, i_2, \ldots$.

You will work with the following specific system:

$$A = \begin{bmatrix} -0.6 & 0.8 & 0.5 \\ -0.1 & 1.5 & -1.1 \\ 1.1 & 0.4 & -0.2 \end{bmatrix}, \quad \Sigma_w = I, \quad \Sigma_v = 0.1^2, \quad \Sigma_0 = I$$

and $K = 3$ with

$$S_1 = \begin{bmatrix} 0.74 & -0.21 & -0.64 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 0.37 & 0.86 & 0.37 \end{bmatrix}, \quad S_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}.$$

(a) **Using One Sensor**. Plot $trace(\Sigma_{t|0:t})$ versus t for the three special cases when $C_t = S_1$ for all t, $C_t = S_2$ for all t, and $C_t = S_3$ for all t.

(b) **Round-robin.** Plot $trace(\Sigma_{t|0:t})$ versus t using the round-robin sensor sequence $1, 2, 3, 1, 2, 3, \ldots$.

(c) **Greedy Sensor Selection.** Plot $trace(\Sigma_{t|0:t})$ versus t using greedy sensor selection. In greedy sensor selection at time t the choice of $i_0, i_1, \ldots, i_{t-1}$ has already been made and it has determined $\Sigma_{t|0:t-1}$. Then $\Sigma_{t|0:t}$ depends on i_t only, i.e., which of $S_1, \ldots, S_K$ is chosen as C_t. Among these K choices you pick the one that minimizes $trace(\Sigma_{t|0:t})$.

See `Homework4_Q.ipynb` part 3 and look for `Your code here` for the parts that you need to fill in. **Results should be included in the completed ipython notebook.**
Note none of these require knowledge of the measurements $z_0, z_1, \ldots$.

6. Extended Kalman Filter (EKF)[20 pts]

In this question you get to implement an extended Kalman Filter (EKF) for state estimation for nonlinear dynamics and observation models.

Notes: Let $\mathbf{x} \in \mathbb{R}^{xDim}$ be the system state, $\mathbf{u} \in \mathbb{R}^{uDim}$ denote the control input applied to the system, and $\mathbf{z} \in \mathbb{R}^{zDim}$ be the vector of observations obtained about the system state using sensors. We are given a discrete-time stochastic dynamics model that describes how the system state evolves and an observation model that relates the obtained observations to the state, given here in state-transition notation:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{q}_t), \qquad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, Q), \tag{1}$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{r}_t), \qquad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, R). \tag{2}$$

Given the current state estimate $\hat{\mathbf{x}}_t$, control input $\mathbf{u}_t$, covariance Σ_t, and observation $\mathbf{z}_{t+1}$, the EKF update equations are given by:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}) + K_t(\mathbf{z}_{t+1} - \mathbf{h}(\mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}), \mathbf{0})), \tag{3a}$$

$$\Sigma_{t+1} = (I - K_t H_t)\Sigma^-_{t+1} \tag{3b}$$

$$\textit{where} \quad A_t = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}), \qquad M_t = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}), \tag{3c}$$

$$H_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}), \mathbf{0}), \qquad N_t = \frac{\partial \mathbf{h}}{\partial \mathbf{r}}(\mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}), \mathbf{0}), \tag{3d}$$

$$\Sigma^-_{t+1} = A_t \Sigma_t A_t^T + M_t Q M_t^T, \quad K_t = \Sigma^-_{t+1} H_t^T (H_t \Sigma^-_{t+1} H_t^T + N_t R N_t^T)^{-1}. \tag{3e}$$

Q. See `Homework4_Q.ipynb` part 4 and look for `Your code here` for the parts that you need to fill in. Use the routine provided in `numerical_jac` function to compute the required Jacobians (Eqs. (3c), (3d)). The script considers a two-dimensional nonlinear system defined by:

$$\begin{aligned} \mathbf{x}_{t+1}[1] &= 0.1 * (\mathbf{x}_t[1])^2 - 2 * \mathbf{x}_t[1] + 20 + \mathbf{q}_t[1]; \\ \mathbf{x}_{t+1}[2] &= \mathbf{x}_t[1] + 0.3 * \mathbf{x}_t[2] - 3 + 3 * \mathbf{q}_t[2] \\ \text{and } \mathbf{z}_t[1] &= \mathbf{x}_t^T \mathbf{x}_t + \sin(5 * \mathbf{r}_t[1]); \\ \mathbf{z}_t[2] &= 3 * (\mathbf{x}_t[2])^2 / \mathbf{x}_t[1] + \mathbf{r}_t[2], \end{aligned}$$

where dynamics noise $\mathbf{q}_t$ and the measurement noise $\mathbf{r}_t$ are assumed to be drawn from a Gaussian distribution with zero mean and specified variances $Q = \left[\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right]$ and $R = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 10 \end{smallmatrix}\right]$, respectively.

For verifying correctness, starting from an initial state $\mathbf{x}_0 \sim \mathcal{N}(\left[\begin{smallmatrix} 10 \\ 10 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right])$, a random trajectory $\mathcal{X}_{1:50}$ was generated and corresponding observations $\mathcal{Z}_{1:50}$ were collected.

The provided code plots the ground truth value of the state $\mathcal{X}_{1:50}$ along each dimension and also plots the state estimate given by the EKF (blue dotted line) and plots 3-standard deviations of the variance corresponding to each dimension. It also prints out the state estimate $\hat{\mathbf{x}}_{50}$ and covariance Σ_{50} at the final time step to help you verify correctness of your implementation. **Results should be included in the completed ipython notebook.**