Bayesian Estimation and Stochastic Optimization Assignments

Yifei Dong

November 20, 2023

1 Assignment 1

For the implementation and simulation of HMM, we utilized the convenient hmmlearn library [hmm23], and the implementation (also the other three assignments) can be found in my GitHub repository [git23].

Problem 1

The HMM is initialized with initial state probabilities $\pi = [0.2, 0.7, 0.1]$, transition probabilities

$$A = \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.3 & 0.3 \end{pmatrix},$$

and the states are represented by $\{0, 1, 2\}$.

Here is the plot showing the Mean Square Error (MSE) of both the HMM filter and smoother as a function of noise variance Fig. 1. As observed, the MSE for both the filter and smoother increases with an increase in noise variance with a slight difference.

Problem 2

As an example, we only vary the observation noise variance here, which is a part of the model parameters. Here is the plot of the log-likelihood of observations with varying noise variance given 50 samples for each variance level and uniform transition probabilities 2.

Problem 3

With the same transition matrix and start probability as in problem 2, as well as a sample size of 500 and observation noise variance of 0.1, we did HMM training using the Expectation Maximization algorithm to estimate the model parameters. The estimated transition matrix

$$\bar{A} = \begin{pmatrix} 0.1829 & 0.3049 & 0.5117 \\ 0.2297 & 0.7637 & 0.0065 \\ 0.7041 & 0.0740 & 0.2218 \end{pmatrix},$$

corrected by the noise, shows a discrepancy from the original transition matrix.

A detailed derivation of the EM algorithm (Baum-Welch Algorithm) can be found in the Appendix, which refers to [Tu].

2 Assignment 2

In a Hidden Markov Model (HMM), we have a linear transition model $x_{k+1} = f(x_k, u_k) = A_k x_k + B_k u_k + w_k$ and an observation model $z_k = g(x_k) = C_k x_k + v_k$. w_k and v_k are the process noise and the measurement noise, assumed to be Gaussian with zero mean and variance Q and R, respectively. Bayesian recursion in the context of filtering involves two steps: prediction and update. It updates the probability distribution of a state based on new measurements.

- A prior step: This step predicts the state at the next time step based on the current state estimate.

$$\hat{x}_{k|k-1} = A_{k-1}\hat{x}_{k-1|k-1} + B_{k-1}u_{k-1}$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k$$

  Here, $\hat{x}_{k|k-1}$ is the predicted state at time k given all observations up to time $k-1$ according to the properties of an HMM. $P_{k|k-1}$ is the corresponding predicted state covariance.

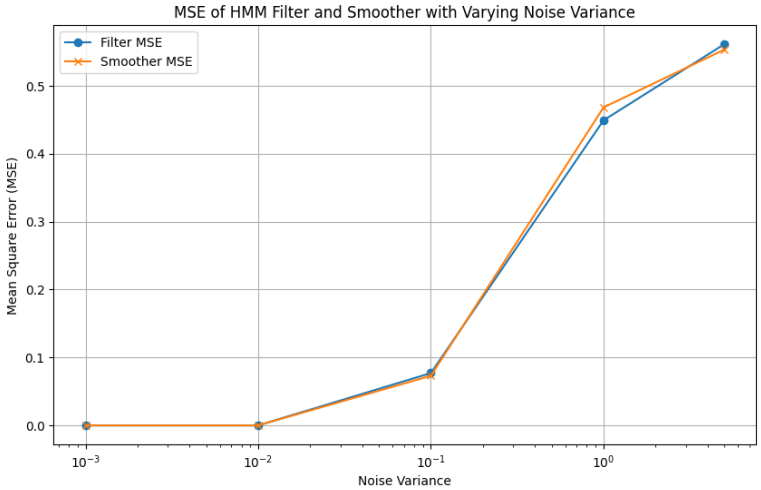


Figure 1: Assignment 1.1 - MSE of both the HMM filter and smoother as a function of noise variance.

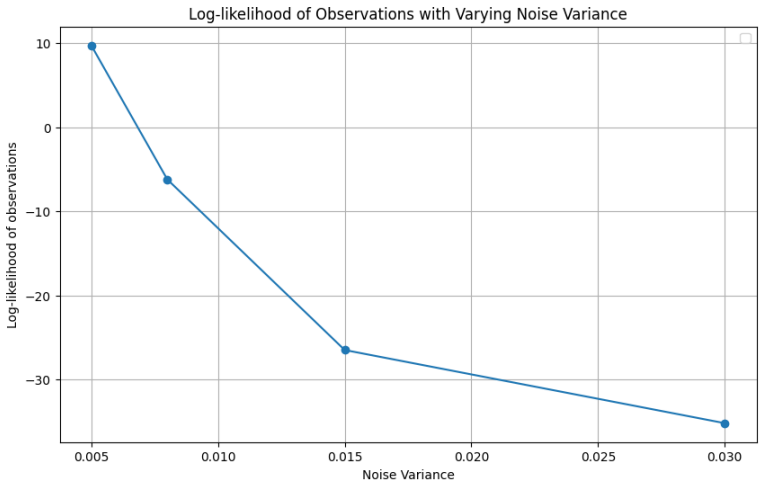


Figure 2: Assignment 1.2 - Log-likelihood of observations with varying noise variance.

- A posterior step: This step updates the predicted state based on new measurements.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C_{k-1}\hat{x}_{k|k-1})$$
$$K_k = P_{k|k-1}C_k^T(C_k P_{k|k-1}C_k^T + R_k)^{-1}$$
$$P_{k|k} = (I - K_k C_k)P_{k|k-1}$$

  K_k is the Kalman gain, y_k is the measurement at time k.

3 Assignment 3

The problem is implemented in Matlab with the following values for parameters:

$$\begin{aligned}\Delta &= 1.0\\ z_0 &= [0,0,0,0]^T\\ r_{k+1} &= [0.1, 0.1]^T\\ R &= \text{diag}\,([r,r])\\ Q &= \text{diag}\,([q,q,q,q])\end{aligned}$$

Two problems are discussed as follows,

- *Illustrate the use of the Kalman filter for estimating the target's state (position and velocity) for various states and observation noise variances.* Four combinations of $[q, r]$,

$$[q,r] \in \{[0.01, 0.1], [0.01, 1.0], [0.1, 0.1], [0.1, 1.0]\}, \tag{1}$$

  are discussed and plotted in Fig. 3. Apparently, a higher observation noise variance leads to measurements and estimated or predicted paths away from the true path. A higher state noise variance makes the measurement points more uneven.

- *Illustrate the performance of the optimal predictor for the target's state.* To achieve this, we update the estimated state $\hat{z}_k$ with the system transition dynamics,

$$\bar{z}_{k+1} = A\hat{z}_k + fr_{k+1}, \tag{2}$$

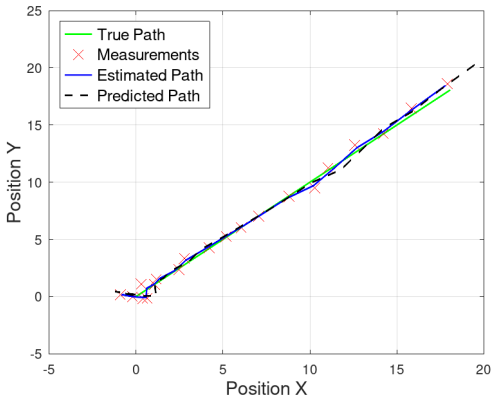
  and compare the predicted next state, $\bar{z}_{k+1}$, with the estimated next state $\hat{z}_{k+1}$, the true state and the measurement in Fig. 3. The optimal predictor can in principle predict the next states, whose performance is affected by state and observation variances.

4 Assignment 4

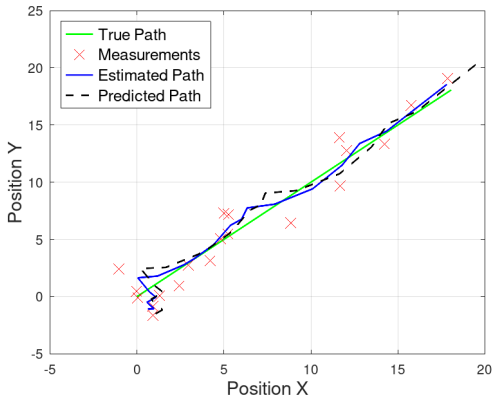
The problem is initialized with

$$\begin{aligned}x_1 &= 0.0\\ a &= 0.5\\ \Sigma_w &= 0.1\\ \Sigma_v &= 0.1\\ n_p &= 1000\\ n_g &= 1000\\ n_t &= 100\end{aligned}$$

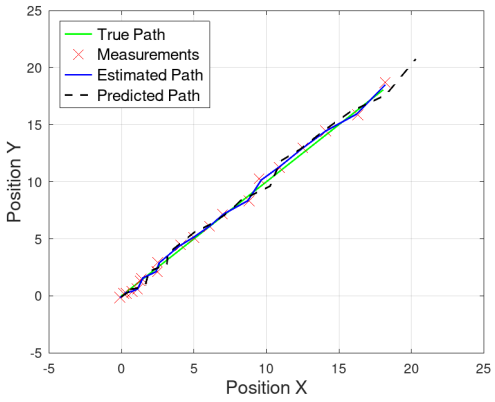
Here, n_p, n_g, and n_t refer to the number of particles in PF, the number of grid points in GQ, and the number of time steps.



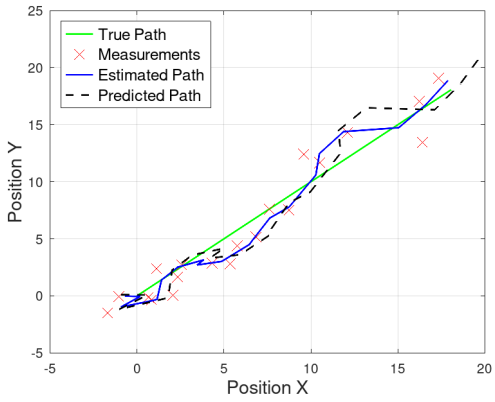
(a) $q = 0.01, r = 0.1$



(b) $q = 0.01, r = 1.0$



(c) $q = 0.1, r = 0.1$



(d) $q = 0.1, r = 1.0$

Figure 3: Assignment 3 - True path, measurements, estimated path, and predicted path in a Kalman filtration problem applied in a 2D constant-acceleration double integrator scenario.

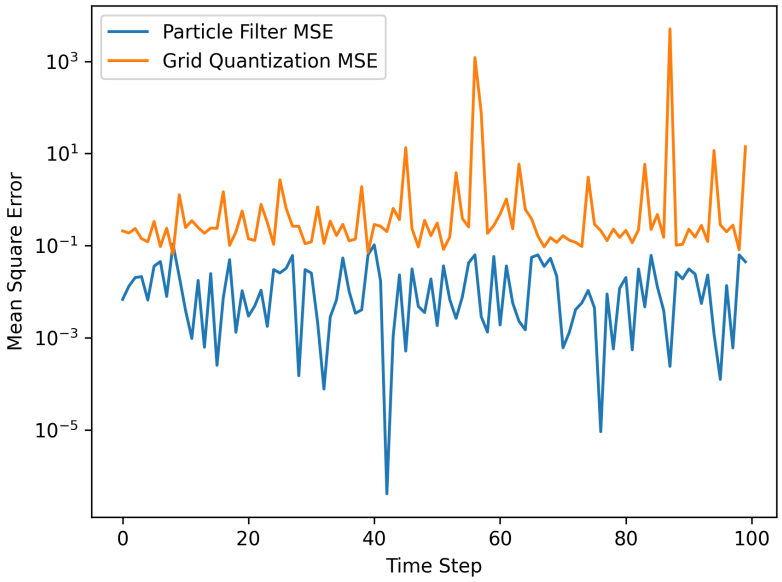


Figure 4: Assignment 4 - The particle filter's performance to a grid quantization of the posterior filtering equation update in terms of the mean square error (MSE) of the state estimate.

The 1D grid is given by n_g evenly distributed points with a bound of $[-2, 2]$, since we initialized $x_1 = 0.0$, together with $|a| < 1$ and arctan operation in the system observation, the scalar state is very likely bounded to a small region around the origin, affected by noises. We compared the particle filter performance to a grid quantization of the posterior filtering equation update in terms of the mean square error (MSE) of the state estimate. The results are visualized in 4, which indicates a better performance of PF over GQ, since GQ very much relies on the resolution of quantization (the number of grid points) and the grid interval.

References

[git23] git. Assignment-related course material, 2023. https://github.com/YvesDong/BayesianEstimation.

[hmm23] hmmlearn Developers. hmmlearn: Unsupervised learning and inference of hidden markov models, 2023. https://github.com/hmmlearn/hmmlearn.

[Tu] Stephen Tu. Derivation of baum-welch algorithm for hidden markov models. https://stephentu.github.io/writeups/hmm-baum-welch-derivation.pdf.