TTIC 31170: Robot Learning and Estimation (Spring 2025)

Problem Set #1

Due Date: April 10, 2025

1 Bayes Mean-Square Error Estimator [20 pts]

Consider the case in which we are interested in a estimating a continuous, multivariate random variable $\boldsymbol{x} \in \mathbb{R}^n$ based upon noisy continuous, vector-valued observations. Suppose that the prior $p(\boldsymbol{x})$ over the latent random variable is known.

(a) **[4pts]** Derive an expression for the estimator that minimizes the mean-square error

$$\hat{\boldsymbol{x}}_{\text{BLS}}(\boldsymbol{z}) = \underset{\boldsymbol{f}(\cdot)}{\arg\min}\ \mathbb{E}_{\boldsymbol{x}|\boldsymbol{z}}\left[\|\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{z})\|_2^2\right] \tag{1}$$

where $\mathbb{E}_{\boldsymbol{x}|\boldsymbol{z}}[\cdot]$ denotes expectation with respect to $p(\boldsymbol{x} \mid \boldsymbol{z})$.[1]

We can rewrite the expectation as

$$E_{x|z}||x - f(z)||_2^2 = \int ||x - f(z)||_2^2 p(x|z)dx$$

We can optimize this with respect to $f(z)$ and set it equal to 0,

$$0 = \int -2(x - f(z))p(x|z)dx = \int xp(x|z)dx - f(z)\int p(x|z)dx$$

And since p is a probability density, its integral is simply 1, which gives us

$$\hat{x}_{BLS} = f(z) = \int xp(x|z)dx$$

(b) **[6pts]** Prove that an estimator $\hat{\boldsymbol{x}}(\cdot)$ minimizes the mean-square error (Eqn. 1) if and only if the associated estimation error $\boldsymbol{e}(\boldsymbol{x}, \boldsymbol{z}) = \hat{\boldsymbol{x}}(\boldsymbol{z}) - \boldsymbol{x}$ is uncorrelated with (orthogonal to) any (vector-valued) function of the data $\boldsymbol{g}(\boldsymbol{z})$, i.e.,

$$\mathbb{E}_{\boldsymbol{x}}\left[(\hat{\boldsymbol{x}}(\boldsymbol{z}) - \boldsymbol{x})\boldsymbol{g}^\top(\boldsymbol{z})\right] = \boldsymbol{0} \tag{2}$$

[1] Note that this is equivalent to minimizing the expectation with respect to $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$, as we effectively can minimize the above quantity for every possible $\boldsymbol{z}$, i.e., optimize the bracketed term in arg min $\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}\|(x) - \boldsymbol{f}(\boldsymbol{z})\|_2^2\, p(\boldsymbol{x} \mid \boldsymbol{z})d\boldsymbol{x}\right]d\boldsymbol{z}$ for every $\boldsymbol{z}$.

Assume that $\hat{x}(z)$ minimizes the MSE and consider $\tilde{x}(z) = \hat{x}(z) + \epsilon g(z)$. We can now consider the MSE for $\tilde{x}$,

$$E[||x - (\hat{(x)}(z) + \epsilon g(z)||_2^2] = E[||x - \hat{x}(z)||^2] - 2\epsilon E[(x - \hat{x}(z))^T g(z)] + \epsilon^2 E||g(z)||^2$$

Since we know $\hat{x}(z)$ minimizes the MSE, the partial of the above wrt ϵ must vanish at $\epsilon = 0$,

$$0 = -2E[(x - \hat{x}(z))^T g(z)] \rightarrow E[(x - \hat{x}(z))^T g(z)] = 0$$

If we have that $\hat{x}(z)$ minimizes the MSE, then for all $g(z)$, the first order derivative vanishes, which shows that the estimation error is orthogonal to any $g(z)$.

Conversely, if for a given $\hat{x}(z)$, the estimation error is orthogonal to any $g(z)$ as written, then this implies that the first order variation of the MSE is zero in all directions, which implies that there is no descent direction and so $\hat{x}(z)$ must minimize the MSE.

(c) **[4pts]** Letting Σ_{BLS} be the error covariance of the estimator derived in (a), show that $\Sigma_{\text{BLS}} \leq \Sigma_e$, where Σ_e is the error covariance associated with any estimator $\hat{\boldsymbol{x}}(\cdot)$. **Hint**: use the orthogonality specified in Eqn. 2.

We know

$$\Sigma_{BLS} = E[e_{BLS} e_{BLS}^T] = E[(\hat{x}_{BLS}(z) - x)(\hat{x}_{BLS}(z) - x)^T]$$

Consider the difference between an arbitrary estimator and the BLS estimator,

$$d(z) = x(z) - \hat{x}_{BLS}(z) \rightarrow e = \hat{x}(z) - x = e_{BLS} + d(z)$$

which we can plug into

$$\Sigma_e = E[(e_{BLS} + d(z))(e_{BLS} + d(z))^T] = \Sigma_{BLS} + E[e_{BLS} d(z)^T] + E[d(z) e_{BLS}^T] = E[d(z) d(z)^T]$$

By eqn 2, we know that $E[e_{BLS} d(z)^T] = 0 = E[d(z) e_{BLS}^T]$, so the above becomes

$$\Sigma_e = \Sigma_{BLS} + E[d(z) d(z)^T]$$

Since the second term is a covariance matrix, it must be positive semidefinite and so we have our inequality.

(d) **[2pts]** Derive the expression for the estimator that minimizes the weighted mean-square error

$$\hat{\boldsymbol{x}}_{\text{WLS}}(\boldsymbol{z}) = \arg\min_{\boldsymbol{f}(\cdot)} \int_{-\infty}^{\infty} (\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{z}))^\top S (\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{z})) p(\boldsymbol{x} \mid \boldsymbol{z}) d\boldsymbol{x},$$

where S is a positive definite matrix.

We can differentiate with respect to $f(z)$ to get

$$\frac{\partial}{\partial f(z)} (e^T S e) = -2S(x - f(z))$$

Taking the expectation and setting it equal to 0 yields

$$E[-2S(x - f(z))|z] = 0 \rightarrow f(z) = E[x|z]$$

since S is invertible and so $\hat{x}_{WLS}(z) = E[x|z]$.

(e) **[4pts]** Derive the expression for the estimator that minimizes the expected loss $\mathbb{E}[L(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{z}))]$, where the loss has the form

$$L(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{z})) = \begin{cases} 0 & \text{if } \|\boldsymbol{f}(\cdot) - \boldsymbol{x}\| < \epsilon \\ 1 & \text{else} \end{cases}$$

where you can assume that $\epsilon \rightarrow 0$.

We can fix z to consider the two cases,

$$E[L(x, f(z))|z] = \int_{|f(z)-x|<\epsilon} 0 \cdot p(x|z)dx + \int_{|f(z)-x|\geq\epsilon|} 1 \cdot p(x|z)dx$$

of which the first term simplifies to 0. Note though, that we can still rewrite the second term,

$$= 1 - \int_{|f(z)-x|<\epsilon} p(x|z)dx$$

So as we consider the ball around x, we can approximate the integral as $\epsilon \rightarrow 0$,

$$\int_{|f(z)-x|<\epsilon} p(x|z)dx \approx p(f(z)|z)k$$

where k represents the volume of the ball around x with radius ϵ (note that it is independent of $f(z)$). It is clear that minimizing the loss here is equivalent to maximizing $p(f(z)|z)$ and so the estimator that minimizes expected loss here in the limit is the estimator that maximizes the posterior,

$$\hat{x}_{WLS}(z) = \max_x p(x|z)$$

2 Linear Estimators [30 pts]

The optimal mean-square error estimator is not guaranteed to be linear in the observations. In such cases, we may prefer a linear estimator for computational reasons.

Consider a system with a latent state represented by a deterministic (but unknown) continuous quantity x. Suppose that you have two observations of x corrupted by additive noise,

$$z_1 = x + v_1$$
$$z_2 = x + v_2,$$

where $v_1 \sim \mathcal{N}(0, \sigma_1^2)$ and $v_2 \sim \mathcal{N}(0, \sigma_2^2)$ are Gaussian random variables with zero-mean and variance σ_1^2 and σ_2^2, respectively.

Your task is to derive an estimator $\hat{x} = k_1 z_1 + k_2 z_2$ for x that is linear in the observations z_1 and z_2.

(a) **[6pts]** Assuming that the additive noises are independent, i.e., $\mathbb{E}(v_1 v_2) = 0$:

(i) Derive a constraint on k_1 and k_2 that ensures that the estimator is unbiased.

For the estimator to be unbiased, we want $E[\hat{x}] = x$ and since $E[z_1] = x = E[z_2]$, we have

$$E[\hat{x}] = k_1 E[z_1] + k_2 E[z_2] = k_1 x + k_2 x = x(k_1 + k_2)$$

We can simplify this to

$$x = (k_1 + k_2)x \implies k_1 + k_2 = 1$$

(ii) Derive the settings for k_1 and k_2 that minimize the mean-square error $\mathbb{E}(\tilde{x}^2)$, where $\tilde{x} = x - \hat{x}$ assuming that the estimator is unbiased. Can you provide intuition for these values for k_1 and k_2?

We use $z_i = x + v_i$,

$$\tilde{x} = x - (k_1 z_1 + k_2 z_2) = (1 - (k_1 + k_2))x - (k_1 v_1 + k_2 v_2)$$

$$\tilde{x} = -(k_1 v_1 + k_2 v_2) \implies E[(\tilde{x})^2] = E[(k_1 v_1 + k_2 v_2)^2]$$

Since the noises are independent and zero-mean, we can ignore the cross-term,

$$E[\tilde{x}^2] = k_1^2 \sigma_1^2 + k_2^2 \sigma_2^2$$

Then, by substituting and differentiating,

$$\frac{dJ}{dk_1} = 2k_1\sigma_1^2 - 2(1-k_1)\sigma_2^2 = 0 \implies k_1 = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}, k_2 = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}$$

In essence, these weights allocate more weight to the observation with lower noise variance. For example, if σ_1^2 is very small compared to σ_2^2, then k_2 becomes small and nearly all the weight is put on z_1.

(iii) What is the resulting mean-square estimation error and how does it compare to the corresponding error that would result from using either z_1 or z_2 to estimate x?

$$E[\tilde{x}^2] = k_1^2\sigma_1^2 + k_2^2\sigma_2^2 = \left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\right)^2 \sigma_1^2 + \left(\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right)^2 \sigma_2^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

Using only either z_1 or z_2 would result in error variance of σ_1^2 or σ_2^2 respectively. Here, $\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2} < \min(\sigma_1^2, \sigma_2^2)$, so using both of them results in improved accuracy.

(b) **[4pts]** Repeat the above derivation for the case that the noise terms are correlated, i.e., $\mathbb{E}[v_1 v_2] = \rho\sigma_1\sigma_2$, where ρ is the correlation coefficient ($|\rho| \leq 1$).

The difference is that now the cross-term does not vanish, so

$$E[\tilde{x}^2] = k_1^2\sigma_1^2 + k_2^2\sigma_2^2 + 2k_1 k_2 \rho\sigma_1\sigma_2$$

We substitute and differentiate,

$$2k_1\sigma_1^2 - 2(1-k_1)\sigma_2^2 + 2\rho\sigma_1\sigma_2 - 4k_1\rho\sigma_1\sigma_2 = 0$$

$$2k_1(\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2) = 2\sigma_2^2 - 2\rho\sigma_1\sigma_2$$

$$\implies k_1 = \frac{\sigma_2^2 - \rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}, k_2 = 1 - k_1 = \frac{\sigma_1^2 - \rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}$$

We can then substitute these back into $E[\tilde{x}^2]$ to get the final MSE, which is pretty messy. Regardless, it is clear that when the noise terms are positively correlated, the benefit from combining them is somewhat reduced.

(c) **[4pts]** More generally, suppose that we are interested in a estimating a continuous, vector-valued quantity $\boldsymbol{x} \in \mathbb{R}^n$ based upon continuous, vector-valued observations $\boldsymbol{z} \in \mathbb{R}^m$ ($m > n$) that are linear in $\boldsymbol{x}$

$$\boldsymbol{z} = H\boldsymbol{x} + \boldsymbol{v},$$

where $\boldsymbol{v}$ denotes additive random noise.

(i) Assuming that $\boldsymbol{x}$ is a non-random constant, derive the expression for the *least-squares estimator* $\hat{\boldsymbol{x}}$ that minimizes the sum of squares of the measurement residual $z_i - \hat{z}_i$, i.e.,

$$\hat{\boldsymbol{x}}(\cdot) = \underset{\boldsymbol{f}(\cdot)}{\text{arg min}} \|\boldsymbol{z} - H\boldsymbol{f}(\boldsymbol{z})\|_2^2.$$

We substitute and take the derivative,

$$||Hx + v - Hf(z)||_2^2 \to -2H^T(z - Hf(z)) = 0$$

$$\implies \hat{x} = (H^TH)^{-1}H^Tz$$

(ii) Derive the expression for the weighted least-squares estimator that minimizes the weighed sum of squares

$$\hat{\boldsymbol{x}}(\cdot) = \underset{\boldsymbol{f}(\cdot)}{\text{arg min}} \|\boldsymbol{z} - H\boldsymbol{f}(\boldsymbol{z})\|_R^2$$

where R is a symmetric, positive definite matrix that specifies the weights associated with deviations.

For a weight matrix R,

$$J(x) = ||z - Hx||_R^2 = (z - Hx)^TR(z - Hx)$$

$$-2H^TR(z - Hx) = 0 \to H^TRHx = H^TRz$$

$$\implies \hat{x} = (H^TRH)^{-1}H^TRz$$

(iii) In the event that the additive noise is zero-mean Gaussian $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, Q)$ with covariance matrix Q, derive the estimate that maximizes the likelihood of the observations, i.e., the maximum likelihood estimate (MLE). How does this relate to the answer that you got for part (i)?

$$\hat{\boldsymbol{x}}(\cdot) = \underset{\boldsymbol{x}}{\text{arg max}}\ p(\boldsymbol{z} \mid \boldsymbol{x})$$

Given $v \sim N(0, Q)$, we have the likelihood as

$$p(z|x) \propto \exp[-\frac{1}{2}(z - Hx)^T Q^{-1}(z - Hx)]$$

So maximizing the likelihood is equivalent to maximizing

$$(z - Hx)^T Q^{-1}(z - Hx) \to -2H^T Q^{-1}(z - Hx) = 0$$

$$\implies \hat{x}_{MLE} = (H^T Q^{-1} H)^{-1} H^T Q^{-1} z$$

This is very similar to the answers from the two previous parts, in particular if we set $R = Q^{-1}$.

(d) **[8pts]** Now, suppose that you have a discrete-time sequence of observations of the form $z_t = x + v_t$ for $t \in \{1, \ldots, T\}$, where v_t is a zero-mean Gaussian white sequence (i.e., elements of the sequence are uncorrelated, $\mathbb{E}[v_s v_t] = 0 \; \forall s \neq t$).

(i) Derive the expression for the minimum variance unbiased estimator $\hat{x}(z^T)$ that is a linear function of the measurements z^T.

Our unbiased linear estimator must satisfy $\hat{x} = \sum_t a_t z_t$ with $\sum_t a_t = 1$. We have the MSE as

$$E[(x - \hat{x})^2] = \sum_t a_t^2 \sigma^2$$

with the minimum as $a_t = \frac{1}{T}$ over all T. This gives us the MVUE as

$$\hat{x}(z^T) = \frac{1}{T} \sum_t z_t, \text{Var}(x - \hat{x}) = \frac{\sigma^2}{T}$$

(ii) Show that the estimator derived in part (i) is actually **the** minimum variance unbiased estimator (i.e., even when compared to nonlinear estimators). **Hint**: You may find it helpful to use the Cramér-Rao bound.

The Cramer-Rao bound gives us the lower bound as $\frac{\sigma^2}{T}$, which implies that the MVUE above is the minimum possible.

(iii) The above filter requires storing the entire sequence of observations. In the interest of having an estimator with constant memory requirements, derive a recursive estimator $\hat{x}_t$ in terms of the current measurement z_t and previous estimate $\hat{x}_{t-1}$.

To avoid storing all T observations, define $\hat{x}_t$ as the estimate after t observations. We can then initialize our recursive setup wtih $\hat{x}_0$, which is either arbitrary or simply zero. Then,

$$\hat{x}_t = \frac{t-1}{t}\hat{x}_{t-1} + \frac{1}{t} z_t$$

This is essentially a running average that only requires constant memory.

(e) **[8pts]** Consider the case in which $\boldsymbol{x}$ is not deterministic, but rather is a multivariate stochastic process with linear, potentially time-variant dynamics

$$\boldsymbol{x}_k = A_k \boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1},$$

where $\boldsymbol{w}_{k-1}$ is a zero-mean white sequence. We have access to noisy vector-valued observations that are linear in the latent state

$$\boldsymbol{z}_k = H_k \boldsymbol{x}_k + \boldsymbol{v}_k,$$

where $\boldsymbol{v}_k$ is zero-mean noise with covariance R. For memory and computational efficiency, we would like to find a linear, recursive estimator of the form

$$\hat{\boldsymbol{x}}_k^+ = K_k^- \hat{\boldsymbol{x}}_k^- + K_k \boldsymbol{z}_k, \tag{3}$$

where $\hat{\boldsymbol{x}}_k^-$ and $\hat{\boldsymbol{x}}_k^+$ denote the estimates before and after incorporating the latest observation $\boldsymbol{z}_k$, respectively. The terms K_k^- and K_k are time-varying matrices that weigh the contributions of the previous estimate $\hat{\boldsymbol{x}}_k^-$ (which incorporates all previous observations $\boldsymbol{z}^{k-1}$) and the latest observation $\boldsymbol{z}_k$.

(i) Assuming that $\hat{\boldsymbol{x}}_k^-$ is unbiased (i.e., $\mathbb{E}[\tilde{\boldsymbol{x}}_k^-] = \mathbf{0}$), derive an expression for K_k^- in terms of K_k that ensures that the estimator remains unbiased. **Hint**: Find a recursive formulation for the estimation error $\tilde{\boldsymbol{x}}_k^+$ in terms of $\boldsymbol{x}_k$ and $\tilde{\boldsymbol{x}}_k^-$.

We substitute

$$\tilde{x}_k^+ = x_k - (K_k^- \hat{x}_k^- + K_k z_k) = (\hat{x}_k^- + \tilde{x}_k^-) - (K_k^- \hat{x}_k^- + K_k z_k)$$

Rearranging yields

$$\tilde{x}_k^+ = (I - K_k^-)\hat{x}_k^- + \tilde{x}_k^- - K_k z_k$$

Given $z_k = H_k x_k + v_k$,

$$z_k = H_k \hat{x}_k^- + H_k \tilde{x}_k^- + v_k$$

Which we can plug into the above

$$\tilde{x}_k^+ = (I - K_k^- - K_k H_k)\hat{x}_k^- + (I - K_k H_k)\tilde{x}_k^- - K_k v_k$$

We need $E[\tilde{x}_k^+] = 0$ and we assumed $E[\tilde{x}_k^-] = 0, E[v_k] = 0$, so the only potential source of bias is the term with $\hat{x}_k^-$. To ensure it vanishes, we need

$$I - K_k^- - K_k H_k = 0 \implies K_k^- = I - K_k H_k$$

(ii) What is the resulting expression for the estimator (Eqn. 3)? In a few sentences, explain what it is doing.

The resulting expression becomes

$$\hat{x}_k^+ = (I - K_k H_k)\hat{x}_k^- + K - k z_k$$

This estimator is taking the prior estimate $\hat{x}_k^-$ and adjusting it by subtracting a portion of the measurement prediction error to ensure that the update is unbiased. We then add the new measurement in z_k weighted by K_k to blend the prediction and observation to get a new state estimate.

(iii) Derive an expression for the estimator covariance $\Sigma_k^+ = \mathbb{E}[\tilde{\boldsymbol{x}}_k^+ \tilde{\boldsymbol{x}}_k{}^{+\top}]$ in terms of Σ_k^-. **Hint**: Remember that the noise is white and uncorrelated with the state.

We showed the estimation error above as

$$\tilde{x}_k^+ = x_k - \hat{x}_k^+ = (I - K_k H_k)\tilde{x}_k^- - K_k v_k$$

The covariance is defined as $\Sigma_k^+ = E[\tilde{x}_k^+ \tilde{x}_k^{+T}]$,

$$\Sigma_k^+ = E[((I - K_k H_k)\tilde{x}_k^- - K_k v_k)((I - K_k H_k)\tilde{x}_k^- - K_k v_k)^T]$$

$$= E[(I - K_k H_k)\tilde{x}_k^- \tilde{x}_k^{-T}(I - K_k H_k)^T] + E[K_k v_k v_k^T K_k^T] - E[(I - K_k H_k)\tilde{x}_k^- v_k^T K_k^T]$$

$$-E[K_k v_k \tilde{x}_k^{-T}(I - K_k H_k)^T]$$

Since the noise is white and uncorrelated, the cross-covar terms vanish,

$$\Sigma_k^+ = (I - K_k H_k)\Sigma_k^-(I - K_k H_k)^T + K_k E[v_k v_k^T] K_k^T$$

(iv) Up to this point, we have derived everything in terms of K_k as well as known parameters, but we haven't decided on the best setting for K_k. We would like to find a setting for K_k that minimizes the weighted mean-square error $J_k = \mathbb{E}[\tilde{\boldsymbol{x}}_k^{+\top} S \tilde{\boldsymbol{x}}_k{}^+]$, where $S \geq 0$. In fact, you can show that the optimal setting for the gain matrix is independent of S, so for simplicity let's assume that $S = I$, in which case $J_k = \text{trace}[\Sigma_k^+]$. Derive the expression for the optimal gain matrix. **Hint**: For matrices A and B, where B is symmetric, $\frac{\partial}{\partial A}\text{trace}[ABA^\top] = 2AB$.

We have Σ_k^+ as above and a cost function $J_k = trace(\Sigma_k^+)$. We begin by expanding to get

$$J_k = trace(\Sigma_k^-) - 2trace(K_k H_k \Sigma_k^-) + trace(K_k(H_k \Sigma_k^- H_k^T + R)K_k^T)$$

We can then differentiate, ignoring the first term since it's independent of K_k. The second term becomes

$$-2trace(K_k H_k \Sigma_k^-) \to -2(H_k \Sigma_k^-)^T = -2\Sigma_k^- H_k^T$$

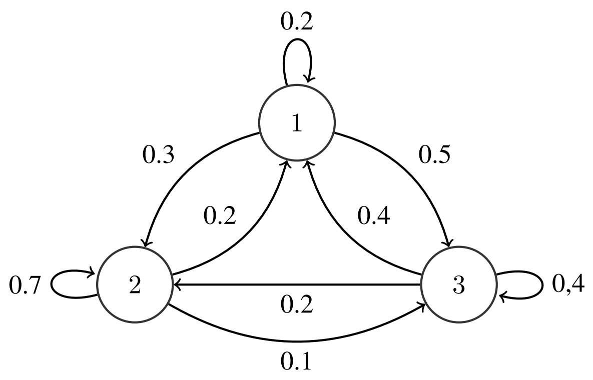
The third term becomes with the hint,

$$K_k(H_k \Sigma_k^- H_k^T + R)K_k^T \to 2K_k(H_k \Sigma_k^- H_k^T + R)$$

We then set these to 0,

$$-2\Sigma_k^- H_k^T + 2K_k(H_k \Sigma_k^- H_k^T + R) = 0$$

$$\to K_k = \Sigma_k^- H_k^T (H_k \Sigma_k^- H_k^T + R)^{-1}$$



		X_t		
		1	2	3
	1	0.6	0.1	0.2
Z_t	2	0.1	0.7	0.3
	3	0.3	0.2	0.5

Figure 1: State transition diagram.

3 Hidden Markov Models: Filtering and Smoothing [20 pts]

Consider a hidden Markov model consisting of a discrete-valued, latent process $X_t \in \{1, 2, 3\}$ and observations $Z_t \in \{1, 2, 3\}$. Figure 1 presents the state transition diagram with the transition and observation (emission) $P(Z_t \mid X_t)$ probabilities. The distribution over the initial state is uniform.

Assume that we observed the sequence $Z^T = \{2, 3, 1\}$.

(a) **[6pts]** Determine the maximum a posteriori (MAP) state estimate at each step using filtering, i.e., $X_t^* = \arg\max_{X_t} P(X_t \mid Z^t)$.

With filtering, we use the current history of observations as

$$P(X_t, Z^t), P(X_t = i|Z_0, Z_1, ..., Z_t) = \frac{\alpha_t(i)}{\sum_j \alpha_t(j)}$$

At time $t = 1$, we initialize with

$$Z_1 = 2, P(X_1 = i) = 1/3$$

$$\rightarrow \alpha_1(i) = P(x_1 = i)P(Z_1 = 2|X_1 = i) = \frac{1}{3} \cdot P(Z = 2|X = i)$$

$$a_1(1) = \frac{1}{3} \cdot P(Z = 2|X = 1) = \frac{1}{3} \cdot 0.1 = \frac{1}{30}$$

$$a_1(2) = \frac{1}{3} \cdot P(Z = 2|X = 2) = \frac{1}{3} \cdot 0.7 = \frac{7}{30}$$

$$a_1(3) = \frac{1}{3} \cdot P(Z = 2|X = 3) = \frac{1}{3} \cdot 0.3 = \frac{1}{10}$$

Therefore, $\sum_j \alpha_1(j) = \frac{11}{30}$, so

$$P(X_1 = 1|Z_1) = \frac{1/30}{11/30} = \frac{1}{11}$$

$$P(X_1 = 2|Z_1) = \frac{7/30}{11/30} = \frac{7}{11}$$

$$P(X_1 = 3|Z_1) = \frac{1/10}{11/30} = \frac{3}{11}$$

We see that $X_1 = 2$ has the highest probability, so the MAP at $t = 1$ is $X_1^* = 2$.
At time $t = 2$, we observe $Z_2 = 3$ and compute transition sums as

$$\alpha_2(j) = \left[\sum_i \alpha_1(i) P(X_2 = j|X_1 = i)\right] P(Z_2 = 3, X_2 = j)$$

So,

$$\alpha_2(1) = \left(\frac{1}{30} \cdot 0.2 + \frac{7}{30} \cdot 0.2 + \frac{1}{10} \cdot 0.4\right) 0.3 = \frac{7}{250}$$

$$\alpha_2(2) = \left(\frac{1}{30} \cdot 0.3 + \frac{7}{30} \cdot 0.7 + \frac{1}{10} \cdot 0.2\right) 0.2 = \frac{29}{750}$$

$$\alpha_2(3) = \left(\frac{1}{30} \cdot 0.5 + \frac{7}{30} \cdot 0.1 + \frac{1}{10} \cdot 0.4\right) 0.5 = \frac{1}{25}$$

With $\sum_j a_2(j) = \frac{8}{75}$,

$$P(X_2 = 1|Z_{1:2}) = \frac{7/250}{8/75} = \frac{21}{80}$$

$$P(X_2 = 2|Z_{1:2}) = \frac{29/750}{8/75} = \frac{29}{80}$$

$$P(X_2 = 3|Z_{1:2}) = \frac{1/25}{8/75} = \frac{3}{8}$$

Therefore the MAP at $t = 2$ is $X_2^* = 3$.
At time $t = 3$, we observe $Z_3 = 1$ and compute transition sums as before,

$$\alpha_3(1) = \left(\frac{7}{250} \cdot 0.2 + \frac{29}{750} \cdot 0.2 + \frac{1}{25} \cdot 0.4\right) 0.6 = \frac{11}{625}$$

$$\alpha_3(2) = \left(\frac{7}{250} \cdot 0.3 + \frac{29}{750} \cdot 0.7 + \frac{1}{25} \cdot 0.2\right) 0.1 = \frac{163}{37500}$$

$$\alpha_3(3) = \left(\frac{7}{250} \cdot 0.5 + \frac{29}{750} \cdot 0.1 + \frac{1}{25} \cdot 0.4\right) 0.2 = \frac{127}{18750}$$

With $\sum_j a_3(j) = \frac{359}{12500}$, we get

$$P(X_3 = 1|Z_{1:3}) = \frac{11/625}{359/12500} = \frac{220}{359}$$

$$P(X_3 = 2|Z_{1:3}) = \frac{163/37500}{359/12500} = \frac{163}{1077}$$

$$P(X_3 = 3|Z_{1:3}) = \frac{127/18750}{359/12500} = \frac{254}{1077}$$

Therefore, the MAP at $t = 3$ is $X_3^* = 1$.

(b) **[6pts]** Determine the maximum a posteriori (MAP) state estimate at each step using smoothing, i.e., $X_t^* = \arg\max_{X_t} P(X_t \mid Z^T)$.

We have the forward pass from above as:

$$\alpha_1(1) = \frac{1}{30}, \alpha_1(2) = \frac{7}{30}, \alpha_1(3) = \frac{1}{10}$$

$$\alpha_2(1) = \frac{7}{250}, \alpha_2(2) = \frac{29}{750}, \alpha_2(3) = \frac{1}{25}$$

$$\alpha_3(1) = \frac{11}{625}, \alpha_3(2) = \frac{163}{37500}, \alpha_3(3) = \frac{127}{18750}$$

The backwards pass is initialized as $\beta_3(i) = 1$ for $i = 1, 2, 3$. We then compute

$$\beta_2(i) = \sum_j^3 P(X_3 = j|X_2 = i)P(Z_3|X_3 = j)\beta_3(j)$$

$$\rightarrow \beta_2(1) = \frac{1}{4}, \beta_2(2) = \frac{21}{100}, \beta_2(3) = \frac{34}{100}$$

Then,

$$\beta_1(1) = \frac{563}{5000}, \beta_1(2) = \frac{307}{5000}, \beta_1(3) = \frac{133}{1250}$$

We can then combine our alphas and betas using smoothing,

$$P(X_t = i|Z_{1:3}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_T(j)\beta_T(j)}$$

$$P(X_1 = 1|Z_{1:3}) = \frac{\frac{1}{30} \cdot \frac{563}{5000}}{\frac{1}{30} \cdot \frac{563}{5000} + \frac{7}{30} \cdot \frac{307}{5000} + \frac{1}{10} \cdot \frac{133}{1250}} = \frac{563}{4308}$$

$$P(X_1 = 2|Z_{1:3}) = \frac{\frac{7}{30} \cdot \frac{307}{5000}}{\frac{1}{30} \cdot \frac{563}{5000} + \frac{7}{30} \cdot \frac{307}{5000} + \frac{1}{10} \cdot \frac{133}{1250}} = \frac{2149}{4308}$$

$$P(X_1 = 3|Z_{1:3}) = \frac{\frac{1}{10} \cdot \frac{133}{1250}}{\frac{1}{30} \cdot \frac{563}{5000} + \frac{7}{30} \cdot \frac{307}{5000} + \frac{1}{10} \cdot \frac{133}{1250}} = \frac{133}{359}$$

which gives us a smoothed MAP estimate at $t = 1$ of $X_1^* = 2$. Then,

$$P(X_2 = 1|Z_{1:3}) = \frac{\frac{7}{250} \cdot \frac{1}{4}}{\frac{7}{250} \cdot \frac{1}{4} + \frac{29}{750} \cdot \frac{21}{100} + \frac{1}{25} \cdot \frac{34}{100}} = \frac{175}{718}$$

$$P(X_2 = 2|Z_{1:3}) = \frac{\frac{29}{750} \cdot \frac{21}{100}}{\frac{7}{250} \cdot \frac{1}{4} + \frac{29}{750} \cdot \frac{21}{100} + \frac{1}{25} \cdot \frac{34}{100}} = \frac{203}{718}$$

$$P(X_2 = 3|Z_{1:3}) = \frac{\frac{1}{25} \cdot \frac{34}{100}}{\frac{7}{250} \cdot \frac{1}{4} + \frac{29}{750} \cdot \frac{21}{100} + \frac{1}{25} \cdot \frac{34}{100}} = \frac{170}{359}$$

This gives us a smoothed MAP estimate at $t = 2$ of $X_2^* = 3$. Finally, at $t = 3$, our $B_3(i) = 1$, which simplifies our calculations:

$$P(X_3 = 1|Z_{1:3}) = \frac{220}{359}, P(X_3 = 2|Z_{1:3}) = \frac{163}{1077}, P(X_3 = 3|Z_{1:3}) = \frac{254}{1077}$$

and gives us a smoothed MAP estimate of $X_3^* = 1$ at $t = 3$.

(c) **[8pts]** Determine the MAP state sequence, i.e., $\{X_1, X_2, X_3\}^*$, using the Viterbi algorithm. Compare the result with the sequence of MAP states from filtering and smoothing.

$$\{X_1, X_2, X_3\}^* = \underset{X_1,X_2,X_3}{\text{arg max}}\; p(X_1, X_2, X_3 \mid Z_1, Z_2, Z_3)$$

We initialize with

$$\delta_1(1) = \frac{1}{3} \cdot 0.1 = \frac{1}{30}, \delta_1(2) = \frac{1}{3} \cdot 0.7 = \frac{7}{30}, \delta_1(3) = \frac{1}{3} \cdot 0.3 = \frac{1}{10}$$

At $t = 2$, we compute

$$\delta_2(1) = P(Z_2|X_2 = 1) \cdot \max_j(P(X_2 = 1|X_1 = j)\delta_1(j)$$

$$= 0.3 \cdot \max(\frac{1}{30} \cdot 0.2, \frac{1}{21} \cdot 0.2, \frac{1}{10} \cdot 0.4) = \frac{7}{500}$$

$$\delta_2(2) = 0.2 \cdot \max(\frac{1}{30} \cdot 0.3, \frac{7}{30} \cdot 0.7, \frac{1}{10} \cdot 0.2) = \frac{49}{1500}$$

$$\delta_2(3) = 0.5 \cdot \max(\frac{1}{30} \cdot 0.5, \frac{7}{30} \cdot 0.1, \frac{1}{10} \cdot 0.4) = \frac{1}{50}$$

We also have

$$Pre_2(1) = 2, Pre_2(2) = 2, Pre_2(3) = 3$$

At $t = 3$, we compute

$$\delta_3(1) = 0.6 \cdot \max(\delta_2(1) \cdot 0.2, \delta_2(2) \cdot 0.2, \delta_2(3) \cdot 0.4) = \frac{3}{625}$$

$$\delta_3(2) = 0.1 \cdot \max(\delta_2(1) \cdot 0.3, \delta_2(2) \cdot 0.7, \delta_2(3) \cdot 0.2) = \frac{343}{150000}$$

$$\delta_3(3) = 0.2 \cdot \max(\delta_2(1) \cdot 0.5, \delta_2(2) \cdot 0.1, \delta_2(3) \cdot 0.4) = \frac{1}{625}$$

With

$$Pre_3(1) = 3, Pre_3(2) = 2, Pre_3(3) = 3$$

At this point, the overall highest probability comes from $\delta_3(1) = \frac{3}{625} \approx 0.0048$, which implies the best final state is $X_3^* = 1$. We can then backtrack with our pointers. $Pre_3(1) = 3$, so the state at $t = 2$ is $X_2^* = 3$. We have $Pre_2(3) = 3$, so the state ate $t = 1$ is $X_1^* = 3$, giving us a joint state sequence of $(3, 3, 1)$.

We see that the filtering and smoothing approaches both yield $(2, 3, 1)$ sequences. In contrast, the Viterbi algorithm, which finds the joint state sequence maximizing the global probability yields $(3, 3, 1)$. This difference probably arises because the filtering and smoothing MAPs pick the most likely state at each time marginally, as independent of the others while the Viterbi sequence is jointly optimal.

4 Hidden Markov Models: Robot Localization [40 pts]

Assume that we have a wheeled robot that moves about in a small environment that consists of a collection of 16 rooms arranged in a 4×4 grid. Each room is painted either red, green, blue, yellow, or black. The robot is free to enter all but the four rooms that are used as storage, which are painted black. Figure 2 depicts this simple environment.

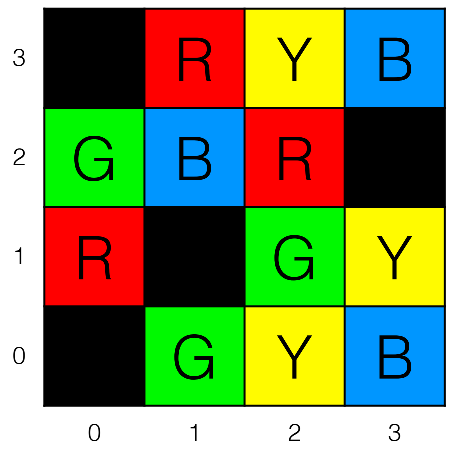


Figure 2: The robot is free to occupy any cell in the 4×4 grid except those colored black.

At each time step, the robot chooses to either stay where it is or move up, down, left, or right (it can not move diagonally). These decisions are made at random and don't depend upon where the robot was previously. If the robot attempts to move to a storage room or outside the confines of the environment, it will draw another action until it chooses one that is valid. When the robot takes the action, it observes the color of the resulting room using its camera. Unfortunately, the white balance is off and the camera may yield an inaccurate observation of the color.

For this problem, the state is the robot's position in the environment, (X, Y) where $X \in \{0, 1, 2, 3\}$ and $Y \in \{0, 1, 2, 3\}$, and the output is the observed color, $Z \in \{r, g, b, y\}$. For example, a ten step random walk through the environment of the robot might look like:

$$\begin{aligned}
t = 0: &\quad (1,3), r\\
t = 1: &\quad (2,3), y\\
t = 2: &\quad (3,3), b\\
t = 3: &\quad (3,3), g\\
t = 4: &\quad (2,3), r\\
t = 5: &\quad (2,3), y\\
t = 6: &\quad (2,3), y\\
t = 7: &\quad (2,3), y\\
t = 8: &\quad (3,3), b\\
t = 9: &\quad (3,3), b
\end{aligned}$$

In this example, the robot starts in the room at $(1, 3)$ where it correctly observes red, moves to the room at $(2, 3)$ and correctly observes yellow, moves to the room at $(3, 3)$ and first correctly observes blue, but after staying in the same room, observes its color as green. Note that it may be advantageous to represent each cell by a number $\{0, 1, \ldots, 15\}$ and use that as the state rather than the coordinate.

We would like to develop an algorithm that estimates the maximum a posteriori (MAP) hidden (latent) trajectory of the robot based upon the sequence of room color observations. As we've seen in lecture, we can model this estimation problem as a hidden Markov model, since the robot's motion obeys the first-order Markov property and the observation likelihood is only a function of the robot's current state. We can then use the forward-backward and Viterbi algorithms to determine the MAP estimates of the robot's state or sequence of states, respectively.

(a) **[20pts]** Formulating this problem as an HMM requires an estimate of the transition, observation, and prior probabilities. While we don't know these, we can estimate them from observation data using the Baum-Welch algorithm. Doing so requires an initial estimate for the HMM parameters. Assume a uniform prior over the initial states; that the robot stays in the current state with likelihood 0.2 and the remaining probability is evenly distributed among valid adjacent states; and that the camera observes the correct color with probability 0.7, with the remaining probability distributed evenly among the other three colors. Starting with this estimate, iterate through successive HMM models, using the (log-)likelihood of the data to determine convergence (i.e., when the change is below a user-defined threshold.) Tip: The $\alpha_t(i)$ and $\beta_t(i)$ terms often become very small. To avoid numerical errors, you should normalize the vector $\boldsymbol{\alpha}_t$ of $\alpha_t(i)$ values at each time step, and use the same normalization factor for the corresponding $\boldsymbol{\beta}_t$ vector

$$\begin{aligned}
C_t &= \sum_{i=1}^{n} \alpha_t(i)\\
\bar{\alpha}_t(i) &= \alpha_t(i)/C_t\\
\bar{\beta}_t(i) &= \beta(i)/C_t
\end{aligned}$$

As part of this problem set, we are providing you with 200 training sequences (random walks) of state-observation pairs, with each sequence consisting of 201 time steps. These sequences are contained in the text file `randomwalk.train.txt`, with sequences separated by a period. Note that you should only be using the observations. We are providing the states so that you can sanity-check your parameter estimates with those determined using knowledge of the true state (be sure to normalize).

(i) Implement the Baum-Welch algorithm as part of the `train` function defined in `HMM.py`, which provides the structure for an HMM class implementation. The `train` function takes as input a set of observation sequences and estimates the transition, emission, and state prior probabilities for the HMM.

Included with this problem set are the file `DataSet.py`, which provides the structure for parsing the sequence data, as well as the file `TrainHMM.py`, which reads in the training data (via the `DataSet` class), calls `HMM.train`, and saves the resulting model as `trained-model.pkl`. The function can be called as follows:

```
$ python TrainHMM.py trainingdata.txt
```

Feel free modify these files or to write your own.

(ii) Plot the (log-)likelihood of the observations (i.e., the (log-)likelihood of the data) as a function of iteration number.

Refer to other submission

(b) **[20pts]** Having identified the likelihoods that define the HMM representation, the next task is to write code that determines the most likely sequence of states (robot locations) based upon the corresponding sequence of observed room colors.

Included as part of this problem set is a second text file `randomwalk.test.txt` that includes 200 sequences of state-observation pairs, with each sequence consisting of 201 time steps. The locations are provided simply for verification—you should only use the sequence of room color observations.

(i) Implement the Viterbi algorithm as part of the `viterbi` function within the `HMM` class. The problem set includes a Python file `RunViterbi.py` that loads in the test data as well as the model generated above, and then calls `HMM.viterbi` on the observations. The code ban be called as:

```
$ python RunViterbi.py testingdata.txt trained-model.pkl
```

Note that `trained-model.pkl` is optional and without it, the function will use the initial model described in Part (a). It may be useful to compare the accuracy when using your learned model to the default model.

(ii) Using the data available in the file `randomwalk.test.txt`, measure the number of times that the ground-truth state is different from the state in your MAP sequence.

Baseline: Accuracy: 57.74 percent, Total Correct: 23213, Total Incorrect: 16987
Refer to other submission

What to hand in: All the code necessary to (a) train the HMM and (b) determine the maximum likelihood state sequence, the data likelihood plot, and the MAP sequence error count.

5 Time and Collaboration Accounting

(a) **[1pts]** Did you work with anyone on this problem set? If so, who?

No

(b) **[1pts]** How long (in hours) did you spend on this problem set (exclusive of going through lecture notes, doing readings, etc.)?

About 10 hours on the written questions and then 7-10 hours on the coding portion (had quite a bit of debugging)