UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Department of Electrical and Computer Engineering

ECE 417 Multimedia Signal Processing
Fall 2023

PRACTICE EXAM 2

Exam will be Tuesday, October 31, 2023

- This will be a CLOSED BOOK exam.
- You will be permitted one sheet of handwritten notes, 8.5x11.
- Calculators and computers are not permitted.
- Don't simplify explicit numerical expressions.
- If you're taking the exam online, you will need to have your webcam turned on. Your exam will appear on Gradescope at exactly 9:30am; you will need to photograph and upload your answers by exactly 11:00am.
- There will be a total of 100 points in the exam. Each problem specifies its point total. Plan your work accordingly.
- You must SHOW YOUR WORK to get full credit.

Name: _____

NetID: _____

Neural Nets

$$a_{i,k}^{(\ell)} = g(\xi_{i,k}^{(\ell)}), \qquad \xi_{i,k}^{(\ell)} = b_k^{(\ell)} + \sum_{j=1}^{p} w_{k,j}^{(\ell)} a_{i,j}^{(\ell-1)}$$

$$\frac{d\mathcal{L}}{d\xi_{i,k}^{(\ell)}} = \frac{d\mathcal{L}}{da_{i,k}^{(\ell)}} \dot{g}(\xi_{i,k}^{(\ell)}), \qquad \dot{\sigma}(x) = \sigma(x)(1-\sigma(x))$$

$$\frac{d\mathcal{L}}{da_{i,j}^{(\ell-1)}} = \sum_k \frac{d\mathcal{L}}{d\xi_{i,k}^{(\ell)}} w_{k,j}^{(\ell)}, \qquad \frac{d\mathcal{L}}{dw_{k,j}^{(\ell)}} = \sum_i \frac{d\mathcal{L}}{d\xi_{i,k}^{(\ell)}} a_{i,j}^{(\ell-1)}$$

$$w_{k,j}^{(\ell)} \leftarrow w_{k,j}^{(\ell)} - \eta \frac{d\mathcal{L}}{dw_{k,j}^{(\ell)}}$$

Viola-Jones

$$II[m,n] = \sum_{m'=1}^{m} \sum_{n'=1^n} I[m',n'], \quad 1 \le m \le M, 1 \le n \le N$$

$$\epsilon_t = \sum_i w_t(x_i)|y_i - h_t(x_i)|, \quad w_{t+1}(x_i) = \beta_t w_t(x_i), \quad \beta_t = \frac{\epsilon_t}{1-\epsilon_t}$$

$$h(x) = \begin{cases} 1 & \sum_t \alpha_t h_t(x) > \frac{1}{2}\sum_t \alpha_t \\ 0 & \text{otherwise} \end{cases}, \quad \alpha_t = -\ln \beta_t$$

Hidden Markov Model

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{i,j} b_j(\boldsymbol{x}_t), \quad 1 \le j \le N, \ 2 \le t \le T, \quad \hat{\alpha}_t(j) = \frac{\sum_{i=1}^{N} \hat{\alpha}_{t-1}(i) a_{i,j} b_j(\boldsymbol{x}_t)}{\sum_{j'=1}^{N} \sum_{i=1}^{N} \hat{\alpha}_{t-1}(i) a_{i,j'} b_{j'}(\boldsymbol{x}_t)}$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{i,j} b_j(\boldsymbol{x}_{t+1}) \beta_{t+1}(j), \quad 1 \le i \le N, \ 1 \le t \le T-1$$

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{k=1}^{N} \alpha_t(k)\beta_t(k)}, \quad \xi_t(i,j) = \frac{\alpha_t(i) a_{i,j} b_j(\boldsymbol{x}_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^{N} \sum_{\ell=1}^{N} \alpha_t(k) a_{k,\ell} b_\ell(\boldsymbol{x}_{t+1}) \beta_{t+1}(\ell)}$$

$$a'_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)}, \quad b'_j(k) = \frac{\sum_{t:x_t=k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Gaussians

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

$$\boldsymbol{\mu}'_i = \frac{\sum_{t=1}^{T} \gamma_t(i) \boldsymbol{x}_t}{\sum_{t=1}^{T} \gamma_t(i)}, \quad \boldsymbol{\Sigma}'_i = \frac{\sum_{t=1}^{T} \gamma_t(i)(\boldsymbol{x}_t - \boldsymbol{\mu}_i)(\boldsymbol{x}_t - \boldsymbol{\mu}_i)^T}{\sum_{t=1}^{T} \gamma_t(i)}$$

1. (10 points) Good days and bad days follow each other with the following probabilities:

q_{t-1}	$p(q_t = G \mid q_{t-1} = \cdot)$	$p(q_t = B \mid q_{t-1} = \cdot)$
G	0.7	0.3
B	0.4	0.6

In winter in Champaign, the temperature on a good day is Gassian with mean $\mu_G = 50$, $\sigma_G = 20$. The temperature on a bad day is Gaussian with mean $\mu_B = 10$, $\sigma_G = 20$. A particular sequence of days has temperatures

$$\{x_1 = 10, x_2 = 20, x_3 = 30\}$$

What is the probability $p(X|q_1 = B)$, the probability of seeing this sequence of temperatures given that the first day was a bad day?

Solution:

$$p(X|q_1 = B) = \left(\frac{1}{50}\right)\left(\frac{1}{20}\right)\left(\frac{1}{20}\right)\left(\frac{6}{25}\right)((0.6)(0.35)(0.6) + (0.4)(0.13)(0.3) + (0.6)(0.35)(0.4) + (0.4)(0.13)(0.7))$$

2. (20 points) Suppose that

$$\begin{aligned} a_{ij} &= p(q_t = j | q_{t-1} = i) \\ b_j(x_t) &= p(x_t | q_t = j) \\ g_t &= p(x_t | x_1, \ldots, x_{t-1}) \end{aligned}$$

And define the scaled forward algorithm to compute

$$\tilde{\alpha}_t(i) = p(q_t = i | x_1, \ldots, x_t) = \frac{p(x_t, q_t = i | x_1, \ldots, x_{t-1})}{g_t} = \frac{p(x_1, \ldots, x_t, q_t = i)}{g_1 g_2 \ldots g_t}$$

(a) Devise an algorithm to iteratively compute g_t and $\tilde{\alpha}_t(i)$. Fill in the right-hand side of each equation, using only the terms a_{jk}, $b_j(x_\tau)$, g_τ, and $\tilde{\alpha}_\tau(j)$ for $1 \leq j \leq N$, $1 \leq k \leq N$, $1 \leq \tau \leq t$.

1. **INITIALIZE:** $g_1 =$
2. **INITIALIZE:** $\tilde{\alpha}_1(i) =$
3. **ITERATE:** $g_t =$
4. **ITERATE:** $\tilde{\alpha}_t(i) =$
5. **TERMINATE:** $p(X) =$

Solution:

1. **INITIALIZE:** $g_1 = \sum_{j=1}^{N} \pi_j b_j(x_1)$
2. **INITIALIZE:** $\tilde{\alpha}_1(i) = \frac{\pi_i b_i(x_1)}{g_1}$
3. **ITERATE:** $g_t = \sum_{i=1}^{N} \sum_{j=1}^{N} \tilde{\alpha}_{t-1}(i) a_{ij} b_j(x_t)$
4. **ITERATE:** $\tilde{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \tilde{\alpha}_{t-1}(i) a_{ij} b_j(x_t)}{g_t}$
5. **TERMINATE:** $p(X) = \prod_{t=1}^{T} g_t$

(b) Suppose $\beta_t(i) = p(x_{t+1}, \ldots, x_T | q_t = i)$. Then

$$\tilde{\alpha}_t(i) \beta_t(i) = p(f | g)$$

for some list of variables f, and some other list of variables g. Specify what variables should be included in each of these two lists.

Solution:

$$f = \{q_t = i, x_{t+1}, \ldots, x_T\}$$
$$g = \{x_1, \ldots, x_t\}$$

3. (10 points) You are creating a recommender system that tries to recommend songs that will be considered to be similar to a given query. Each song is characterized by a two-dimensional vector $\mathbf{x}_k = [b_k, v_k]^T$ where b_k is the number of beats per minute, and v_k is the fraction of air-time during which there is a human voice. Your customer considers the following four songs to be similar:

$$[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4] = \begin{bmatrix} 120 & 140 & 140 & 120 \\ 0.3 & 0.3 & 0.5 & 0.5 \end{bmatrix}$$

You are given two more test data, $\mathbf{x}_5 = [b_5, v_5]^T$ and $\mathbf{x}_6 = [b_6, v_6]^T$, and you are asked whether or not $\mathbf{x}_5$ and $\mathbf{x}_6$ should be considered similar.

Estimate a diagonal data covariance matrix directly from the data, and use it to write the squared Mahalanobis distance $d^2_\Sigma(\mathbf{x}_5, \mathbf{x}_6)$.

Solution:

$$d^2_\Sigma(\mathbf{x}_5, \mathbf{x}_6) = \frac{(b_5 - b_6)^2}{100} + \frac{(v_5 - v_6)^2}{0.01}$$

4. (10 points) The stock market alternates between long bull markets (state 1) and short bear markets (state 2). This HMM has the following parameters:

$$\pi = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad A = \begin{bmatrix} 0.999 & 0.001 \\ 0.005 & 0.995 \end{bmatrix}, \quad \mu_1 = 0.1, \quad \mu_2 = -0.3, \quad \sigma_1^2 = \sigma_2^2 = 1,$$

where $\pi_i = p(q_1 = i)$, $a_{ij} = p(q_{t+1} = j | q_t = i)$, and $p(x_t | q_t = j) = \mathcal{N}(x_t; \mu_j, \sigma_j^2)$.

You observe x_2 on day 2.

For what values of x_2 does the forward algorithm yield probabilities $\alpha_t(i)$ such that $\alpha_2(2) > \alpha_2(1)$?

Asking exactly the same question in different words: for what values of x_2 would it be rational to conclude that a bear market has started?

Solution: $x_2 < -0.1 - 2.5 \ln(999)$

5. (10 points) As you know, in any given vector space, it's possible to define an infinite number of different Mahalanobis distances. Consider the Mahalanobis distance measures $d_a(\mathbf{x}, \mathbf{y})$ and $d_b(\mathbf{x}, \mathbf{y})$, parameterized, respectively, by the covariance matrices

$$\Sigma_a = \begin{bmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & 0 & \vdots \\ \vdots & 0 & \dots & 0 \\ 0 & \dots & 0 & a_d \end{bmatrix}, \quad \Sigma_b = \begin{bmatrix} b_1 & 0 & \dots & 0 \\ 0 & b_2 & 0 & \vdots \\ \vdots & 0 & \dots & 0 \\ 0 & \dots & 0 & b_d \end{bmatrix}$$

Your friend Amit wishes to define a third dissimilarity measure, as

$$d_c(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{1}{2}\left(d_a^2(\mathbf{x}, \mathbf{y}) + d_b^2(\mathbf{x}, \mathbf{y})\right)}$$

Is $d_c(\mathbf{x}, \mathbf{y})$ a Mahalanobis distance? If so, find the elements of the covariance matrix Σ_c in terms of the variables $a_1, \dots, a_d, b_1, \dots, b_d$. If not, demonstrate that no such covariance matrix exists.

Solution:

$$\begin{aligned} d_c^2(\mathbf{x}, \mathbf{y}) &= \frac{1}{2}\left(d_a^2(\mathbf{x}, \mathbf{y}) + d_b^2(\mathbf{x}, \mathbf{y})\right) \\ &= \sum_{i=1}^{d} \frac{1}{2}\left(\frac{(x_i - y_i)^2}{a_i} + \frac{(x_i - y_i)^2}{b_i}\right) \\ &= \sum_{i=1}^{d} \frac{1}{2}\left(\frac{(b_i + a_i)(x_i - y_i)^2}{a_i b_i}\right) \end{aligned}$$

This is a Mahalanobis distance with a covariance matrix Σ_c whose diagonal elements are

$$c_i = \frac{a_i b_i}{a_i + b_c}$$

6. (10 points) You want to classify zoo animals. Your zoo only has two species: elephants and giraffes. There are more elephants than giraffes: if Y is the species,

$$\begin{aligned} p_Y(\text{elephant}) &= \frac{e}{e+1} \\ p_Y(\text{giraffe}) &= \frac{1}{e+1} \end{aligned}$$

where $e = 2.718\ldots$ is the base of the natural logarithm. The height of giraffes is Gaussian, with mean $\mu_G = 5$ meters and variance $\sigma_G^2 = 1$. The height of elephants is also Gaussian, with mean $\mu_E = 3$ and variance $\sigma_E^2 = 1$. Under these circumstances, the minimum probability of error classifier is

$$\hat{y}(x) = \begin{cases} \text{giraffe} & x > \theta \\ \text{elephant} & x < \theta \end{cases}$$

Find the value of θ that minimizes the probability of error.

Solution:

$$p_{Y|X}\left(1 \middle| \begin{bmatrix} x_{10} \\ 0 \end{bmatrix}\right) = \begin{cases} 0 & x_{10} < \frac{3}{4} \\ \frac{1}{3} & \frac{3}{4} < x_{10} < 3 \\ \frac{2}{3} & 3 < x_{10} < \frac{13}{4} \\ 1 & \frac{13}{4} < x_{10} \end{cases}$$

7. (10 points) A particular hidden Markov model is parameterized by $\lambda = \{\pi_i, a_{ij}, b_j(\mathbf{x})\}$ where π_i is uniform ($\pi_i = \frac{1}{N}$). Devise an algorithm to compute $p(q_1 = k|\mathbf{x}_1, \ldots, \mathbf{x}_T, \lambda)$.

Solution: This is solved by the forward-backward algorithm:

$$\alpha_1(j) = \pi_j b_j(\mathbf{x}_1), \quad 1 \leq j \leq N$$

$$\beta_T(i) = 1, \quad 1 \leq j \leq N$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j), \quad 1 \leq i \leq N, \ 1 \leq t \leq T-1$$

$$p(q_1 = k|\mathbf{x}_1, \ldots, \mathbf{x}_T) = \frac{\alpha_1(k)\beta_1(k)}{\sum_{k'=1}^{N} \alpha_1(k')\beta_1(k')}$$

8. (10 points) The scaled forward algorithm is provided for you on the formula page at the beginning of this exam. In terms of the quantities $\pi_i, a_{ij}, b_j(\mathbf{x})$, and/or $\hat{\alpha}_t(j), g_t$, find a formula for the quantity $p(q_{t-1} = i, q_t = j, \mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-2}, \lambda)$.

Solution:

$$p(q_{t-1} = i, q_t = j, \mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-2}, \lambda) = \sum_{k=1}^{N} \hat{\alpha}_{t-2}(k) a_{ki} b_i(\mathbf{x}_{t-1}) a_{ij} b_j(\mathbf{x}_t)$$

9. (10 points) Your training database contains matched pairs $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ where $\mathbf{x}_i$ is the i^{th} observation vector, and $\mathbf{y}_i$ is the i^{th} label vector. For some initial weight matrix $\mathbf{W} = \begin{bmatrix} w_{11} & \cdots \\ \cdots & w_{qp} \end{bmatrix}$, you have already computed the following two quantities:

$$f_\ell(\mathbf{x}_i, \mathbf{W}) \qquad 1 \le \ell \le r,\ 1 \le i \le n \tag{1}$$

$$\frac{\partial f_\ell(\mathbf{x}_i, \mathbf{W})}{\partial w_{kj}} \qquad 1 \le \ell \le r,\ 1 \le k \le q,\ 1 \le j \le p,\ 1 \le i \le n \tag{2}$$

You want to find a new matrix $\mathbf{W}' = \begin{bmatrix} w'_{11} & \cdots \\ \cdots & w'_{qp} \end{bmatrix}$ such that $\mathcal{J}(\mathbf{W}') \ge \mathcal{J}(\mathbf{W})$ (that is, you want to **maximize** $\mathcal{J}$), where

$$\mathcal{J}(\mathbf{W}) = \sum_{i=1}^{n} \sum_{\ell=1}^{r} y_{\ell r} \ln\left(f_\ell(\mathbf{x}_i, W)\right)$$

Give a formula for w'_{kj} in terms of w_{kj}, $f_\ell(\mathbf{x}_i, \mathbf{W})$, $\frac{\partial f_\ell(\mathbf{x}_i, \mathbf{W})}{\partial w_{kj}}$, and in terms of a step size, η, such that for suitable values of η, $\mathcal{J}(\mathbf{W}') \ge \mathcal{J}(\mathbf{W})$.

Solution:

$$w'_{k,j} = w_{k,j} + \eta \sum_{i=1}^{n} \sum_{\ell=1}^{r} \frac{y_{\ell,r}}{f_\ell(\mathbf{x}_i, \mathbf{W})} \frac{\partial f_\ell(\mathbf{x}_i, \mathbf{W})}{\partial w_{k,j}}$$

10. (10 points) An integral image is computed from image $i(x, y)$ according to

$$ii(x, y) = \sum_{x' \leq x} \sum_{y' \leq y} i(x', y')$$

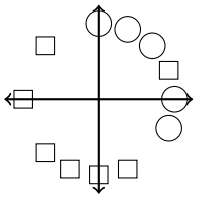
You want to compute the following feature:

$$f(i) = \sum_{y=101}^{200} \left(\sum_{x=51}^{100} i(x, y) - \sum_{x=101}^{150} i(x, y) + \sum_{x=151}^{200} i(x, y) \right)$$

Suppose that $ii(x, y)$ has already been computed. Find a formula for $f(i)$, in terms of $i(x, y)$ and/or $ii(x, y)$, that requires no more than seven additions (a sum with no more than eight terms).

Solution:

$$\begin{aligned} f(i) = {} & ii(200, 200) - 2ii(150, 200) + 2ii(100, 200) - ii(50, 200) \\ & - ii(200, 100) + 2ii(150, 100) - 2ii(100, 100) + ii(50, 100) \end{aligned}$$



11. (10 points) Sun Tzu is surrounded by 12 armies. If Sun Tzu's position is $(0, 0)$, the position of the i^{th} army is given by $(\cos\phi_i, \sin\phi_i)$, where

$$\{\phi_1, \ldots, \phi_{12}\} = \left\{-\frac{\pi}{8}, 0, \frac{\pi}{4}, \frac{3\pi}{8}, \frac{\pi}{2}, \frac{\pi}{8}, \frac{3\pi}{4}, \pi, -\frac{3\pi}{4}, -\frac{5\pi}{8}, -\frac{\pi}{2}, -\frac{3\pi}{8}\right\}$$

The armies shown as circles, in the figure above, are allies; those shown as squares are enemies. Let's label allies as $y_i = 0$, and enemies as $y_i = 1$, so that

$$\{y_1, \ldots, y_{12}\} = \{0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1\}$$

Sun Tzu has two tripod-mounted motion detectors, called $h_1(\phi)$ and $h_2(\phi)$. Each motion detector outputs a 1 whenever an army in its field of view moves. The field of view is exatly π radians, therefore if the army at ϕ_i moves, and if the t^{th} motion director is pointed in the direction θ_t, then

$$h_t(\phi_i) = \begin{cases} 1 & \cos(\theta_t - \phi_i) > 0 \\ 0 & \cos(\theta_t - \phi_i) \leq 0 \end{cases}$$

Unfortunately, when a motion detector goes off, there's no way to tell which of the armies in its field of view has moved. Sun Tzu therefore wants to average the motion detectors in such a way that the average output is positive if and only if an enemy army moves. In other words, the goal is to find θ_1, θ_2, α_1 and α_2 in order to maximize the number of armies for which $\text{sign}(h(x)) = \text{sign}(2y_i - 1)$, where

$$h(x) = \sum_{t=1}^{2} \alpha_t \left(2h_t(\phi_i) - 1\right) \tag{3}$$

Use the AdaBoost algorithm to find values of θ_1, θ_2, α_1 and α_2 that maximize the accuracy of Eq. 3. (Note: θ_1 and θ_2 should be real-valued fractions of π; α_1 and α_2 should each be the logarithm of a real number.)

Solution: The best single classifier is at $\theta_1 = -\frac{7\pi}{8}$, with an error rate of only $\epsilon_1 = \frac{1}{12}$, therefore $\beta_1 = \frac{\epsilon_1}{1-\epsilon_1} = \frac{1}{11}$ and $\alpha_1 = \ln(11)$.

After finding the best classifier, we multiply all of the tokens it correctly classified by β_1, then renormalize the weights so they add up to one, giving $w_{2,6} = \frac{1}{2}$ for the token at $\phi_6 = \frac{\pi}{8}$, and $w_{2,i} = \frac{1}{22}$ for all of the other 11 tokens. The best classifier is now the one that correctly classifies ϕ_6 while also correctly classifying as many other tokens as possible; this is the classifier $\theta_2 = -\frac{5\pi}{16}$, which only makes 4 mistakes, thus $\epsilon_2 = \frac{4}{22} = \frac{2}{11}$, $\beta_2 = \frac{\epsilon_2}{1-\epsilon_2} = \frac{2}{9}$, and $\alpha_2 = \ln(9/2)$.

12. (10 points) Suppose you have a dataset including the vectors

$$\mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 2 \\ 0 \\ 3 \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

Find a diagonal matrix Σ such that $d^2_\Sigma(\mathbf{x}, \mathbf{y}) > d^2_\Sigma(\mathbf{x}, \mathbf{z})$.

Solution: Any solution such that $\frac{1}{\sigma_1^2} > \frac{1}{\sigma_2^2} + \frac{1}{\sigma_3^2}$

13. (20 points) Define $\Phi(z)$ as follows:

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} du$$

Suppose $\mathbf{X} = [X_1, X_2]^T$ is a Gaussian random vector with mean and covariance given by

$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 9 & 0 \\ 0 & 4 \end{bmatrix}$$

(a) Sketch the set of points such that $f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{12\pi} e^{-\frac{1}{8}}$, where $f_{\mathbf{X}}(\mathbf{x})$ is the pdf of $\mathbf{X}$.

Solution: The sketch should show an ellipse with axes parallel to the main axes, passing through the points $(\frac{5}{2}, 0)$, $(-\frac{1}{2}, 0)$, $(1, 1)$, and $(1, -1)$.

(b) In terms of $\Phi(z)$, find the probability $\Pr\{-1 < X_1 < 1, -1 < X_2 < 1\}$.

Solution: $(\Phi(0) - \Phi(-\frac{2}{3}))(\Phi(\frac{1}{2}) - \Phi(-\frac{1}{2}))$

14. (10 points) Suppose that, for a particular classification problem, the correct label of every data point is as follows:

$$y^*(\mathbf{x}) = \begin{cases} 1 & \|\mathbf{x}\|_2 < 1.5 \\ 0 & \|\mathbf{x}\|_2 > 1.5 \end{cases} \tag{4}$$

Unfortunately, you aren't allowed to use the correct labeling function. Instead, you have to try to learn a Bayesian classifier. Suppose that $f_{\mathbf{X}|Y}(\mathbf{x}|0)$ and $f_{\mathbf{X}|Y}(\mathbf{x}|1)$ are both zero-mean Gaussian pdfs, with the covariance matrices Σ_0 and Σ_1 respectfully, where

$$\Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Define η to be the ratio of the class prior probabilities, $\eta = p_Y(0)/p_Y(1)$. Find a value of η such that a Bayesian classifier gives exactly the decision boundary shown in Eq. (4).

Solution: $\eta = \frac{1}{2}e^{9/16}$

15. (10 points) A particular two-layer neural network accepts a two-dimensional input vector $\mathbf{x} = [x_1, x_2, 1]^T$, and generates an output $z = h(\mathbf{v}^T g(\mathbf{U}\mathbf{x}))$. Choose network weights $\mathbf{v}$ and U, and element-wise scalar nonlinearities $h()$ and $g()$, that will generate the following output:

$$z = \begin{cases} 1 & |x_1| < 2 \text{ and } |x_2| < 2 \\ -1 & \text{otherwise} \end{cases}$$

Solution: There are many solutions. One possibility is:

$$\mathbf{U} = \begin{bmatrix} -1 & 0 & 2 \\ 1 & 0 & 2 \\ 0 & -1 & 2 \\ 0 & 1 & 2 \end{bmatrix}$$

$$g(\mathbf{x}) = u(\mathbf{x})$$

$$\mathbf{v}^T = [1, 1, 1, 1]$$

$$h(z) = \begin{cases} 1 & z \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

16. (20 points) A bimodal HMM uses a common state sequence, $Q = [q_1, \ldots, q_T]$, to explain two different observation sequences $X = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ and $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_T]$. The HMM is parameterized by

$$\begin{aligned} \pi_i &= p(q_1 = i) \\ a_{ij} &= p(q_t = j | q_{t-1} = i) \\ b_j(\mathbf{x}_t) &= p_X(\mathbf{x}_t | q_t = j) \\ c_j(\mathbf{y}_t) &= p_Y(\mathbf{y}_t | q_t = j) \end{aligned}$$

Define

$$\begin{aligned} \alpha_t(i) &= p(\mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_t, \mathbf{y}_t, q_t = i) \\ \beta_t(i) &= p(\mathbf{x}_{t+1}, \mathbf{y}_{t+1}, \ldots, \mathbf{x}_T, \mathbf{y}_T | q_t = i) \end{aligned}$$

(a) Specify initialization formulas for $\alpha_1(i)$ and $\beta_T(i)$ in terms of π_i, a_{ij}, $b_j(\mathbf{x}_t)$, and $c_j(\mathbf{y}_t)$.

Solution:

$$\begin{aligned} &\alpha_1(i) = \pi_i b_i(\mathbf{x}_1) c_i(\mathbf{y}_1), \quad 1 \le i \le N \\ &\beta_T(i) = 1, \quad 1 \le i \le N \end{aligned}$$

(b) Specify iteration formulas for $\alpha_t(i)$ and $\beta_t(i)$ in terms of π_i, a_{ij}, $b_j(\mathbf{x}_t)$, $c_j(\mathbf{x}_t)$, $\alpha_{t-1}(j)$, and $\beta_{t+1}(j)$.

Solution:

$$\begin{aligned} \alpha_t(j) &= \sum_{i=1}^{N} \alpha_{t-1}(i) a_{i,j} b_j(\mathbf{x}_t) c_j(\mathbf{y}_t), \quad 1 \le j \le N \\ \beta_t(i) &= \sum_{j=1}^{N} a_{i,j} b_j(\mathbf{x}_{t+1}) c_j(\mathbf{y}_{t+1}) \beta_{t+1}(j), \quad 1 \le i \le N \end{aligned}$$

17. (10 points) Suppose that you have a training database with three training vectors $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ whose correct labels are y_1, y_2, and y_3. You also have a set of three weak classifiers h_1, h_2, and h_3, each of which is right for exactly two of the three training tokens, as follows:

$$h_t(\mathbf{x}_i) = \begin{cases} y_i & i \neq t \\ \text{incorrect} & i = t \end{cases}$$

Adaboost begins with the weights $w_{1,i} = \frac{1}{3}$, and runs for three iterations, resulting in the strong classifier

$$H(\mathbf{x}) = \sum_{t=1}^{3} \alpha_t h_t(\mathbf{x})$$

You may assume that the weak classifiers are selected in order: h_1 is selected in the first iteration of Adaboost, h_2 in the second iteration, and h_3 in the third iteration. Find α_1, α_2, and α_3.

Solution: The first error is $\epsilon_1 = \frac{1}{3}$, so the two correct tokens are multiplied by $\beta_1 = \frac{\epsilon_1}{1-\epsilon_1} = \frac{1}{2}$, then the whole weight vector is renormalized to sum to one, so that

$$\tilde{\mathbf{w}}_2^T = \left[1, \frac{1}{2}, \frac{1}{2}\right]$$
$$\mathbf{w}_2^T = \left[\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right]$$

The second error is therefore $\epsilon_2 = \frac{1}{4}$, so the two correct tokens are multiplied by $\beta_2 = \frac{\epsilon_2}{1-\epsilon_2} = \frac{1}{3}$ giving

$$\tilde{\mathbf{w}}_3^T = \left[\frac{1}{6}, \frac{1}{4}, \frac{1}{12}\right]$$
$$\mathbf{w}_3^T = \left[\frac{1}{3}, \frac{1}{2}, \frac{1}{6}\right]$$

The third error is therefore $\epsilon_3 = \frac{1}{6}$, and therefore $\beta_3 = \frac{\epsilon_3}{1-\epsilon_3} = \frac{1}{5}$. The alphas are therefore

$$\alpha_1 = \ln(2)$$
$$\alpha_2 = \ln(3)$$
$$\alpha_3 = \ln(5)$$

18. (10 points) In terms of $\alpha_t(i)$, $\beta_t(i)$, a_{ij}, π_i and $b_i(\mathbf{x}_t)$, find

$$p\left(q_6 = i, q_7 = j | \mathbf{x}_1, \ldots, \mathbf{x}_{20}\right)$$

Solution:

$$\begin{aligned} p(q_6 = i, q_7 = j, \mathbf{x}_1, \ldots, \mathbf{x}_{20}) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_6, q_6 = i) p(q_7 = j | q_6 = i) p(\mathbf{x}_7 | q_7 = j) p(\mathbf{x}_8, \ldots, \mathbf{x}_{20} | q_7 = j) \\ &= \alpha_6(i) a_{ij} b_j(\mathbf{x}_7) \beta_7(j) \end{aligned}$$

$$p\left(q_6 = i, q_7 = j | \mathbf{x}_1, \ldots, \mathbf{x}_{20}\right) = \frac{\alpha_6(i) a_{ij} b_j(\mathbf{x}_7) \beta_7(j)}{\sum_{k=1}^{N} \sum_{\ell=1}^{N} \alpha_6(k) a_{k\ell} b_\ell(\mathbf{x}_7) \beta_7(\ell)}$$

19. (10 points) A particular HMM-based speech recognizer only knows two words: word w_0, and word w_1. Word w_0 has a higher *a priori* probability: $p_Y(w_0) = 0.7$, while $p_Y(w_1) = 0.3$. Each of the two words is modeled by a four-state Gaussian HMM ($N = 4$) with three-dimensional observations ($D = 3$). All states, in both HMMs, have identity covariance ($\Sigma_i = I$). Both HMMs have *exactly* the same transition probabilities and state-dependent means, given by:

$$\textbf{Both Words: } A = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}, \ \mu_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \ \mu_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \ \mu_3 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \ \mu_4 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

But the initial residence probabilities are different:

$$\textbf{Word 0: } \pi_i = \begin{cases} 1 & i = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{Word 1: } \pi_i = \begin{cases} 1 & i = 4 \\ 0 & \text{otherwise} \end{cases}$$

Suppose that you have a two-frame observation, $X = [\mathbf{x}_1, \mathbf{x}_2]$, where $\mathbf{x}_t = [x_{1t}, x_{2t}, x_{3t}^T]$. The MAP decision rule, in this case, can be written as a linear classifier,

$$\hat{y} = \begin{cases} w_1 & \mathbf{w}_1^T\mathbf{x}_1 + \mathbf{w}_2^T\mathbf{x}_2 + b > 0 \\ w_0 & \text{otherwise} \end{cases}$$

Find $\mathbf{w}_1$, $\mathbf{w}_2$, and b.

Solution: The Bayesian classifier chooses w_1 if

$$\begin{aligned} p(w_0)p(X|w_0) &< p(w_1)p(X|w_1) \\ 0.7\mathcal{N}(\mathbf{x}_1|\mu_1)\sum_j a_{1j}\mathcal{N}(\mathbf{x}_2|\mu_j) &< 0.3\mathcal{N}(\mathbf{x}_4|\mu_4)\sum_j a_{4j}\mathcal{N}(\mathbf{x}_2|\mu_j) \\ 0.7\mathcal{N}(\mathbf{x}_1|\mu_1) &< 0.3\mathcal{N}(\mathbf{x}_1|\mu_4) \\ \ln(0.7) - \frac{1}{2}(\mathbf{x}_1 - \mu_1)^T(\mathbf{x}_1 - \mu_1) &< \ln(0.3) - \frac{1}{2}(\mathbf{x}_1 - \mu_4)^T(\mathbf{x}_1 - \mu_4) \\ \ln(0.7) - \frac{1}{2}\|\mathbf{x}_1\|^2 &< \ln(0.3) - \frac{1}{2}\|\mathbf{x}_1\|^2 + \mu_4^T\mathbf{x}_1 - \frac{1}{2}\|\mu_4\|^2 \end{aligned}$$

Which is satisfied if

$$\mu_4^T\mathbf{x}_1 + \ln\left(\frac{3}{7}\right) - \frac{3}{2} > 0$$

So

$$\mathbf{w}_1 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad b = \ln\left(\frac{3}{7}\right) - \frac{3}{2}$$

20. (10 points) In class, we have been working with the exponential softmax function, but other forms exist. For example, the polynomial softmax function transforms inputs b_ℓ into outputs z_ℓ according to

$$z_\ell = \frac{b_\ell^p}{\sum_k b_k^p},$$

for some constant integer power, p. The cross-entropy loss is

$$E = -\sum_\ell \zeta_\ell \ln z_\ell, \qquad \zeta_\ell = \begin{cases} 1 & \ell = \ell^* \\ 0 & \text{otherwise} \end{cases}$$

Find $\frac{\partial E}{\partial b_j}$ for all j.

Solution:

$$\begin{aligned} \frac{\partial E}{\partial b_j} &= -\sum_\ell \frac{\zeta_\ell}{z_\ell}\left(\frac{p b_j^{p-1}\delta_{j\ell}}{\sum_k b_k^p} - \frac{p b_\ell^p b_j^{p-1}}{\left(\sum_k b_k^p\right)^2}\right) = -\sum_\ell \frac{p}{b_j}\zeta_\ell\left(\delta_{j\ell} - z_j\right) \\ &= -\frac{p}{b_j}\left(\delta_{j\ell^*} - z_j\right) = -\frac{p}{b_j}\left(\zeta_j - z_j\right) \end{aligned}$$

21. (10 points) Suppose you have a 10-pixel input image, $x[n]$. This is processed by a one-pixel "convolution" (really just multiplication by a scalar coefficient, w), followed by a stride-2 max pooling layer, thus:

$$a[n] = wx[n], \quad 1 \leq n \leq 10$$
$$y[k] = \max\left(0, \max_{2k-1 \leq n \leq 2k} a[n]\right), 1 \leq k \leq 5$$

Suppose you know the input $x[n]$, and you know $\epsilon[k] = \frac{\partial E}{\partial y[k]}$. Find $\frac{\partial E}{\partial w}$ in terms of $x[n]$ and $\epsilon[k]$.

Solution:

$$\frac{\partial E}{\partial w} = \sum_{k=1}^{5} \epsilon[k] x \left[\operatorname*{argmax}_{2k-1 \leq n \leq 2k} a[n]\right]$$

22. (10 points) Suppose that you have a training dataset with n training tokens $\{(\mathbf{x}_1, \zeta_1), \ldots, (\mathbf{x}_n, \zeta_n)\}$, where $\mathbf{x}_i = [x_{i1}, \ldots, x_{ip}]^T$, and $\zeta_i \in \{0, 1\}$. You have a one-layer neural network that tries to approximate ζ_i with z_i, computed as $z_i = \sigma\left(\mathbf{w}^T\mathbf{x}_i\right)$, where $\sigma(\cdot)$ is the logistic function, and $\mathbf{w}$ is a weight vector. Suppose that you want to maximize the accuracy of z_i, but you also want to make $\mathbf{w}^T\mathbf{x}_i$ as small as possible. One way to do this is by using a two-part error metric,

$$E = -\frac{1}{n}\sum_{i=1}^{n}\left(\zeta_i \ln z_i + (1-\zeta_i)\ln(1-z_i)\right) + \frac{1}{2n}\sum_{i=1}^{n}\left(\mathbf{w}^T\mathbf{x}_i\right)^2$$

Find $\nabla_{\mathbf{w}} E$, the gradient of E with respect to $\mathbf{w}$.

Solution:

$$\nabla_{\mathbf{w}} E = \frac{1}{n}\sum_{i=1}^{n}\left((1-\zeta_i)z_i - \zeta_i(1-z_i) + \mathbf{w}^T\mathbf{x}_i\right)\mathbf{x}_i^T$$

23. (20 points) An HMM has initial-value probabilities $\pi_i = p(s_1 = i)$, transition probabilities $a_{ij} = p(s_{t+1} = j | s_t = i)$, and observation probabilities $b_j(\mathbf{x}_t) = p(\mathbf{x}_t | s_t = j)$. The forward algorithm is defined as

$$\alpha_1(i) = \pi_i b_i(\mathbf{x}_1), \quad 1 \le i \le N \tag{5}$$

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(\mathbf{x}_{t+1}), \quad 1 \le i, j \le N, \;\; 1 \le t \le T-1 \tag{6}$$

The Viterbi algorithm is defined as

$$\delta_1(i) = \pi_i b_i(\mathbf{x}_1), \quad 1 \le i \le N \tag{7}$$

$$\delta_{t+1}(j) = \max_{i=1}^{N} \delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1}), \quad 1 \le i, j \le N, \;\; 1 \le t \le T-1 \tag{8}$$

Assume that $a_{ij} = 0$ for all values of i except $i = j$ and $i = j - 1$.

(a) Prove that, for any particular value of t, the following implication is true:

$$\delta_t(i) \le \alpha_t(i) \; \forall i \in \{1, \ldots, N\} \Rightarrow \delta_{t+1}(j) \le \alpha_{t+1}(j) \; \forall j \in \{1, \ldots, N\} \tag{9}$$

Solution:

$$\begin{aligned} \delta_{t+1}(j) &= \max_{i=1}^{N} \delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1}) \\ &\le \sum_{i=1}^{N} \delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1}) \\ &\le \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(\mathbf{x}_{t+1}) \\ &= \alpha_{t+1}(j) \end{aligned}$$

(b) Prove that $p(X) \ge \max_S p(X, S)$, where $X = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$, and $S = [s_1, \ldots, s_T]$. You may prove this by induction using Eq. 9, or from first principles.

Solution:

$$\begin{aligned} p(X, S) &= \max_{i=1}^{N} \delta_T(i) \\ &\le \sum_{i=1}^{N} \delta_T(i) \\ &\le \sum_{i=1}^{N} \alpha_T(i) \\ &= p(X) \end{aligned}$$

The argument from first principles simply notes that

$$\begin{aligned} p(X) &= \sum_{S} p(X, S) \\ &\ge p(X, S) \end{aligned}$$

24. (10 points) You are given the integral image $ii[n_1, n_2]$, defined in terms of the image $i[n_1, n_2]$ as

$$ii[n_1, n_2] = \sum_{m_1=0}^{n_1} \sum_{m_2=0}^{n_2} i[m_1, m_2]$$

Write an equation that computes the complementary integral image, $c[n_1, n_2]$, in a small constant number of operations per output pixel, where

$$c[n_1, n_2] = \sum_{m_1=n_1}^{N_1-1} \sum_{m_2=n_2}^{N_2-1} i[m_1, m_2]$$

Solution:

$$\begin{aligned} c[n_1, n_2] &= \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} i[m_1, m_2] - \sum_{m_1=0}^{n_1-1} \sum_{m_2=0}^{N_2-1} i[m_1, m_2] - \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{n_2-1} i[m_1, m_2] + \sum_{m_1=0}^{n_1-1} \sum_{m_2=0}^{n_2-1} i[m_1, m_2] \\ &= ii[N_1 - 1, N_2 - 1] - ii[n_1 - 1, N_2 - 1] - ii[N_1 - 1, n_2 - 1] + ii[n_1 - 1, n_2 - 1] \end{aligned}$$

25. (30 points) The Maesters of the Citadel need to determine when winter starts. The temperature on day t is x_t. The state of day t is either $q_t = 0$ (Autumn) or $q_t = 1$ (Winter). Nobody really knows how cold this winter will be or how long it will last, but the Maesters have created an initial model $\Lambda = \{a_{ij}, b_j(x)\}$ where $a_{ij} \equiv p(q_t = j|q_{t-1} = i)$ and $b_j(x) \equiv p(x_t = x|q_t = j)$.

(a) Suppose we have a particular three day sequence of measurements, x_1, x_2, and x_3. Given that the preceding day was still autumn ($q_0 = 0$), we want to determine the joint probability that it continued to be autumn for days 1, 2, and 3, and that the three observed temperatures were measured. In other words, we want an estimate of

$$G_1 = p(q_1 = 0, x_1, q_2 = 0, x_2, q_3 = 0, x_3|q_0 = 0, \Lambda)$$

Find G_1 in terms of a_{ij} and $b_j(x_t)$, for whatever particular values of i, j, and t are most useful to you.

Solution:

$$G_1 = a_{00}^3 b_0(x_1)b_0(x_2)b_0(x_3)$$

(b) Suppose it is known that the preceding day was still autumn ($q_0 = 0$). Now, on day 1, the Maesters have determined that the temperature is x_1. Find the conditional probability, given this measurement, that it is still autumn, i.e., find

$$G_2 = p(q_1 = 0|x_1, q_0 = 0, \Lambda)$$

Find G_2 in terms of a_{ij} and $b_j(x_t)$, for whatever particular values of i, j, and t are most useful to you.

Solution:

$$G_2 = \frac{p(q_1 = 0, x_1|q_0 = 0, \Lambda)}{\sum_i p(q_1 = i, x_1|q_0 = 0, \Lambda)} = \frac{a_{00}b_0(x_1)}{a_{00}b_0(x_1) + a_{01}b_1(x_1)}$$

(c) The Maesters have collected a long series of measurements, $\{x_1, \ldots, x_T\}$ for T consecutive days. From these measurements, the Maesters have applied the forward-backward algorithm in order to calculate the following two quantities:

$$\alpha_t(i) \equiv p(x_1, \ldots, x_t, q_t = i|\Lambda), \quad \beta_t(i) \equiv p(x_{t+1}, \ldots, x_T|q_t = i, \Lambda)$$

Using these quantities, the Maesters wish to calculate the probability that Winter started on a particular day, $t = w$. That is, they wish to find

$$G_3 = p(q_{w-1} = 0, q_w = 1|x_1, \ldots, x_T, \Lambda)$$

Find G_3 in terms of $\alpha_t(i)$, $\beta_t(i)$, a_{ij} and $b_j(x_t)$, for whatever particular values of i, j, and t are most useful to you.

Solution:

$$G_3 = \frac{p(q_{w-1} = 0, q_w = 1, x_1, \ldots, x_T|\Lambda)}{\sum_i \sum_j p(q_{w-1} = i, q_w = j, x_1, \ldots, x_T|\Lambda)} = \frac{\alpha_{w-1}(0)a_{01}b_1(x_w)\beta_w(1)}{\sum_i \sum_j \alpha_{w-1}(i)a_{ij}b_j(x_w)\beta_w(j)}$$

26. (20 points) Suppose we're trying to predict the sequence $\zeta_1, \ldots, \zeta_{100}$ from the sequence $x_1, \ldots, x_{100}$. We want to use some type of neural net (fully-connected or CNN) to compute $z_1, \ldots, z_{100}$ in order to minimize the error

$$E = \frac{1}{200} \sum_{t=1}^{100} (z_t - \zeta_t)^2$$

We only have one training sequence $(x_1, \ldots, x_{100}, \zeta_1, \ldots, \zeta_{100})$.

(a) Suppose we use a **fully-connected one-layer neural net**, with 10,000 trainable network weights w_{kj}, and 100 trainable bias terms w_{k0}, such that

$$z_k = \sigma \left(w_{k0} + \sum_{j=1}^{100} w_{kj} x_j \right)$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the logistic nonlinearity. Find the derivatives of the error with respect to the weights (dE/dw_{kj}) and biases (dE/dw_{k0}). Express your answers in terms of x_j, z_k, and ζ_k for appropriate values of k and j; **the terms w_{kj} and w_{k0} should not show up on the right-hand-side of any of your equations**.

Solution:

$$\frac{dE}{dw_{k0}} = \frac{1}{100}(z_k - \zeta_k) z_k (1 - z_k)$$
$$\frac{dE}{dw_{kj}} = \frac{1}{100}(z_k - \zeta_k) z_k (1 - z_k) x_j$$

(b) Suppose we use a **CNN (convolutional neural net)** with 99 trainable weights $w[\tau]$ and a single scalar bias term, b, i.e.,

$$z_t = \sigma \left(b + \sum_{\tau=-49}^{49} w[\tau] x_{t-\tau} \right)$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the logistic nonlinearity. Find the derivatives of the error with respect to the weights ($dE/dw[\tau]$) and bias (dE/db). Assume that $x_t = 0$ for $t \leq 0$ or $t \geq 101$. Express your answers in terms of x_j, z_k, and ζ_k for appropriate values of k and j; **the terms $w[\tau]$ and b should not show up on the right-hand-side of any of your equations**.

Solution:

$$\frac{dE}{db} = \frac{1}{100} \sum_{t=1}^{100} (z_t - \zeta_t) z_t (1 - z_t)$$
$$\frac{dE}{dw[\tau]} = \frac{1}{100} \sum_{t=1}^{100} (z_t - \zeta_t) z_t (1 - z_t) x_{t-\tau}$$

27. (10 points) Suppose $\mathbf{X} = [X_1, X_2]^T$ is a Gaussian random variable with mean and covariance matrix given by

$$\mu = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

Sketch the set of points such that $p_{\mathbf{X}}(\mathbf{x}) = \frac{1}{4\pi} e^{-\frac{1}{2}}$, where $p_{\mathbf{X}}(\mathbf{x})$ is the pdf of $\mathbf{X}$. Clearly label at least four of the points included in this set.

Solution: The set should be an ellipse, centered at $[3, 1]^T$. The following four points are included in the set:

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \quad \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

28. (10 points) The binary random variable Y has the following prior distribution:

$$p_Y(0) = a, \quad p_Y(1) = 1 - a$$

The random vector $\mathbf{X}$ depends on Y, with the conditionally Gaussian pdf $p_{\mathbf{X}|Y}(\mathbf{x}|y) = \mathcal{N}(\mathbf{x}; \mu_y, \Sigma_y)$, where the mean vectors and covariance matrices are given by

$$\mu_0 = \begin{bmatrix} b \\ c \end{bmatrix}, \quad \mu_1 = \begin{bmatrix} d \\ e \end{bmatrix}, \quad \Sigma_0 = \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Given a sample measurement of $\mathbf{X} = \mathbf{x}$, it's possible to infer the value of $Y = \hat{y}$ with minimum probability of error using the following decision rule:

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + \beta > 0 \\ 0 & \text{otherwise} \end{cases}$$

Find $\mathbf{w}$ and β in terms of the constants a, b, c, d, e.

Solution:

$$\mathbf{w} = \begin{bmatrix} d - b \\ e - c \end{bmatrix}$$

$$\beta = \ln(1 - a) - \ln(a) - \frac{1}{2}(d^2 + e^2) + \frac{1}{2}(b^2 + c^2)$$

29. (10 points) An HMM has the parameters $\Lambda = \{\pi_i, a_{i,j}, b_j(\mathbf{x}_t) : 1 \le i, j \le N, 1 \le t \le T\}$, with the standard definitions:

$$\begin{aligned} \pi_i &= p(q_1 = i) \\ a_{i,j} &= p(q_t = j | q_{t-1} = i) \\ b_j(\mathbf{x}_t) &= p(\mathbf{x} = \mathbf{x}_t | q_t = j), \end{aligned}$$

where q_t is the state index at time t. Suppose you have software available that will compute the forward, backward, scaled forward, and/or scaled backward algorithm for you, and will therefore provide you with any or all of the following quantities, for any values of $1 \le i, j \le N$ and $1 \le t \le T$:

$$\begin{aligned} \alpha_t(i) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = i | \Lambda) \\ \beta_t(i) &= p(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_T | q_t = i, \Lambda) \\ \hat{\alpha}_t(i) &= p(q_t = i | \mathbf{x}_1, \ldots, \mathbf{x}_t, \Lambda) \\ g_t &= p(\mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \Lambda) \\ \hat{\beta}_t(i) &= \beta_t(i) / \max_j \beta_t(j) \end{aligned}$$

In terms of $\pi_i, a_{ij}, b_j(\mathbf{x}_t), \alpha_t(i), \beta_t(i), \hat{\alpha}_t(i), g_t$, and/or $\hat{\beta}_t(i)$, find a formula for the following quantity, assuming that T is much larger than 19:

$$p(q_{16} = 4, q_{17} = 5 | \mathbf{x}_1, \ldots, \mathbf{x}_{18}, \Lambda)$$

Solution:

$$\begin{aligned} p(q_{16} = 4, q_{17} = 5 | \mathbf{x}_1, \ldots, \mathbf{x}_{18}, \Lambda) &= \frac{p(q_{16} = 4, q_{17} = 5, \mathbf{x}_{17}, \mathbf{x}_{18} | \mathbf{x}_1, \ldots, \mathbf{x}_{16}, \Lambda)}{p(\mathbf{x}_{17}, \mathbf{x}_{18} | \mathbf{x}_1, \ldots, \mathbf{x}_{16}, \Lambda)} \\ &= \frac{\sum_{k=1}^N \hat{\alpha}_{16}(4) a_{4,5} b_5(\mathbf{x}_{17}) a_{5,k} b_k(\mathbf{x}_{18})}{\sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \hat{\alpha}_{16}(i) a_{i,j} b_j(\mathbf{x}_{17}) a_{j,k} b_k(\mathbf{x}_{18})} \\ &= \frac{\sum_{k=1}^N \hat{\alpha}_{16}(4) a_{4,5} b_5(\mathbf{x}_{17}) a_{5,k} b_k(\mathbf{x}_{18})}{g_{17} g_{18}} \end{aligned}$$

The last two lines are just three different valid ways to write the denominator. Other valid solutions include

$$\begin{aligned} p(q_{16} = 4, q_{17} = 5 | \mathbf{x}_1, \ldots, \mathbf{x}_{18}, \Lambda) &= \frac{p(q_{16} = 4, q_{17} = 5, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_1, \ldots, \mathbf{x}_{16} | \Lambda)}{p(\mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_1, \ldots, \mathbf{x}_{16} | \Lambda)} \\ &= \frac{\sum_{k=1}^N \alpha_{16}(4) a_{4,5} b_5(\mathbf{x}_{17}) a_{5,k} b_k(\mathbf{x}_{18})}{\sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \alpha_{16}(i) a_{i,j} b_j(\mathbf{x}_{17}) a_{j,k} b_k(\mathbf{x}_{18})} \\ &= \frac{\sum_{k=1}^N \alpha_{16}(4) a_{4,5} b_5(\mathbf{x}_{17}) a_{5,k} b_k(\mathbf{x}_{18})}{\sum_{k=1}^N \alpha_{18}(k)} \end{aligned}$$

30. (10 points) A second-order HMM is like a standard HMM, except that the state at each time step depends on the two preceding states. The parameters are $\Lambda = \{\pi_{i,j}, a_{i,j,k}, b_k(\mathbf{x}_t) : 1 \leq i, j, k \leq N, 1 \leq t \leq T\}$, with the definitions:

$$\begin{aligned} \pi_{i,j} &= p(q_1 = i, q_2 = j) \\ a_{i,j,k} &= p(q_t = k | q_{t-2} = i, q_{t-1} - j) \\ b_k(\mathbf{x}_t) &= p(\mathbf{x} = \mathbf{x}_t | q_t = k), \end{aligned}$$

where q_t is the state index at time t. Suppose you have software available that will compute the forward and backward algorithms for you, and will therefore provide you with the following quantities, for any values of $1 \leq i, j \leq N$ and $2 \leq t \leq T$:

$$\begin{aligned} \alpha_t(i,j) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_{t-1} = i, q_t = j | \Lambda) \\ \beta_t(i,j) &= p(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_T | q_{t-1} = i, q_t = j, \Lambda) \end{aligned}$$

In terms of $\pi_{i,j}, a_{i,j,k}, b_k(\mathbf{x}_t), \alpha_t(i,j)$, and/or $\beta_t(i,j)$, find the following expected value:

$$\mathbb{E}\left[\#\text{ times, } t, \text{ for which } q_{t-2} = i, q_{t-1} = j, q_t = k | \mathbf{x}_1, \ldots, \mathbf{x}_T, \Lambda\right]$$

Solution:

$$\begin{aligned} &\mathbb{E}\left[\#\text{ times that } q_{t-2} = i, q_{t-1} = j, q_t = k | \mathbf{x}_1, \ldots, \mathbf{x}_T, \Lambda\right] \\ &= \sum_{t=3}^{T} p\left(q_{t-2} = i, q_{t-1} = j, q_t = k | \mathbf{x}_1, \ldots, \mathbf{x}_T, \Lambda\right) \\ &= \sum_{t=3}^{T} \alpha_{t-1}(i,j) a_{i,j,k} b_k(\mathbf{x}_t) \beta_t(j,k) \end{aligned}$$

31. (10 points) Suppose you are studying the running behaviors of trained vs. untrained athletes. You have a sequence of feature vectors $\mathbf{x}_t$, where t is time (measured in centiseconds) and $\mathbf{x}_t$ is a vector of features computed from a motion sensor being worn at the ankle. You have trained a neural network to compute $b_j(\mathbf{x}_t) = p(\mathbf{x}_t|q_t = j)$, where $q_t \in \{1 = \text{heel strike}, 2 = \text{roll}, 3 = \text{lift}, 4 = \text{swing}\}$ denotes the gait phase. You also know the following probabilities:

$$\begin{aligned} a_{i,j} &= p(q_t = j|q_{t-1} = i) \\ \alpha_t(i) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = i) \\ \beta_t(i) &= p(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_T|q_t = i) \end{aligned}$$

Your goal is to identify all of the instants when the heel first touches the ground, i.e., at each time step τ $(1 \leq \tau \leq T)$, you want to find

$$P_{HS}(\tau) = p(q_{\tau-1} = 4, q_\tau = 1|\mathbf{x}_1, \ldots, \mathbf{x}_T)$$

Write a formula for $P_{HS}(\tau)$ in terms of $\alpha_t(i)$, $\beta_t(i)$, $a_{i,j}$, and $b_i(\mathbf{x}_t)$, for any values of i, j, t that you find useful.

Solution:

$$P_{HS}(\tau) = \frac{\alpha_{\tau-1}(4)a_{4,1}b_1(\mathbf{x}_\tau)\beta_\tau(1)}{\sum_{i=1}^{4}\sum_{j=1}^{4}\alpha_{\tau-1}(i)a_{i,j}b_j(\mathbf{x}_\tau)\beta_\tau(j)}$$

32. (10 points) In a neural network with residual connections (ResNet), the k^{th} activation at layer ℓ, $h_k^{(\ell)}$, is equal to the activation of the same node at the previous layer, plus a computed residual $g(\xi_k^{(\ell)})$:

$$\xi_k^{(\ell)} = \sum_{j=1}^{N} w_{k,j}^{(\ell)} h_j^{(\ell-1)}, \quad 1 \le k \le N,$$
$$h_k^{(\ell)} = h_k^{(\ell-1)} + g\left(\xi_k^{(\ell)}\right), \quad 1 \le k \le N,$$

where $g(\cdot)$ is a scalar nonlinearity, and $w_{k,j}^{(\ell)}$ is a network weight. Suppose that the training loss is $\mathcal{L}$, and suppose you already know $\frac{d\mathcal{L}}{dh_k^{(\ell)}}$. Find $\frac{d\mathcal{L}}{dh_j^{(\ell-1)}}$ in terms of $\frac{d\mathcal{L}}{dh_k^{(\ell)}}$, $\dot{g}(\xi) = \frac{\partial g}{\partial \xi_k^{(\ell)}}$, and $w_{k,j}^{(\ell)}$.

Solution: The total derivative rule gives us

$$\begin{aligned}\frac{d\mathcal{L}}{dh_j^{(\ell-1)}} &= \sum_{k=1}^{N} \frac{d\mathcal{L}}{dh_k^{(\ell)}} \frac{\partial h_k^{(\ell)}}{\partial h_j^{(\ell)}} \\ &= \frac{d\mathcal{L}}{dh_j^{(\ell)}} + \sum_{k=1}^{N} \frac{d\mathcal{L}}{dh_k^{(\ell)}} \dot{g}(\xi_k^{(\ell)}) w_{k,j}^{(\ell)}\end{aligned}$$

33. (10 points) An RBF-softmax is similar to a regular softmax nonlinearity, but instead of being a generalization of the logistic sigmoid, it is a generalization of a nonlinearity called a **radial basis function** (RBF), which is a kind of simplified Gaussian. An RBF-softmax has the following form:

$$\hat{y}_k = \frac{w_k e^{-\|\mathbf{x}-\mu_k\|^2}}{\sum_{\ell=1}^{N} w_\ell e^{-\|\mathbf{x}-\mu_\ell\|^2}},$$

where $\mathbf{x} = [x_1, \ldots, x_D]^T$ is the input vector, $\hat{y}_k$ is the k^{th} output, and w_k and $\mu_k = [\mu_{1,k}, \ldots, \mu_{D,k}]^T$, for $1 \leq k \leq K$, are trainable parameters.

Find $\frac{d\hat{y}_k}{dw_j}$ for all $j \in \{1, \ldots, K\}$. Your answer may contain any of the variables used in the problem statement. Your answer should not include any unresolved derivatives.

Solution:

$$\begin{aligned}
\frac{d\hat{y}_k}{dw_j} &= \frac{e^{-\|\mathbf{x}-\mu_k\|^2}}{\sum_{\ell=1}^{N} w_\ell e^{-\|\mathbf{x}-\mu_\ell\|^2}} 1\,[k=j] - \frac{e^{-\|\mathbf{x}-\mu_k\|^2}}{\left(\sum_{\ell=1}^{N} w_\ell e^{-\|\mathbf{x}-\mu_\ell\|^2}\right)^2} e^{-\|\mathbf{x}-\mu_j\|^2} \\
&= \begin{cases} \frac{1}{w_k}\hat{y}_k(1-\hat{y}_k) & k=j \\ -\frac{1}{w_k}\hat{y}_k\hat{y}_j & \text{otherwise} \end{cases}
\end{aligned}$$

34. (10 points) A particular CNN has a grayscale image input, $x[n_1, n_2]$, and a one-channel output:

$$\xi[n_1, n_2] = w[n_1, n_2] * x[n_1, n_2],$$

where $*$ denotes convolution. The output is then max-pooled over the entire image:

$$\hat{y} = \max_{0 \le n_1 < N_1} \max_{0 \le n_2 < N_2} \xi[n_1, n_2]$$

Suppose the weights and the input image are given by

$$w[n_1, n_2] = \begin{cases} e^{-(n_1^2 + n_2^2)} & -3 \le n_1 \le 3, \ -3 \le n_2 \le 3 \\ 0 & \text{otherwise} \end{cases}$$

$$x[n_1, n_2] = \begin{cases} e^{-((n_1-15)^2 + (n_2-12)^2)} & 0 \le n_1 \le 63, \ 0 \le n_2 \le 63 \\ 0 & \text{otherwise} \end{cases}$$

What is $\frac{d\hat{y}}{dw[2,1]}$? Your answer should be an explicit function of numerical constants; there should not be any variables in your answer.

Solution:

$$\frac{d\hat{y}}{dw[2,1]} = e^{-5}$$

35. (10 points) Sometimes, it's not obvious, in advance, what loss function should be used to train a neural network. For example, suppose that we have a training database containing vector triples of the form $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Suppose we know that the set of vectors, $\mathbf{x}$, can be divided in half through the origin such that for half of the vectors, $\mathbf{y}$ is a linear transformation of $\mathbf{x}$, while for the other half, $\mathbf{z}$ is a linear transformation of $\mathbf{x}$. In other words, for some matrices U_{ideal} and V_{ideal} that we don't know, and for some vector $\mathbf{w}_{\text{ideal}}$ that we don't know:

- If $\mathbf{w}_{\text{ideal}}^T \mathbf{x} \geq 0$ then $\mathbf{y} = U_{\text{ideal}} \mathbf{x}$.
- If $\mathbf{w}_{\text{ideal}}^T \mathbf{x} < 0$ then $\mathbf{z} = V_{\text{ideal}} \mathbf{x}$.

Devise a differentiable non-negative loss function, $\mathcal{L}$, that will approach zero as the estimated values of $\mathbf{w}, U$, and V approach their true values. Write your loss as a function of the estimated parameters $\mathbf{w}, U$, and V, and as a function of the vectors in just one data triple, $(\mathbf{x}, \mathbf{y}, \mathbf{z})$.

Solution: First, we want differentiable functions of U and V that will be minimized when $\mathbf{y} = U\mathbf{x}$ and $\mathbf{z} = V\mathbf{x}$. Most of the functions that do this are norms of the vectors $(\mathbf{y} - U\mathbf{x})$ and $(\mathbf{z} - V\mathbf{x})$, for example, the squared L2 norms, $\|\mathbf{y} - U\mathbf{x}\|^2$ and $\|\mathbf{z} - V\mathbf{x}\|^2$, are good choices.

Second, we want to multiply $\|\mathbf{y} - U\mathbf{x}\|^2$ by some modifier that goes to zero when $\mathbf{w}^T\mathbf{x} < 0$. The unit step function would do the trick, but it's not differentiable; we need something that can be differentiated. The ReLU nonlinearity will do the trick:

$$\mathcal{L} = \text{ReLU}(\mathbf{w}^T\mathbf{x})\|\mathbf{y} - U\mathbf{x}\|^2 + \text{ReLU}(-\mathbf{w}^T\mathbf{x})\|\mathbf{z} - V\mathbf{x}\|^2$$

The sigmoid is also a good choice. It doesn't go to zero immediately when $\mathbf{w}^T\mathbf{x} < 0$, but it goes to zero when $\mathbf{w}^T\mathbf{x} \ll 0$. Since the problem specification doesn't actually dictate the norm of $\mathbf{w}$ (it can be any scalar times $\mathbf{w}_{\text{ideal}}$, and still meet the problem specifications), the sigmoid will also work here:

$$\mathcal{L} = \sigma(\mathbf{w}^T\mathbf{x})\|\mathbf{y} - U\mathbf{x}\|^2 + \sigma(-\mathbf{w}^T\mathbf{x})\|\mathbf{z} - V\mathbf{x}\|^2$$

36. (10 points) Suppose y is a scalar continuous piece-wise linear function of the scalar variable x, with

$$\frac{dy}{dx} = \begin{cases} 0 & x < x_0 \\ s_i & x_i \leq x < x_{i+1}, \quad 0 \leq i < N \\ s_N & x_N \leq x \end{cases}$$

This function, $y(x)$, can be exactly represented by a ReLU neural network of the form

$$y(x) = \sum_{i=0}^{N} w_i \text{ReLU}(x + b_i)$$

Find w_i and b_i, for all $0 \leq i \leq N$, in terms of s_j and x_j, for any $0 \leq j \leq N$ that you find to be useful.

Solution: We know that

$$\text{ReLU}(x + b_i) = \begin{cases} x + b_i & x + b_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

So we can get the breakpoints exactly right by setting

$$b_i = -x_i$$

Setting the slopes equal, we get that

$$s_i = \sum_{j=0}^{i} w_j$$

which can be inverted to find that

$$\begin{aligned} w_0 &= s_0 \\ w_i &= s_i - s_{i-1}, \quad 1 \leq i \leq N \end{aligned}$$